ISSN 2464–4617 (print) ISSN 2464–4625 (CD/DVD)

CSRN 3101

(Eds.)

- **Vaclav Skala**
  University of West Bohemia, Czech Republic

Computer Science Research Notes

29. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2021
Plzen, Czech Republic
May 17 – 20, 2021

Proceedings

WSCG 2021

Proceedings

ISSN 2464–4617 (print) ISSN 2464–4625 (CD/DVD)

ISSN 2464–4617 (print) ISSN 2464–4625 (CD/DVD)

CSRN 3101

(Eds.)

- **Vaclav Skala**
  University of West Bohemia, Czech Republic

Computer Science Research Notes

29. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision
WSCG 2021
Plzen, Czech Republic
May 17 – 20, 2021

Proceedings

WSCG 2021

Proceedings

Vaclav Skala - UNION Agency

ISSN 2464–4617 (print) ISSN 2464–4625 (CD/DVD)

Computer Science Research Notes
CSRN 3101

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia
Univerzitni 8
CZ 306 14 Plzen
Czech Republic
skala@kiv.zcu.cz http://www.VaclavSkala.eu

Managing Editor: Vaclav Skala

Publisher & Author Service Department & Distribution:
Vaclav Skala - UNION Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic
Reg.No. (ICO) 416 82 459

Published in cooperation with the University of West Bohemia
Univerzitní 8, 306 14 Pilsen, Czech Republic

ISSN 2464-4617 (Print) ***ISSN 2464-4625*** (CD/DVD)
e-ISBN 978-80-86943-34-3

WSCG 2021

International Program Committee

Baranoski,G, (Canada)
Barton,M, (Spain)
Benes,B, (United States)
Benger,W, (Austria)
Birra,F, (Portugal)
Bouatouch,K, (France)
Bourke,P, (Australia)
Bujack,R, (United States)
Coquillart,S, (France)
Dachsbacher,C, (Germany)
Debelov,V, (Russia)
Drakopoulos,V, (Greece)
Drechsler,K, (Germany)
Durikovic,R, (Slovakia)
Eisemann,M, (Germany)
Feito,F, (Spain)
Feng,J, (China)
Ferguson,S, (United Kingdom)
Galo,M, (Brazil)
Gavrilova,M, (Canada)
Gdawiec,K, (Poland)
Giannini,F, (Italy)
Gobbetti,E, (Italy)
Goebel,M, (Germany)
Groeller,E, (Austria)
Gudukbay,U, (Turkey)
Gunther,T, (Germany)
Hast,A, (Sweden)
Hast,A, (Sweden)
Hauenstein,J, (United States)
Hitzer,E, (Japan)
Chaudhuri,D, (India)
Chmielewski,L, (Poland)
Choi,S, (Korea)
Juan,C, (Spain)
Karim,A.,S, (Malaysia)
Klosowski,J, (United States)
Kumar,S, (India)
Kurt,M, (Turkey)
Lee,J, (United States)
Li,X, (United States)
Liu,S, (China)
Lobachev,O, (Germany)
Marco,C, (Brazil)
Max,N, (United States)
Molla,R, (Spain)
Montrucchio,B, (Italy)
Muller,H, (Germany)
Nasri,A, (United Arab Emirates)
Oliveira,J, (Portugal)
Pan,R, (China)
Paquette,E, (Canada)
Pedrini,H, (Brazil)
Platis,N, (Greece)
Radouane,K, (France)
Renaud,c, (France)
RESHETOV,A, (United States)
Richardson,J, (United States)
Rodrigues,J, (Portugal)
Rojas-Sola,J, (Spain)
Rushmeier,H, (United States)
Santos,L, (Portugal)
Savchenko,V, (Japan)
Segura,R, (Spain)
Scheuermann,G, (Germany)
Skala,V, (Czech Republic)
Szecsi,L, (Hungary)
Teschner,M, (Germany)
Thalmann,D, (Switzerland)
Tokuta,A, (United States)
Trapp,M, (Germany)
Wu,S, (Brazil)
Wuensche,B, (New Zealand)
Wuethrich,C, (Germany)
Zwettler,G, (Austria)

WSCG 2021

Board of Reviewers

Aguirre-Lopez,M, (Mexico)
Al-Ameen,Z, (Iraq)
Al-Darraji,S, (Iraq)
Baranoski,G, (Canada)
Barton,M, (Spain)
Benes,B, (United States)
Benger,W, (Austria)
Benziane,S, (Algeria)
Bouatouch,K, (France)
Bourke,P, (Australia)
Burova,I, (Russia)
Cabiddu,D, (Italy)
Cakmak,H, (Germany)
Cannavo,A, (Italy)
Carmo,M, (Portugal)
Dachsbacher,C, (Germany)
De Martino,J, (Brazil)
Debelov,V, (Russia)
Delibasoglu,I, (Turkey)
Drakopoulos,V, (Greece)
Drechsler,K, (Germany)
Durikovic,R, (Slovakia)
Dziembowski,A, (Poland)
Eid,A, (Egypt)
Eisemann,M, (Germany)
Etemad,K, (Canada)
Feito,F, (Spain)
Feld,S, (Netherlands)
Ferguson,S, (United Kingdom)
Fortunato Costa,B, (Brazil)
Fuenfzig,C, (Germany)
Gallucci,A, (Netherlands)
Galo,M, (Brazil)
Gangisetty,S, (India)
Gavrilova,M, (Canada)
Gdawiec,K, (Poland)
Gerhards,J, (France)
Giannini,F, (Italy)
Giraldo,J, (France)
Gobbetti,E, (Italy)
Goebel,M, (Germany)
Goncalves,A, (Portugal)
Grajek,T, (Poland)
Gudukbay,U, (Turkey)
Gunther,T, (Germany)
Ha,N, (Viet Nam)
Hast,A, (Sweden)
Hauenstein,J, (United States)
Hitzer,E, (Japan)
Horain,P, (France)
Hu,C, (Taiwan)
Hu,S, (China)
Hu,P, (Belgium)
Chaudhuri,D, (India)
Chmielewski,L, (Poland)
Jacek,K, (Poland)
Juan,C, (Spain)
Jurado,J, (Spain)
Justine,B, (France)
Karim,A.A.,S, (Malaysia)
Klosowski,J, (United States)
Komati,K, (Brazil)
Kumar,S, (India)
Kumar,S, (India)
Kurasova,O, (Lithuania)
Kurt,M, (Turkey)
Last,P, (Germany)
Lee,J, (United States)
Liu,S, (China)
Liu,B, (China)
Lobachev,O, (Germany)
Manzke,M, (Ireland)
Manzke,M, (Ireland)
Marco,C, (Brazil)
Marques,R, (Spain)
Mastmeyer,A, (Germany)
Matey,L, (Spain)
Max,N, (United States)
Meyer,A, (France)
Mieloch,D, (Poland)

MOHD SUAIB,N, (Malaysia)
Montrucchio,B, (Italy)
Muller,H, (Germany)
Muraleedharan,L, (India)
Nguyen,V, (Viet Nam)
Oliveira,J, (Portugal)
Pagnutti,G, (Italy)
Papagiannakis,G, ()
Papaioannou,G, (Greece)
Paquette,E, (Canada)
Parakkat,A, (Netherlands)
Pedrini,H, (Brazil)
Pereira,J, (Portugal)
Pérez-Díaz,S, (Spain)
Pintus,R, (Italy)
Platis,N, (Greece)
Radouane,K, (France)
Raffin,R, (France)
Ray,B, (India)
Reshetov,A, (United States)
Richardson,J, (United States)
Rodrigues,N, (Portugal)
Rodrigues,J, (Portugal)
Rojas-Sola,J, (Spain)
Rushmeier,H, (United States)
S P,R, (India)
Savchenko,V, (Japan)
Segura,R, (Spain)
Scheuermann,G, (Germany)
Sik-Lanyi,C, (Hungary)
Sirakov,N, (United States)
Skala,V, (Czech Republic)
Skopin,I, (Russia)
Szecsi,L, (Hungary)
Tas,F, (Turkey)
Tavares,J, (Portugal)
Thalmann,D, (Switzerland)
Todt,E, (Brazil)
Toler-Franklin,C, (United States)
Tourre,V, (France)
Trapp,M, (Germany)
Tuan,N, (United Kingdom)
Tytkowski,K, (Poland)
Wiegreffe,D, (Germany)
Wu,S, (Brazil)
Wuethrich,C, (Germany)
Yoshizawa,S, (Japan)
Zwettler,G, (Austria)

CSRN 3101

WSCG 2021 Proceedings

Contents

Reduction of JPEG Artifacts using BSDEs

Dariusz Borkowski
Faculty of Mathematics and Computer Science
Nicolaus Copernicus University
Chopina 12/18, 87-100
Toruń, Poland
dbor@mat.umk.pl

Katarzyna Jańczak-Borkowska
Institute of Mathematics and Physics University of Science and Technology
al. prof. S. Kaliskiego 7,
85-789 Bydgoszcz,
Poland
kaja@utp.edu.pl

ABSTRACT

In this paper we propose a novel approach for reduction of JPEG artifacts using advanced techniques of stochastic calculus. In order to solve this problem we use backward stochastic differential equations (in short BSDEs) and the non local means method. In our algorithm we consider two processes. One of them has values in the image domain and determines pixels that will be involved in the reconstruction, the second one has values in the image codomain and gives weight to values of pixels. To calculate the weights, we use the idea of the patches similarity used in the non local means. Our experiments show that the new approach gives very good results and compares favourably with other methods.

Keywords

Image reconstruction, JPEG artifacts, Stochastic differential equations, Non local means

1 INTRODUCTION

Lossy compression algorithms discard data that is least important to the recipient's visual sense, leaving the data of greater importance. The human eye is not sensitive to high frequency information, which is why most lossy image compression methods are implemented by quantization or approximation in the frequency domain. JPEG is a representative standard for lossy compression and is used worldwide. The JPEG compression standard splits the image, performs a discrete cosine transformation and then transformation coefficients are quantized and coded. The amount of data rejected is determined by the quality of compression.

There are many different techniques for the reconstruction of JPEG images. Early preliminary works [Ree84, Lis03, Lee04, Bre12, Wan13, Pou14, Gay15, Pan15] perform filtering to reduce only blocking artifacts. Many interesting works are done by low-pass filtering and frequency domain techniques [Cho98, Lee98, Lia02, Tri03, Abb06, Sin07, Sin11, Gol14]. Another popular method uses the concept of projection onto convex sets [Wee02, Gan03]. Several proposals have been made that make use of wavelet transforms [Hsu98, Cho00]. An important role in reducing of artifacts play the traditional noise reduction algorithms [Dab07, Foi07]. A different approach is based on multiple dictonary and machine learning methods [Cha13, Che15, Li16]. Dong et al. [Don14] first introduced a deep neural network to solve the problem of reduction of artifacts. The concept of deep neural networks has gained wide recognition and its later versions are successfully used to JPEG restoration [Don15, Svo16, Cav17, Zha17, Zha18, Ehr20, Kim20, Wan20].

In this paper we present a new method of artifacts reduction in the case of RGB images. We combine the idea of image denoising based on backward stochastic differential equations from [Bor17] and the concept of applying the non local means method to Feynman-Kac formula from the paper [Bor14].

The contribution of this work is twofold. We give a new noise reduction approach based on BSDE and non local means method. Moreover, we apply the obtained algorithm for JPEG reconstruction.

The rest of the paper is constructed as follows. In Section 2 we recall the method of denoising in terms of backward stochastic differential equations from [Bor17]. Section 3 provides a new method of restoration of the noisy image. Section 4 contains information about using an algorithm in order to remove JPEG

artifacts. Finally, in Section 5 experimental results and comparison to other methods are presented.

2 IMAGE RECONSTRUCTION BASED ON BSDES

2.1 Continuous model

Let D be a bounded, convex domain in $\mathbf{R}^2$, $u : \overline{D} \to \mathbf{R}^3$ be an original RGB image and $u_0 : \overline{D} \to \mathbf{R}^3$ be the observed image of the form $u_0 = u + \eta$, where η stands for a white Gaussian noise (independently added to all coordinates). The BSDE model to restoration of the image $u(x)$ is the following

$$\begin{cases} X_t = x + \int_0^t \sigma(s, X_s)\, dW_s + K_t^{\overline{D}}, \quad t \in [0,T], \\ Y_t = u_0(X_S) + \int_t^T c(s)(Y_s - u_0(X_s))ds - \\ \qquad\qquad \int_t^T Z_s\, dW_s, \quad t \in [0,T], \end{cases}$$

where $S < T$, $\{X_t\}_{t\in[0,T]}$ is a stochastic diffusion process, $\{W_t\}_{t\in[0,T]}$ is two-dimensional Wiener process, the term $\{K_t^{\overline{D}}\}_{t\in[0,T]}$ is the minimal push needed to keep process X in $\overline{D}$, $\{Y_t\}_{t\in[0,T]}$ is the first component of the solution to the BSDE, $\{Z_t\}_{t\in[0,T]}$ is the second component of the solution to the BSDE and determines the measurability of the process Y,

$$\sigma(s,x) = \left[\left(1 - \tfrac{c(s)}{c}\right) \theta_-(G_\gamma * u_0, x), \tfrac{c(s)}{c} \theta_+(G_\gamma * u_0, x) \right]$$

$$c(t) = \begin{cases} 0 & \text{if } t < S \text{ or } N(G_\gamma * u_0, x) < d, \\ c & \text{if } t \geq S \text{ and } N(G_\gamma * u_0, x) \geq d, \end{cases}$$

$\theta_+(u,x) \in \mathbf{R}^2$, $\theta_-(u,x) \in \mathbf{R}^2$ and $N(u,x) \in \mathbf{R}$ is Di Zenzo [Diz86, Der02] geometry of the RGB image $u((x_1,x_2)) = (R((x_1,x_2)), G((x_1,x_2)), B((x_1,x_2))$ at point x

$$\theta_\pm(u,x) = \frac{\nu_\pm(u,x)}{|\nu_\pm(u,x)|},$$

$$N(u,x) = \sqrt{\lambda(u,x)},$$

$$\lambda(u,x) = \frac{\sqrt{\chi(u,x)} + \frac{\partial R}{\partial x_1}(x)^2 + \frac{\partial G}{\partial x_1}(x)^2 + \frac{\partial B}{\partial x_1}(x)^2}{2} + \frac{\frac{\partial R}{\partial x_2}(x)^2 + \frac{\partial G}{\partial x_2}(x)^2 + \frac{\partial B}{\partial x_2}(x)^2}{2}$$

$$\nu_\pm(u,x) = \begin{bmatrix} 2\left(\frac{\partial R}{\partial x_1}(x)\frac{\partial R}{\partial x_2}(x) + \frac{\partial G}{\partial x_1}(x)\frac{\partial G}{\partial x_2}(x) + \frac{\partial B}{\partial x_1}(x)\frac{\partial B}{\partial x_2}(x)\right) \\ \frac{\partial R}{\partial x_2}(x)^2 + \frac{\partial G}{\partial x_2}(x)^2 + \frac{\partial B}{\partial x_2}(x)^2 - \frac{\partial R}{\partial x_1}(x)^2 - \frac{\partial G}{\partial x_1}(x)^2 - \frac{\partial B}{\partial x_1}(x)^2 \pm \sqrt{\chi(u,x)} \end{bmatrix}$$

$$\chi(u,x) = \frac{\partial R}{\partial x_1}(x)^2 + \frac{\partial G}{\partial x_1}(x)^2 + \frac{\partial B}{\partial x_1}(x)^2 - \frac{\partial R}{\partial x_2}(x)^2 - \frac{\partial G}{\partial x_2}(x)^2 - \frac{\partial B}{\partial x_2}(x)^2 + +4\left(\frac{\partial R}{\partial x_1}(x)\frac{\partial R}{\partial x_2}(x) + \frac{\partial G}{\partial x_1}(x)\frac{\partial G}{\partial x_2}(x) + \frac{\partial B}{\partial x_1}(x)\frac{\partial B}{\partial x_2}(x)\right)^2$$

G_γ is a 3×3 Gaussian kernel and $S \in \mathbf{R}_+ < T \in \mathbf{R}_+$, $d \in \mathbf{R}_+$, $c \in \mathbf{R}_+$ are parameters of the method.

For a fixed pixel x we consider a certain BSDE equation. The values of the process X determines pixels from domain of the image $\overline{D}$ which we will use in process reconstruction. We can say that this process determines neighbourhood of the pixel x (with irregular shape). The reconstructed value $u(x)$ is the sum of pixels from its neighbourhood multiplied by some weights. The weight values are determined by the process Y. Appropriate definition of the function $c(t)$ allows as to give weight values (also negative) which depend on direction and distance from reconstructed pixel.

Parameter T defines the size of the neighbourhood used in the reconstruction procedure. We deblur from time T to S and smooth out from S to 0. The parameter d determines which pixels will be reconstructed with using smoothing model and which with using enhancing model. The parameter c is responsible for effect of edge sharpening. The values of all parameters depend on standard noise deviation ρ.

2.2 Algorithm

Consider a time discretization $0 = t_0 < t_1 < \ldots < t_j \leq S < t_{j+1} < \ldots < t_m = T, t_i - t_{i-1} = \frac{T}{m}$. In the first step we generate a trajectory of the stochastic process X for $k = 0, 1, \ldots, m-1$ using the Euler formula [Slo01]

$$\begin{aligned} &X_0 = x, \\ &X_{t_k} = \Pi_{\overline{D}}[X_{t_{k-1}} + \sigma(t_{k-1}, X_{t_{k-1}})(W_{t_k} - W_{t_{k-1}})], \end{aligned} \tag{1}$$

where $\Pi_{\overline{D}}(x)$ denotes a projection of x on the set $\overline{D}$ and W is a Wiener process. The difference $W_{t_k} - W_{t_{k-1}}$ we approximate using random number generator. Since W_t is 2 – dimensional, the value of the difference $W_{t_k} - W_{t_{k-1}}$ is equal to two independent values (vector) obtained with a generator of the normal distribution $\mathcal{N}(0, t_k - t_{k-1})$.

Example of the sequence X given by formula (1) is shown on Figure 1 (a). The image has two areas: gray and white. The process X starts from the reconstructed pixel x located on the edge. Next, the process X has values along the edge until time t_j, then after time t_j moves towards the vector θ_+ (gradient vector).

Now, we can define the process Y. From a definition of BSDE [Bor17] it starts from

$$Y_{t_m} = u_0(X_{t_j})$$

and then we backwardly count values $Y_{t_{m-1}}, Y_{t_{m-2}}, \ldots, Y_0$

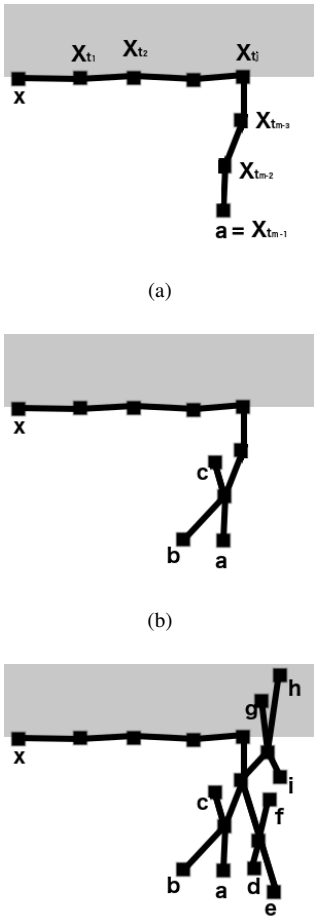


Figure 1: Trajectory of the process X.

$$Y_{t_k} = \mathbf{E}\left[Y_{t_{k+1}}|\mathscr{F}_{t_k}\right] + \frac{T}{m}c(t_k)\left(\mathbf{E}\left[Y_{t_{k+1}}|\mathscr{F}_{t_k}\right] - u_0(X_{t_k})\right),$$

$k = m-1, m-2, \ldots, 0$, where by $\mathbf{E}$ we denote the expected value and by $\mathscr{F}_t$ the filtration generated by the discretization of the Wiener process [Ma02].

For $k = m-1$ we have

$$Y_{t_{m-1}} = \mathbf{E}\left[Y_{t_m}|\mathscr{F}_{t_{m-1}}\right] + \frac{T}{m}c(t_{m-1})\left(\mathbf{E}\left[Y_{t_m}|\mathscr{F}_{t_{m-1}}\right] - u_0(X_{t_{m-1}})\right).$$

Note that Y_{t_m} is $\mathscr{F}_{t_{m-1}}$ measurable. Therefore

$$Y_{t_{m-1}} = Y^a_{t_{m-1}} = u_0(X_{t_j}) + \frac{T}{m}c(t_{m-1})\left(u_0(X_{t_j}) - u_0(a)\right).$$

Next, for $k = m-2$

$$Y_{t_{m-2}} = \mathbf{E}\left[Y_{t_{m-1}}|\mathscr{F}_{t_{m-2}}\right] + \frac{T}{m}c(t_{m-2})\left(\mathbf{E}\left[Y_{t_{m-1}}|\mathscr{F}_{t_{m-2}}\right] - u_0(X_{t_{m-2}})\right).$$

Since $Y_{t_{m-1}}$ is not $\mathscr{F}_{t_{m-2}}$ measurable, we need to count $\mathbf{E}\left[Y_{t_{m-1}}|\mathscr{F}_{t_{m-2}}\right]$ by using Monte Carlo method with M iterations. We start M-times from point $X_{t_{m-2}}$. Example for $M = 3$ is shown on Figure 1 (b) (in practise we need to use about 10 iterations). As before we count $Y^b_{t_{m-1}}$ and $Y^c_{t_{m-1}}$ and then

$$\mathbf{E}\left[Y_{t_{m-1}}|\mathscr{F}_{t_{m-2}}\right] \approx \frac{Y^a_{t_{m-1}} + Y^b_{t_{m-1}} + Y^c_{t_{m-1}}}{3},$$

$$Y_{t_{m-2}} \approx Y^{a,b,c}_{t_{m-2}} = \frac{Y^a_{t_{m-1}} + Y^b_{t_{m-1}} + Y^c_{t_{m-1}}}{3} + \frac{T}{m}c(t_{m-2})\left(\frac{Y^a_{t_{m-1}} + Y^b_{t_{m-1}} + Y^c_{t_{m-1}}}{3} - u_0(X_{t_{m-2}})\right).$$

Next, value for t_{m-3} is equal to

$$Y_{t_{m-3}} = \mathbf{E}\left[Y_{t_{m-2}}|\mathscr{F}_{t_{m-3}}\right] + \frac{T}{m}c(t_{m-3})\left(\mathbf{E}\left[Y_{t_{m-2}}|\mathscr{F}_{t_{m-3}}\right] - u_0(X_{t_{m-3}})\right)$$

and again, similarly to $Y_{t_{m-2}}$ we need to count it by using Monte Carlo method. For $M = 3$ (see Figure 1 (c)) we have the formula

$$Y_{t_{m-3}} \approx Y^{a,b,c,d,e,f,g,h,i}_{t_{m-3}} = \frac{Y^{a,b,c}_{t_{m-2}} + Y^{d,e,f}_{t_{m-2}} + Y^{g,h,i}_{t_{m-2}}}{3} + \frac{T}{m}c(t_{m-3})\left(\frac{Y^{a,b,c}_{t_{m-2}} + Y^{d,e,f}_{t_{m-2}} + Y^{g,h,i}_{t_{m-2}}}{3} - u_0(X_{t_{m-3}})\right).$$

The above reasoning is repeated until we determine Y_0 which is a reconstructed value i.e. $u(x)$.

3 MODIFICATION BASED ON FEYNMAN-KAC FORMULA AND NON LOCAL MEANS

Note, that for times $0 < t_0 < t_1 < \ldots < t_j$ and from definition of the function $c(t)$ ($c(t) = 0$, for $t < S$) the algorithm described in the previous section works as follows

$$Y_{t_{j-1}} = \mathbf{E}\left[Y_{t_j}|\mathscr{F}_{t_{j-1}}\right] \approx \frac{1}{M}\sum_{i=1}^{M} Y^{\omega_i}_{t_j} = \sum_{i=1}^{M}\frac{1}{M}Y^{\omega_i}_{t_j},$$

$$Y_{t_{j-2}} \approx \sum_{i=1}^{M}\frac{1}{M}Y^{\omega_i}_{t_{j-1}},$$

$$\vdots$$

$$Y_{t_0} \approx \sum_{i=1}^{M}\frac{1}{M}Y^{\omega_i}_{t_1}$$

The above approximation is based on Feynman-Kac formula, which means that each value of pixel $Y^{\omega_i}_{t_k}$,

$k = 1,2,\ldots,j$ is weighted with the same value $\frac{1}{M}$. But since pixels have different colours we may consider them with different weights depending on their neighbourhood. We follow the non local means algorithm [Bua05] and propose to think of weights that depend on patches similarity i.e.

$$Y_{t_{j-1}} \approx \sum_{i=1}^{M} w(B_{x,r}, B_{X_{t_j},r}) Y_{t_j}^{\omega_i},$$

$$Y_{t_{j-2}} \approx \sum_{i=1}^{M} w(B_{x,r}, B_{X_{t_{j-1}},r}) Y_{t_{j-1}}^{\omega_i},$$

$$\vdots$$

$$Y_{t_0} \approx \sum_{i=1}^{M} w(B_{x,r}, B_{X_{t_1},r}) Y_{t_1}^{\omega_i},$$

where by $B_{x,r}$ we denote $(2r+1)\times(2r+1)$ RGB pixels centered at point x and by $w(\cdot,\cdot)$ we denote a weight function defined in [Bua11].

For times $S < t_{j+1} < \ldots < t_m = T$ the function $c(t)$ is greater than zero which means that we have negative weights (to obtain the enhancement effect) and the algorithm remains unchanged.

4 REDUCTION OF JPEG ARTIFACTS

Note, that if we apply values of parameters as default values recommended by the authors of papers [Bua11, Bor14, Bor17] then algorithm still has one parameter: ρ. In this case algorithm can be used successfully to reconstruction of Gaussian noisy images with given standard deviation of the noise.

In the case of reduction of JPEG artifacts we have to combine the ρ parameter with the compression quality q. In JPEG standards the compression quality q is always known and is expressed as a percentage. An image at 100% quality has no loss. We propose the following formula to count the ρ parameter:

$$\rho = \max\{-0.3q + 20, 0\}, \tag{2}$$

where q is a JPEG compression quality of the image. Choosing this function, we followed the principle of maximizing the Peak Signal to Noise Ratio (in short PSNR) :

$$\mathrm{PSNR}(u,\hat{u}) = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}(u,\hat{u})}\right),$$

where u, $\hat{u}$ denote the original and the restored RGB image and $\mathrm{MSE}(u,\hat{u})$ is a mean square error between u and $\hat{u}$. To determine formula (2) we used several standard test images. For each, fixed $q \in \{5,10,20,30,40,50\}$ and for each test image we calculated ρ at which we have the maximum PSNR. These values were averaged – for each q we obtained a mean value of ρ. Finally, for these data we used the linear regression model to count the linear function $-0.3q+20$.

Figure 2: Test RGB images: Toysflash 912×684, Lighthouse 480×640, Gantry 400×264, Onion 198×135.

5 EXPERIMENTAL RESULTS

In this section we present experimental results illustrating the difference between our algorithm and other methods of removing artifacts: SADCT [Foi07], CBM3D [Dab07], ARCNN [Don15], DNCNN [Zha17] and BSDE [Bor17] as the parent method for our approach. We use the MATLAB implementation of compared methods: SADCT [Foi20b], CBM3D [Foi20a], ARCNN [Yuk20], DNCNN [Jpe20]. Parameters of these approaches were set to the default values as recommended by the authors. Some results for our evaluation experiments are presented in Fig. 3, Fig. 4, Fig. 5, Fig. 6, Fig. 7. The results refer to RGB colour images *Toysflash*, *Lighthouse*, *Gantry*, *Onion* corrupted with the JPEG compression algorithm with different values of quality. The maximum values of Peak Signal to Noise Ratio and Structural SIMilarity (in short SSIM) index [Wan04] obtained using tested methods are given in tables: Table 2, Table 3. The reconstruction time of our method on Intel Core i7 is shown in the Table 1.

Table 1: Time of the reconstruction (in seconds) of proposed method. It has been tested for 2 × CPU 1,7 GHz.

Image\ JPEG Quality	10	20	30	40
Lighthouse 480×640	4.07	4.53	4.45	5.86
Toysflash 912×684	7.65	8.02	8.46	9.50
Onion 198×135	1.30	1.40	1.33	1.51
Gantry 400×264	2.23	2.40	2.32	2.50

The analysis of the measures of image quality from the Table 2 shows that the new method performs better, especially in the case of low value of JPEG quality. In particular, our modification of the algorithm by adding weights improves results of the parent method [Bor17]. Interesting examples are Fig. 3 and Fig. 4 in which

one should look at artifacts marked with black arrows. In the Fig. 3 we can see ringing artifacts. These artifacts are mainly due to the coarse quantization of the high-frequency DCT coefficients, making the decompressed image to exhibit noisy patterns known as ringing or mosquito noise near the edge. In the Fig. 4 we can see blocking artifacts, which are mainly due to the coarse quantization of low-frequency DCT coefficients yielding decompressed image look like a mosaic at smooth regions [Ozt07]. Our method removed artifacts very well in both cases.

It can be noticed from tables that the proposed method and SADCT give similar results. However looking at the pictures Fig. 5, Fig. 6, Fig. 7 we see the visual difference between these two methods. After using SADCT, artifacts at the edges are still visible but after using the proposed method these artifacts are smoothed out.

6 CONCLUSION

In this paper we present a new method for reducing of JPEG artifacts based on backward stochastic differential equations. The main purpose of this work is to present a new mathematical tool to restore digital images that are qualitatively indistinguishable from other approaches. It seems that further exploration of this tool will also improve the time complexity of this stochastic algorithm.

7 REFERENCES

[Abb06] Abboud, I. Deblocking in BDCT image and video coding using a simple and effective method. Information Technology Journal, No. 5(3), pp. 422-426, 2006.

[Bor14] Borkowski, D., Jakubowski, A., and Jańczak-Borkowska, K. Feynman-Kac formula and restoration of high iso images. International Conference on Computer Vision and Graphics, pp. 100-107, 2014.

[Bor17] Borkowski, D. and Jańczak-Borkowska, K. Image denoising using backward stochastic differential equations. International Conference on Man-Machine Interactions, pp. 185-194, 2017.

[Bre12] Bredies, K. and Holler, M. A total variation–based jpeg decompression model. SIAM Journal on Imaging Sciences, No. 5(1), pp. 366-393, 2012.

[Bua05] Buades, A., Coll, B., and Morel, J. M. A non local algorithm for image denoising. IEEE Computer Vision and Pattern Recognition, No. 2, pp. 60-65, 2005.

[Bua11] Buades, A., Coll, B., and Morel, J. M. (2011). Non-local means denoising. Image Processing On Line, No. 1, pp. 208-212, 2011.

[Cav17] Cavigelli, L., Hager, P., and Benini, L. Cascnn: A deep convolutional neural network for image compression artifact suppression. International Joint Conference on Neural Networks, pp. 752-759, 2017.

[Cha13] Chang, H., Ng, M. K., and Zeng, T. Reducing artifacts in jpeg decompression via a learned dictionary. IEEE Transactions on Signal Processing, No. 62(3), pp. 718-728, 2013.

[Che15] Chen, Y., Yu, W., and Pock, T. On learning optimized reaction diffusion processes for effective image restoration. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5261-5269, 2015.

[Cho00] Choi, H. and Kim, T. Blocking-artifact reduction in block-coded images using wavelet-based subband decomposition. IEEE Transactions on Circuits and Systems for Video Technology, No. 10(5), pp. 801-805, 2000.

[Cho98] Chou,J., Crouse, M., and Ramchandran, K. A simple algorithm for removing blocking artifacts in block-transform coded images. IEEE Signal Processing Letters, No. 5(2), pp. 33-35, 1998.

[Dab07] Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. Image denoising by sparse 3-d transform-domain collaborative filtering. IEEE Transactions on Image Processing, No. 16(8), pp. 2080-2095, 2007.

[Der02] Deriche, R. and Tschumperlé, D. Diffusion pde's on vector-valued images: local approach and geometric viewpoint. IEEE Signal Processing Magazine, No. 19(5), pp. 16-25, 2002.

[Diz86] Di Zenzo, S. A note on the gradient of a multi-image. Computer Vision, Graphics, and Image Processing, No. 33(1), pp. 116-125, 1986.

[Don15] Dong, C., Deng, Y., Change Loy, C., and Tang, X. Compression artifacts reduction by a deep convolutional network. Proceedings of the IEEE International Conference on Computer Vision, pp. 576-584, 2015.

[Don14] Dong, C., Loy, C. C., He, K., and Tang, X. Learning a deep convolutional network for image super-resolution. European Conference on Computer Vision, pp. 184-199, 2014.

[Ehr20] Ehrlich M., Davis L., Lim SN., and Shrivastava A. Quantization guided JPEG artifact correction. Lecture Notes in Computer Science, No. 12353, 2020.

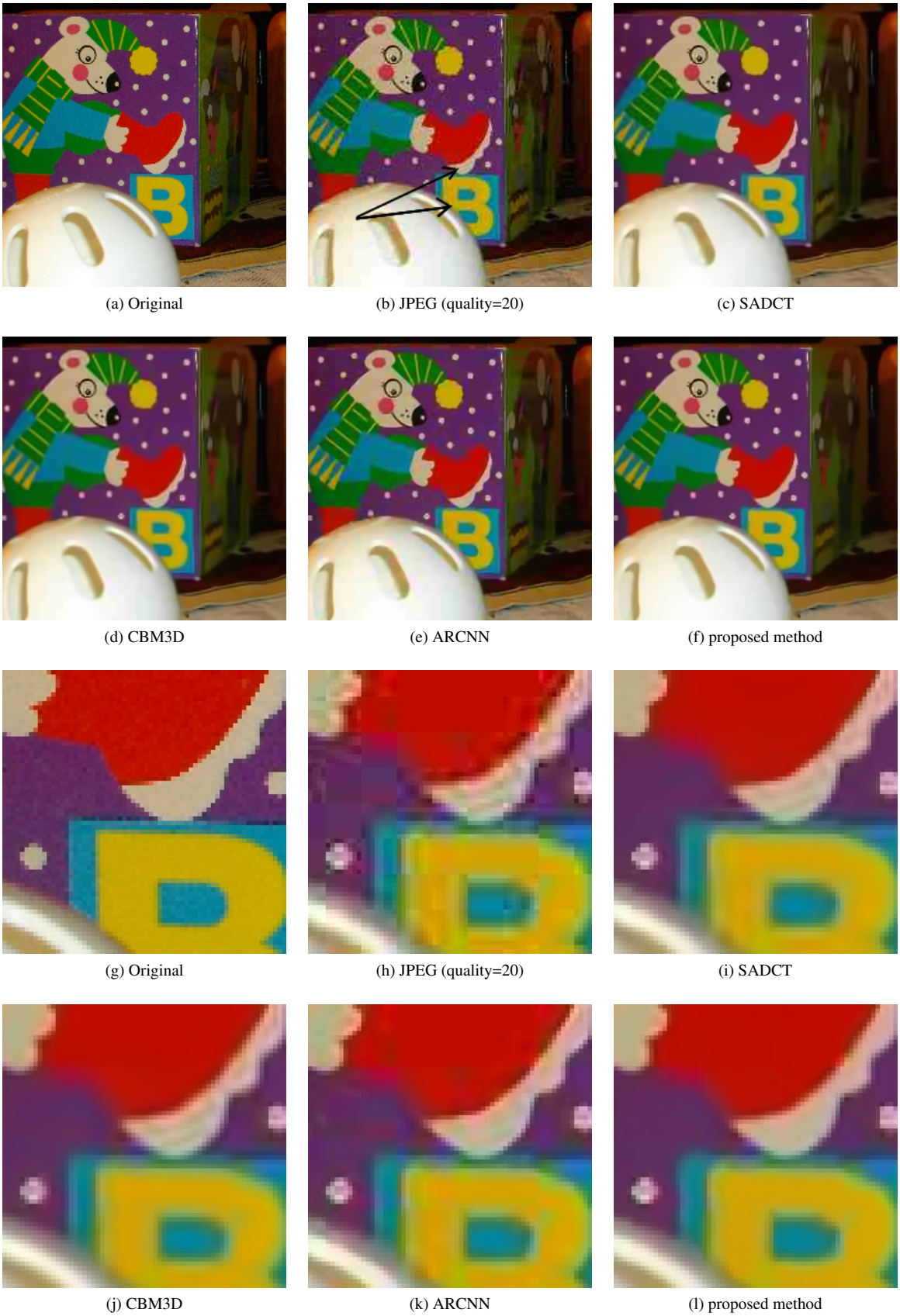
[Foi07] Foi, A., Katkovnik, V., and Egiazarian, K. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. IEEE transactions on Image Processing, No. 16(5), pp. 1395-1411, 2007.

Table 2: PSNR

Test Image	JPEG Quality	JPEG	SADCT [Foi07]	CBM3D [Dab07]	ARCNN [Don15]	DNCNN [Zha17]	BSDE [Bor17]	proposed method
	10	27.63	28.50	28.61	28.14	27.89	28.18	**28.68**
Toysflash	20	29.88	30.48	30.68	30.25	30.18	30.17	**30.69**
912×684	30	30.99	31.45	**31.69**	31.39	31.28	31.16	31.67
	40	31.70	32.07	**32.32**	32.03	31.98	31.84	32.30
	10	26.60	27.67	27.51	27.38	27.08	27.13	**27.69**
Lighthouse	20	28.80	29.54	29.56	29.45	29.34	28.91	**29.64**
480×640	30	30.02	30.54	30.69	30.71	30.54	30.04	**30.72**
	40	30.84	31.24	31.43	31.36	31.29	30.87	**31.45**
	10	26.17	27.45	27.37	26.69	26.47	27.03	**27.48**
Onion	20	28.25	29.10	29.00	28.70	28.57	28.56	**29.17**
198×135	30	29.46	30.30	30.23	29.89	29.80	29.70	**30.36**
	40	30.10	**30.87**	30.76	30.48	30.41	30.28	**30.87**
	10	23.15	24.06	24.01	23.80	23.53	23.56	**24.10**
Gantry	20	25.05	25.87	25.85	25.67	25.48	25.47	**25.92**
400×264	30	25.99	**26.77**	26.71	26.59	26.35	26.38	26.61
	40	26.68	**27.51**	27.42	27.16	27.01	27.08	27.13

Table 3: SSIM

Test Image	JPEG Quality	JPEG	SADCT [Foi07]	CBM3D [Dab07]	ARCNN [Don15]	DNCNN [Zha17]	BSDE [Bor17]	proposed method
	10	0.9100	0.9273	0.9258	0.9163	0.9139	0.9186	**0.9288**
Toysflash	20	0.9443	0.9529	0.9537	0.9431	0.9466	0.9485	**0.9539**
912×684	30	0.9556	0.9614	**0.9625**	0.9557	0.9571	0.9580	0.9620
	40	0.9620	0.9663	**0.9674**	0.9615	0.9633	0.9640	0.9672
	10	0.9062	**0.9232**	0.9180	0.9167	0.9129	0.9108	0.9198
Lighthouse	20	0.9490	**0.9621**	0.9586	0.9550	0.9544	0.9506	0.9600
480×640	30	0.9617	**0.9693**	0.9687	0.9670	0.9660	0.9622	0.9689
	40	0.9685	**0.9748**	0.9742	0.9709	0.9711	0.9692	0.9747
	10	0.9419	0.9600	0.9580	0.9486	0.9458	0.9551	**0.9601**
Onion	20	0.9606	0.9711	0.9701	0.9650	0.9636	0.9670	**0.9712**
198×135	30	0.9699	**0.9774**	0.9765	0.9730	0.9723	0.9736	0.9768
	40	0.9736	**0.9801**	0.9788	0.9763	0.9757	0.9765	0.9790
	10	0.8367	0.8665	0.8626	0.8526	0.8460	0.8502	**0.8665**
Gantry	20	0.8947	0.9158	0.9161	0.9074	0.9035	0.9075	**0.9178**
400×264	30	0.9159	0.9333	0.9334	0.9266	0.9221	0.9269	**0.9335**
	40	0.9244	**0.9412**	0.9409	0.9334	0.9308	0.9373	0.9399



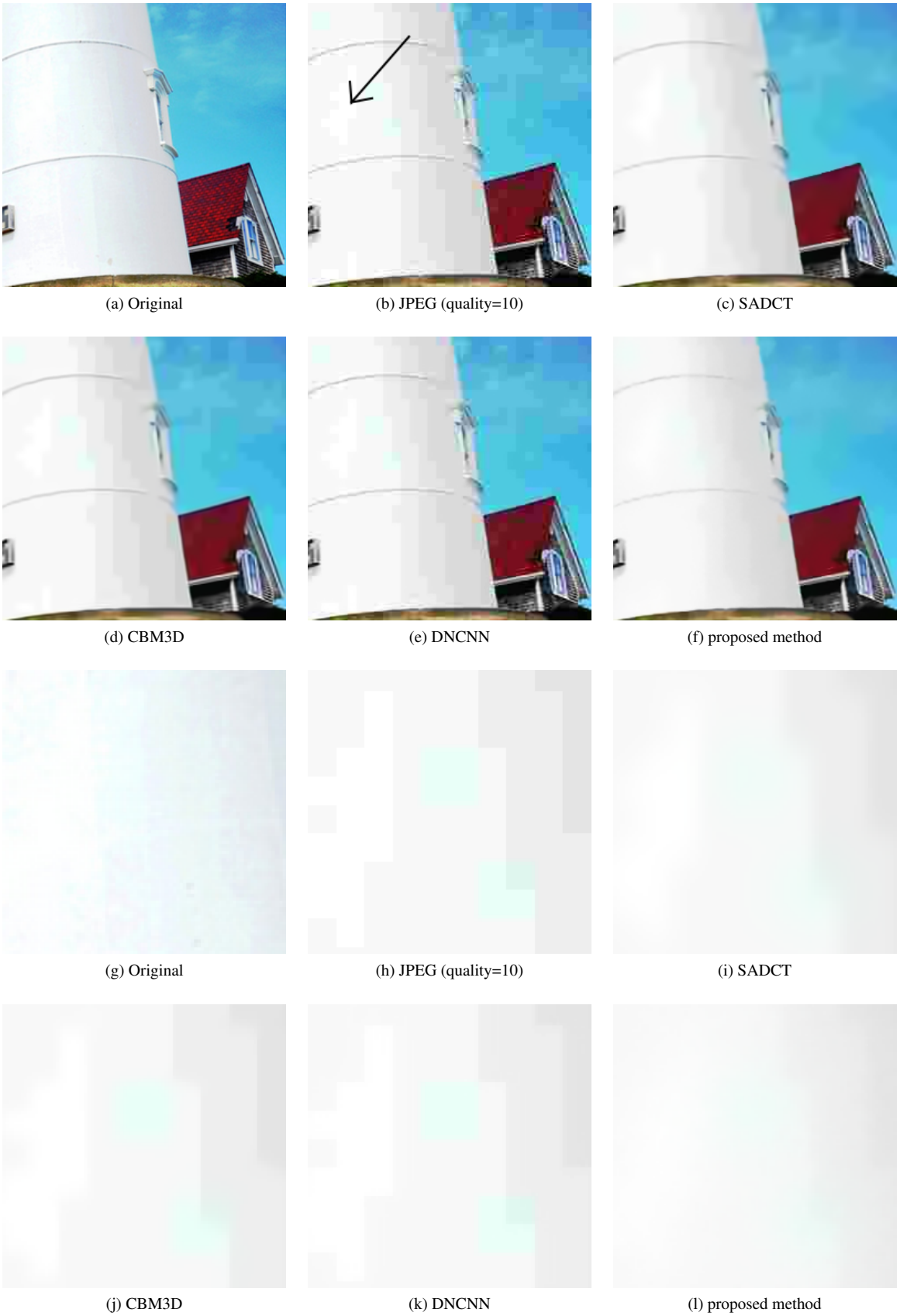
(a) Original (b) JPEG (quality=20) (c) SADCT

(d) CBM3D (e) ARCNN (f) proposed method

(g) Original (h) JPEG (quality=20) (i) SADCT

(j) CBM3D (k) ARCNN (l) proposed method

Figure 3: Restoration of the Toysflash image.



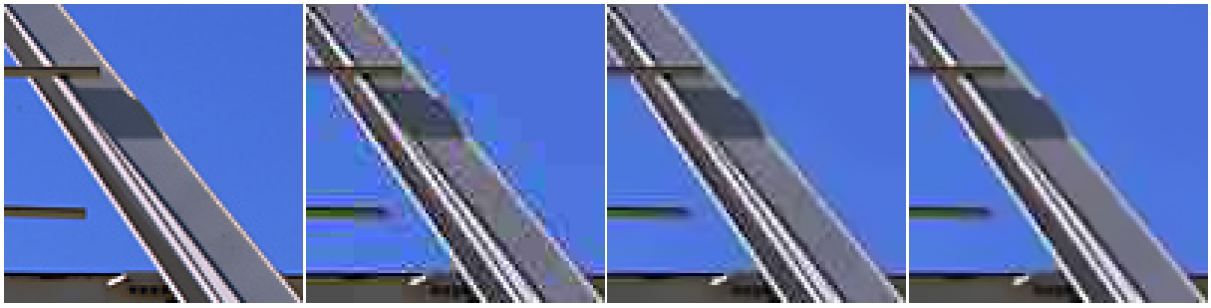
(a) Original (b) JPEG (quality=10) (c) SADCT

(d) CBM3D (e) DNCNN (f) proposed method

(g) Original (h) JPEG (quality=10) (i) SADCT

(j) CBM3D (k) DNCNN (l) proposed method

Figure 4: Restoration of the Lighthouse image.



(a) Original (b) JPEG (quality=20) (c) SADCT (d) proposed method

Figure 5: Restoration of the Gantry image for quality=20.

(a) JPEG (quality=30) (b) SADCT (c) proposed method

Figure 6: Restoration of the Gantry image for quality=30.

(a) JPEG (quality=40) (b) SADCT (c) proposed method

Figure 7: Restoration of the Gantry image for quality=40.

[Foi20a] Foi, A. Image and video denoising by sparse 3D transform-domain collaborative filtering. http://www.cs.tut.fi/~foi/GCF-BM3D/, accessed 2020.

[Foi20b] Foi, A. Shape-adaptive transforms filtering. https://www.cs.tut.fi/~foi/SA-DCT/, accessed 2020.

[Gan03] Gan, X., Wee-Chung Liew, A., and Yan, H. Blocking artifact reduction in compressed images based on edge-adaptive quadrangle meshes. Journal of Visual Communication and Image Representation, No. 14, pp. 492-507, 2003.

[Gay15] Gayathri Tejaswini, S., Ramalakshmi, M., Santhi, M., Rahul H., and Nag, H. Reduction of blocking artifacts of DCT compressed image based on block Wiener filtering. International Journal of Advanced Research in Electronics and Communication Engineering, No. 4(3), pp. 662-665, 2015.

[Gol14] Golestaneh, S. A., and Chandler, D. M. Algorithm for JPEG artifact reduction via local edge regeneration. Journal of Electronic Imaging, No. 23(1), 2014.

[Hsu98] Hsung, T, C., and Lun, D. P. K. Application of singularity detection for the deblocking of JPEG decoded images. IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, No. 45(5), pp. 640-644, 1998.

[Jpe20] JPEG image deblocking using deep learning. https://www.mathworks.com/help/images/jpeg-image-deblocking-using-deep-learning.html, accessed 2020.

[Kim20] Kim, Y., Soh, J. and Cho, N. AGARNet: Adaptively gated JPEG compression artifacts removal network for a wide range quality factor. IEEE Access, No. 8, pp. 20160–20170, 2020.

[Lee98] Lee, Y.L., Kim, H.C., and Park, H.W. Blocking effect reduction of JPEG images by signal adaptive filtering. IEEE Transactions on Image Processing, No. 7(2), pp. 229-234, 1998.

[Lee04] Lee, K., Kim, D. S., and Kim, T. Regression-based prediction for blocking artifact reduction in jpeg-compressed images. IEEE Transactions on Image Processing, No. 14(1), pp. 36-48, 2004.

[Li16] Li, S., Wang, G., Zhao, X. Multiplicative noise removal via adaptive learned dictionaries and TV regularization, Digital Signal Processing, No. 50, pp. 218-228, 2016.

[Lia02] Liaw, Y. C., Lo, W., and Lai, J. Z. C. Image restoration of compressed images using classified vector quantization. Pattern Recognition, No. 35, pp. 329-340, 2002.

[Lis03] List, P., Joch, A., Lainema, J., Bjontegaard, G., and Karczewicz, M. Adaptive deblocking filter. IEEE transactions on circuits and systems for video technology. No. 13(7), pp. 614-619, 2003.

[Ma02] Ma, J., Protter, P., San Martin, J., Torres, S., et al. Numerical method for backward stochastic differential equations. The Annals of Applied Probability, No. 12(1), pp. 302-316, 2002.

[Ozt07] Oztan, B., Malik, A., Fan, Z., and Eschbach, R. Removal of artifacts from JPEG compressed document images. Color Imaging XII: Processing, Hardcopy, and Applications No. 6493, 2007.

[Pan15] Pandey, S. S., Manu, Singh, P., and Pandey, V. Block wise image compression & reduced blocks artifacts using discrete cosine transform. International Journal of Scientific and Research Publications, No. 5(3), 2015.

[Pou14] Pourreza-Shahri, R., Yousefi, S. and Kehtarnavaz, N. Optimization method to reduce blocking artifacts in JPEG images. Journal of Electronic Imaging, No. 23(6), 2014.

[Ree84] Reeve, H. C. and Lim, J. S. Reduction of blocking effects in image coding. Optical Engineering, No. 23(1), 230134, 1984.

[Sin07] Singh, S., Kumar, V., and Verma, H. K. Reduction of blocking artifacts in JPEG compressed images. Digital Signal Processing, No. 17, pp. 225-243, 2007.

[Sin11] Singh, J., Singh, S., Singh, D., and Uddin, M. A signal adaptive filter for blocking effect reduction of JPEG compressed images. International Journal of Electronic and Communications, No. 65, pp. 827-839, 2011.

[Slo01] Słomiński, L. Euler's approximations of solutions of sdes with reflecting boundary. Stochastic Processes and their Applications, No. 94(2), pp. 317-337, 2001.

[Svo16] Svoboda, P., Hradiš, M., Bařina D., and Zemčík, P. Compression artifacts removal using convolutional neural networks. Journal of WSCG, No. 24(2), pp. 63–72, 2016.

[Tri03] Triantafyllidis, G. A., Varnuska, M., Sampson, D., Tzovaras, D., and Strintzis, M. G. An effiecient algorithm for the enhancement of JPEG-coded images. Computers & Graphics, No. 27, pp. 529-534, 2003.

[Wan04] Wang, Z., Simoncelli, E.P., and Bovik, A.C. Multiscale Structural similarity for image quality assessment. The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, pp. 1398–1402, 2003.

[Wan13] Wang, C., Zhou, J., and Liu, S. Adaptive non-local means filter for image deblocking. Signal Processing: Image Communication, No. 28(5), pp. 522–530, 2013.

[Wan20] Wang, M., Fu, X., Sun, Z., and Zha, Z. JPEG artifacts removal via compression quality ranker-guided networks. International Joint Conferences on Artificial Intelligence Organization, pp. 566–572, 2020.

[Wee02] Weerasinghe, C., Wee-Chung Liew, A., and Yan, H. Artifact reduction in compressed images based on region homogeneity constraints using the projection onto convex sets algorithm. IEEE Transactions on Circuits and Systems for Video Technology, No. 12(10), pp. 891-897, 2002.

[Yuk20] Yu, K., Dong, C., Deng, Y., Loy, C. C. and Tang, X. Deep convolution networks for compression artifacts reduction. http://mmlab.ie.cuhk.edu.hk/projects/ARCNN.html, accessed 2020.

[Zha17] Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing, No. 26(7), pp. 3142-3155, 2017.

[Zha18] Zhang, X., Yang, W., Hu, Y., and Liu, J. Dmcnn: Dual-domain multi-scale convolutional neural network for compression artifacts removal. 25th IEEE International Conference on Image Processing, pp. 390-394, 2018.

Line Clustering and Contour Extraction in the Context of 2D Building Plans

Andreas Pointner[1], Christoph Praschl[1], Oliver Krauss[1],
Andreas Schuler[1,2], Emmanuel Helm[1,2] and Gerald Zwettler[1,3]
firstname.lastname@fh-hagenberg.at

[1]Research Group Advanced Information Systems and Technology, Research and Development Department, University of Applied Sciences Upper Austria
Softwarepark 11
4232, Hagenberg im Mühlkreis, Austria

[2]Department of Medical and Bioinformatics, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria
Softwarepark 11
4232, Hagenberg im Mühlkreis, Austria

[3]Department of Software Engineering, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria
Softwarepark 11
4232, Hagenberg im Mühlkreis, Austria

ABSTRACT

For the purpose of analyzing a building according to its accessibility or structural resilience, printed 2D floor plans are not sufficient because of the missing link to semantic information. This paper tackles this issue and introduces a concept for clustering classified lines of a floor plan and for creating semantically enriched contour elements based on different image processing, computer vision and machine learning algorithms. Based on a general line clustering approach, we introduce type specific methods for *walls*, *windows*, *doors* and *stairs*. The resulting clusters are in turn used for a contour creation, which uses minimal rotated rectangles. Those rectangles are transformed to polygons that are refined using post processing steps. The approach is evaluated via positive testing using a pixel-based comparison of the process's result. For this, automatically generated as well as real world building plans are used. The final evaluation shows, that the concept reaches a confidence of >90% for door, stair and windows and only around 10% for stairs with the run-time linearly scaling with the size of the input.

Keywords

Clustering, Contour Extraction, Building Plan, Image Processing, Machine Learning

1 INTRODUCTION

In architecture, floor plans are a well-established concept to get a basic understanding about a building structure. This allows to quickly get an overview of size ratios, rooms and the general layout of a building. Despite this, there are a few scenarios where a 2D floor plan is not sufficient. Consequently, it is necessary to transform the plan into a 3D representation to be able to e.g. place furniture in a room or to take a virtual tour through the entire building using a head-mounted display. This paper does not address such digital, semantically rich floor plans, but printed versions, which no longer include that much meta-information.

We have developed a transformation process that uses an image as input, extracts all lines contained in it and classifies those into the types *door*, *wall*, *window* or *stair* using a neural network. These classified lines get clustered as contour elements, that are in turn used to create a 3D model. The final model is used for additional analysis e.g escape routes, or the building's accessibility. This paper focuses on one part in this pipeline; the transformation process that consists of the classified line clustering and the contour creation. This is targeted by our research question: How can classified lines be transformed to contour elements using clustering approaches? The research question leads to the paper's contributions:

- Introducing a concept for clustering classified building plan lines and
- Creating contour elements of the clusters represented as polygons

The remainder of this work outlines an overview of related work on automatic transformation of 2D building plans into 3D models in section 2. In section 3 the background of this work in terms of the used algorithms is presented. Section 4 describes our novel approach of combining different algorithms from the field of image processing, computer vision and machine learning for the transformation of a set of classified lines to contour elements. This transformation process is evaluated in section 5 based on a case study, that utilizes an automatic building plan generation process for creating test data. The results of our experiment are presented in section 6 and are finally concluded in section 7 with some final remarks and possible future work.

2 RELATED WORK

Liu et al. [Liu2017] present a concept on how to transform a rasterized floor plan into a vector-graphics representation which can further be converted to a popup 3D building plan. Consequently, they use predefined shape models, to detect different corner points. In contrast to their work, we do not have any default plan structure, and provide a generalized concept, that works on every building plan within the required visual appearance of the elements.

Zhiliang et al. [Zeng2019] show a deep-learning based system to recognize floor plans. They use a multitask neuronal network with two stages. In the first one, they predict room-boundary elements like doors, windows and walls and in the second task they predict the type of the room. Their work differs from our work w.r.t. the input data, utilizing classified lines and furthermore focusing on the clustering of those. In contrast, their approach combines both steps into one, but on building plans with simple wall types, where each wall is represented by a single line, instead of blocks with multiple filling patterns inside.

Lewis and Séquin [Lewis1998] describe a system for automatically converting 2D floor plans into 3D representations. The presented Building Model Generator (BMG) system takes common CAD formats and transforms them into an internal model, which can be used for adaptions of the plan and analysis. This system is required since the 3D generation of professional CAD tools does not fix faults in the plan. For this reason, the transformation process of BMG also applies corrections to the internal model in terms of geometrical inconsistencies, e.g. not perfectly matching wall corners to improve the results of analysis tasks. The derived model is extruded to a user defined height and elements as doors and windows are adapted to a default height and z-position. The approach of Lewis and Séquin differs to our work in the way that we don't rely on digital CAD data but offer a possibility to create a 3D model from scanned plans.

Stojanovic et al. [Stojanovic2019] are using 3D point clouds to generated 2D floor plans as well as 3D meshes. Their approach is similar to our in the way, that they use clustering techniques to find the relevant contours, but differs by the fact, that they are only focusing on wall elements, whereas our approach deals with windows, doors and stairs as well.

Gerstweiler et al. [Gerstweiler2018] present an approach to create 3D building plans out of 2D floor plans. In contrast to our work they rely on models and heuristics to extract the structural elements, whereas our work uses clustering and contour extraction methods.

Yin et al. [Yin2009] compare different approaches on how to transform floor plans into 3D models. They do not only compare fully automated concepts, but also take a look at semi-automated processes. So et al. [So1998] present a semi-automated way for reconstructing 3D virtual buildings from 2D architecture floor plans. In contrast to our work, they focus on the general concepts and provide mathematical concepts on how to extract poly-lines from walls. Lu et al. [Lu2007] focus on the automatic analysis and integration of architectural drawings. They present concepts for recognizing typical structural objects and architectural symbols and reconstruct the original 3D model out of it. In contrast to our concept, they use predefined shapes for extracting wall lines followed by extracting other elements based on the detected wall lines. Further studies in this field are done by: Dosch et al. [Dosch2000], Ah-Soon and Tombre [AhSoon2001] and Or et al. [Or2008].

3 BACKGROUND

To achieve clustering of linear structures, several algorithmic concepts have been available in the image processing, machine learning and computer graphics domain since decades that are utilized and re-combined in this paper.

While the detection of linear structures can be achieved by edge detection, e.g. applying a Sobel high-pass filter, followed by line analysis in Hough space [Duda1972], perfectly applicable to detecting an ID card in images for scale determination [Pointner2018]. Nevertheless, thereby the continuous line definition has to be locally restricted which is then generally hard to achieve in 2D building construction plans with generally short lines at small construction elements. To detect or prolongate smaller line segments, local hysteresis concepts known from Canny edge detection [Canny1986] or the Bresenham line algorithm [Bresenham1965] known from rasterization in the computer graphics domain is applicable. Utilizing the Bresenham algorithm, a set of 2D pixels is transformed into a continuous line segment at linear run-time complexity without need for time-consuming PCA (principal component analysis)

and SVP (singular value decomposition) [Lee2006]. In the work of Von Gioi et al. [VonGioi2012] a sophisticated line detector is introduced, analyzing the local gradients and level-line fields for sub-pixel accuracy detection of continuous lines on a discrete pixel raster.

To chain up smaller line segments in an iterative manner, the orientation, location and extent of the particular line can be derived by the Minimal Rotated Rectangle algorithm [Eberly2015]. Initial bounding areas are thereby implicitly calculated utilizing the rotating calipers [Preparata1985] algorithm.

For structural analysis of lines in terms of graph analysis, only the mid course at thickness of 1px is required. While for input image gradients detected by Sobel filtering this is achievable by Canny line detection Canny edge detection [Canny1986], for existing binarized line segments generally showing a $width > 1px$, a common thinning approach is applicable, e.g. Zhang-Suen thinning [Zhang1984] constructing the line course from the derived skeleton then or by utilizing stochastic randomized erosion as thinning [Zwettler2010] for speed-up on large images and volumes.

For assembling pre-processed geometric structures such as lines the common vector-based classification algorithm k-Means clustering is applicable, too. Thereby, the clusters are refined in several iterations, where the clusters' centroids are updated and utilized for re-associating the data tuples. This process is repeated until the result converges and there is no re-association of any data tuples anymore or the maximal number of iterations is reached. Due to the randomly selected seed points it is not ensured that the algorithm finds the best solution [Macqueen1967]. To overcome this known limitation of k-Means clustering, the k-means++ algorithm [Arthur2006k] is applicable.

4 APPROACH

We present our clustering approach together with methods to transform *door*, *wall*, *window* and *stair* lines. Additionally, we present the case study, which is used for the actual evaluation of the transformation process. We are using a comma separated values (CSV) representation as input of our approach. The given data is defined for every line by five columns including the x- and y-coordinates of the startpoint, as well as the coordinates of the endpoint. In addition to that, the type of the line is also specified as *door*, *stair*, *wall*, *window* or *unknown*. The approach results in a set of typed contours that are represented by polygons.

4.1 Transformation Approach

The transformation process starts with general input steps that are used for a type-based pre-clustering of all given lines. These general steps consist of reading the

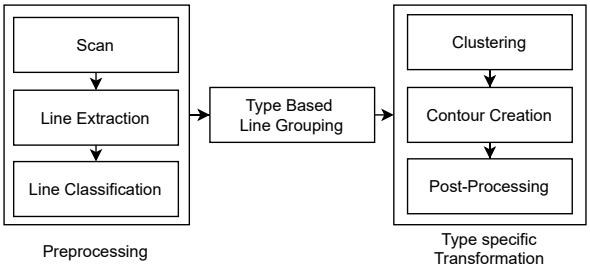


Figure 1: The base process with the preprocessing on the left side and the concept of a type specific processing on the right one.

CSV input, removing all lines typed as unknown and collecting all type equal lines. After that, (1) the actual type specific clustering, (2) the conversion into a 2D contour representation and (3) post-processing steps for removing overlapping wall and window lines are performed. This process is shown in figure 1. For those three steps, there are also some default approaches that are used on the type-pure line collections (created with algorithm 1), before the type specific steps are executed.

Algorithm 1: `getInput()`: Reads lines from CSV (csvFile) and clustering the lines by its type based on a given distance ratio.

```
Data: csvFile, ratio
Result: typed pure line collections
1 minDistance ← sizeOfPlan(lines) * ratio
2 lines ← readCsv(csvFile)
3 linesPerType ← map(type, lineList)
4 forall line in lines do
5     if line.type ≠ Unknown then
6         add(linesPerType[line.type], line)
7     end
8 end
```

For step (1), the line clustering, the method initially performs the Bresenham algorithm [Bresenham1965]. The resulting points are used to determine the distance between two lines of the given collection. If any two points of different lines are within a given distance threshold, the lines are grouped into a cluster. This process is shown in algorithm 2 and is repeated recursively until all input lines are associated to a line cluster.

Step (2), the default conversion method between a line cluster and a contour, is achieved by creating a minimal rotated rectangle of the clusters' point cloud, that consists of all start- and endpoints of the grouped lines. This is shown in algorithm 3.

Finally, the post-processing steps (3), shown in algorithm 4, for realigning overlapping windows or walls are applied.

Algorithm 2: `defaultClustering()`: Clustering of lines by the distance between the two closest points of two lines using a list of all lines, a list of already used lines, the current line and the cluster, as well as the building plan size.

```
Data: allLines, usedLines, currLine, cluster, size
Result: current processed cluster is initialized
if contains(usedLines, currLine) then
    return
end
add(usedLines, currLine)
add(cluster, currLine)
threshold ← sqrt(size.width * size.height) / 30
currPoints ← bresenham(currLine)
forall line in allLines do
    if notContained(usedLines, line) then
        linePoints ← bresenham(line)
        if smallestDistance(currPoints, linePoints)
          < threshold then
            defaultClustering(allLines, usedLines,
              line, cluster, size)
        end
    end
end
```

Algorithm 3: `defaultContourCreation()`: Creating contours of elements by using a Minimal Rotated Rectangle around the lines of the given clusters.

```
Data: clusters
Result: creates map of polygonal contour elements
contourElements ← list()
forall cluster in clusters do
    points ← list()
    forall line in cluster do
        add(points, line.start)
        add(points, line.end)
    end
    contour ← minimalRotatedRectangle(points)
    contour.type ← cluster.type
    add(contourElements, contour)
end
```

The method starts by checking if a *window*/*wall* contour is intersecting with any other *window*/*wall* contour. When the condition applies, the algorithm checks if the orientation of both contours is similar, too. As all contours are defined by rotated rectangles the orientation of them is available, too. This second check is used to avoid information loss by manipulating L- or T-formed elements as e.g. corner windows. For similarly orientated typed contours, the algorithm adapts both elements to the same height and aligns those on the virtual center line defined by the elements' center points.

Algorithm 4: `defaultPostProcessing()`: Window/Wall contour post-processing that aligns two neighboring elements by adjusting their height and rotation based on two lists of contours, one for the windows and the other for the walls.

```
Data: windows, walls
Result: realigned window/wall contours
contours ← combine(windows, walls)
forall contour1 in contours do
    forall contour2 in contours do
        if contour1.intersects(contour2) and
          areSimilar(orientation(contour1),
          orientation(contour2)) then
            height ← average(contour2, contour1)
            scaleTo(contour2, height)
            scaleTo(contour1, height)
            centerline ← line(center(contour2),
              center(contour1))
            orientation ← orientation(line)
            rotateTo(contour2, orientation)
            rotateTo(contour1, orientation)
        end
    end
end
```

In addition to the realignment of the contours, there is also a post-processing step for separating overlapping window and wall elements, that is shown in algorithm 5. This process checks every window for containment in a wall. In that case, the shorter sides of the window are extended until they are intersecting with two sides of the enclosing wall element. This process is shown in figure 2. The intersection points are used to split the wall elements into three parts, where part *A* is defined by the original top-left rectangle corner, the top-left intersection point, the bottom-left intersection point and the bottom-left original rectangle corner. The second part *B* is defined by the four intersection points and the third part *C* consists of the top-right intersection point, the top-right rectangle corner, the bottom-right rectangle corner and the bottom-right intersection point. The two outer parts *A* and *C* are used as new wall elements and the intersection area *B* is removed from the result.

4.1.1 Door specific approach

Since doors are represented as arcs in building plans, it is necessary to convert this representation to a rectangular form. In addition, doors are always surrounded by walls or windows. With this precondition, the transformation approach for door lines is based on the default clustering process and reuses its result to fit doors between surrounding wall or window clusters.

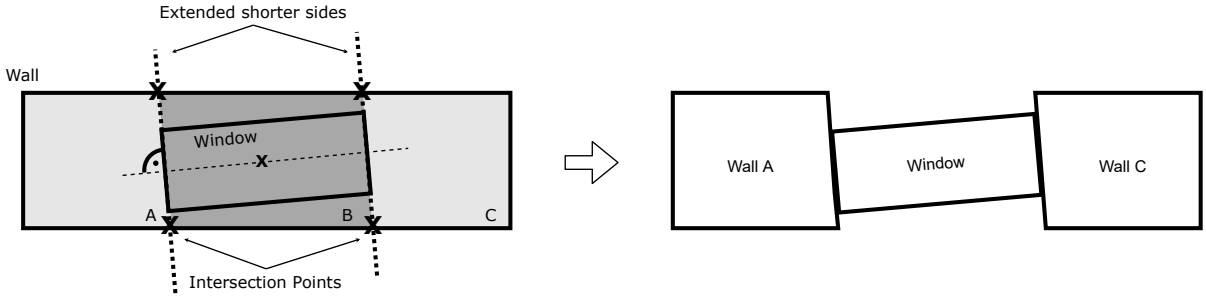


Figure 2: Process of separating a window's contour, that is contained in a wall's contour, by extending the window's shorter sides and using the intersections for cutting the wall element into three parts *A* (light-grey left side), *B* (dark-grey middle) and *C* (light-grey right side), where *B* is removed from the result.

Algorithm 5: `windowWallSeparation()`: Separation of overlapping window and wall elements, by removing the intersection area, based on two lists of contours, one for the windows and the other for the walls.

```
Data: windows, walls
Result: final wall contours
finalWalls ← list()
forall wall in walls do
    forall window in windows do
        if isContainedIn(window, wall) then
            sides ← window.shorterSides
            intersectionPoints ← intersection(sides,
              wall)
            while size(intersectionpoints) < 4 do
                extend(sides)
                intersectionPoints ←
                  intersection(sides, wall)
            end
            toRemove ←
              polygon(intersectionPoints)
            wallParts ← cut(wall, toRemove)
            add(finalWalls, wallParts[0])
            add(finalWalls, wallParts[2])
        else
            add(finalWalls, wall);
        end
    end
end
```

Algorithm 6: `doorClustering()`: Door specific clustering approach, that uses the two closest non door clusters (walls, windows) to find anchor points for the position of the pre-clustered door clusters using the default approach.

```
Data: doors, walls, windows
Result: door clusters
result ← list()
wclusters ← combine(walls, windows)
forall door in doors do
    nearestClusters ←
      getTwoNearestClusters(door, wclusters)
    nearestLine1 ← getNearestLine(door,
      nearestCluster[0])
    nearestLine2 ← getNearestLine(door,
      nearestCluster[1])
    doorCluster ← minimalRotatedRectan-
      gle(nearestLine1.center, nearestLine2.center)

    notOverlapping ← true
    forall cluster in wclusters do
        if overlapping(doorCluster, cluster) then
            notOverlapping ← false;
        end
    end
    if notOverlapping then
        add(result, doorCluster)
    end
end
```

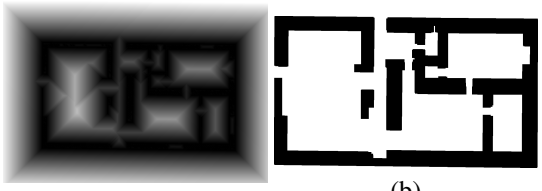
This is done by finding the two nearest lines from two different clusters. The lines' mid points are used as anchor points for the recreated door elements, which is done by fitting in a minimal rotated rectangle between those points. Since the algorithm may take incorrect lines for defining the anchor points a check verifies if the created door is overlapping any other contour. In such a case, the door is considered invalid and is removed from the result. The door specific approach is shown in algorithm 6.

4.1.2 Stair specific approach

Due to the fact that all lines of a stair are somehow connected but different stairs are separated by distance, the characteristics of stairs on building plans are quite unique. For this reason the default clustering and contour conversion approach are sufficient and it was not required to implement a stair specific approach.

4.1.3 Wall specific approach

The default clustering approach is not working for walls because nearly all wall lines are connected and it would result in a few, large clusters. Consequently, the default clustering algorithm is not applicable. Instead, the wall clustering (shown in algorithm 7) uses different image processing approaches. This process starts by creating a distance map (shown in figure 3 (a)) using all wall lines. On the resulting distance image a threshold filtering is applied (shown in figure 3 (b)), with a threshold factor based on the building plan's size. The thresholded result is further manipulated using the Zhang-Suen thinning algorithm [Zhang1984] (shown in figure 4). On the resulting skeleton image, a line segment detector is applied. These skeleton lines are then used as seed points for the clustering process. We create bounding boxes around the extracted skeleton lines and all contained input lines are added to that cluster. The size of the created bounding boxes is in turn normalized based on the size of the building plan. This process is repeated until all input lines are associated with a cluster. Finally, all clusters are checked for intersection. In the case that a cluster is completely contained in another one, these two clusters are merged.



(a) (b)

Figure 3: (a) The calculated distance map and (b) the distance map with an applied threshold.

In addition to the adapted clustering process, the wall specific approach also contains additional contour post-processing steps. Those steps are used to remove clusters which an area smaller than a given threshold that is determined by the overall size of the building plan, and merging cluster that are overlapping with at least 40%. The first of the two steps uses the area of the rotated rectangles and checks if it is greater than a threshold, which is a ratio based on the total building plan size. If the area is below the specified threshold, the contour is removed from the result. For the merging step, the size of the intersection of two overlapping clusters is utilized to determine whether the clusters are merged or not. If the intersection concerns $> 40\%$ of one the clusters, the two clusters are merged. This wall post-processing is shown in algorithm 8.

4.1.4 Window specific approach

The approach for clustering window lines (shown in algorithm 9) is based on the default implementation and uses its result as input. The window specific method uses the pre-clustered elements and applies the k-means clustering algorithm [Macqueen1967] based on the lines' orientation to create four line clusters with lines around 0°, 45°, 90° and 135°. This is done due to the assumption that windows that are close to each other have the characteristics that their lines differ in

Algorithm 7: `wallClustering()`: The wall specific clustering approach that uses Distance Map, Zhang-suen thinning and Line Segment Detection to cluster wall lines.

```
Data: walls
Result: all clusters
1  image ← drawlines(walls)
2  distanceMap ← distanceMap(image)
3  thresholdedMap ← threshold(distanceMap)
4  thinned ← thinning(thresholdedMap)
5  lines ← lineSegmentDetection(thinned)
6  clusters ← list()
7  forall line in lines do
8      cluster ← list()
9      boundingbox ← boundingBox(line)
10     forall wall in walls do
11         if containedIn(wall, boundingbox) then
12             add(cluster, wall)
13             remove(walls, wall)
14         end
15     end
16     if notEmpty(cluster) then
17         add(clusters, cluster)
18     end
19 end
20 forall cluster1 in clusters do
21     forall cluster2 in clusters do
22         if contains(cluster1, cluster2) or
             contains(cluster2, cluster1) then
23             cluster1 ← merge(cluster1, cluster2)
24             remove(clusters, cluster2)
25         end
26     end
27 end
```

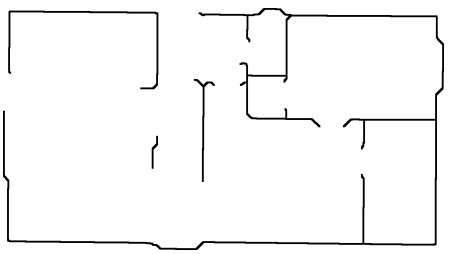


Figure 4: The line skeleton after applying the Zhang-Suen thinning algorithm on the threshold distance map image.

Algorithm 8: `wallPostProcessing()`: Wall specific contour post-processing, that removes small walls and merged overlapping walls into a single bigger wall using the size of the building plan to calculate the threshold for the merging area.

```
Data: walls, buildingPlanSize
Result: postprocessed wall contours
threshold ← getThreshold(buildingPlanSize)
forall wall in walls do
    rotatedRectangle ← rotatedRectangle(wall)
    if area(rotatedRectangle) < threshold then
        remove(walls, wall)
    end
end
forall wall1 in walls do
    forall wall2 in walls do
        if intersectionRatio(wall1, wall2) > 0.4
         then
            merged ← merge(wall1, wall2);
             remove(walls, wall1)
            remove(walls, wall2)
            add(walls, merged)
        end
    end
end
```

Algorithm 9: `windowClustering()`: Window specific clustering with line outlier removal

```
Data: windows
Result: window clusters
forall window in windows do
    k ← 4
    do
        clusters ← kMeans(window.line, k)
        k ← k - 1
    while clustersAreSimilar(clusters)
    remove(windows, window)
    addAll(windows, clusters)
end
forall cluster in windows do
    weightedCenter ←
     averagePointPosition(cluster.lines)
    averageLength ← averageLength(cluster.lines)
    medianDistance ←
     medianDistance(weightedCenter,
     cluster.lines)
    upperQuantil ← medianDistance * 1.5
    forall line in cluster.lines do
        if distance(center(line), weightedCenter) >
         medianDistance then
            remove(cluster.lines, line);
        end
        if length(line) >= averageLength * 2 then
            remove(cluster.lines, line);
        end
    end
end
```

rotation with a certain threshold. Those clusters are checked on similarity using the average line rotation per cluster. If at least two clusters are too similar (difference < 30°), the k-means clustering is repeated with a smaller number of clusters until all clusters are unique. Afterwards, outliers are removed from the clusters. This is done by two different approaches. One approach removes positional outliers and the other removes length outliers. The first outlier group is identified using the average point position of all lines' start- and endpoints of a cluster as weighted center and the median distance from the start- and endpoints to this center. A line is considered an outlier if the distance of its midpoint to the cluster's weighted center is in the upper quantile of all line distances. In addition to that, lines are removed if their length is at least twice as long as the average line length in the cluster. The resulting and filtered clusters are used as input for the default contour conversion.

4.2 Case Study

In order to evaluate the floor plan transformation, a case study is designed. In this case study we use a CSV representation of a real world 2D floor plan, which is shown in figure 6(c), as well as 100 generated, i.e. synthesized, building plans described in section 4.3, that consists of classified lines, which are then transformed into contour elements. The real world floor plan was scanned and provided with a resolution of 1358×988 pixels. Since the input image is classified before it is used in our clustering approach, it is also pre-processed and a threshold is applied to remove levels of gray and noise.

4.3 Random Building Plan Generator

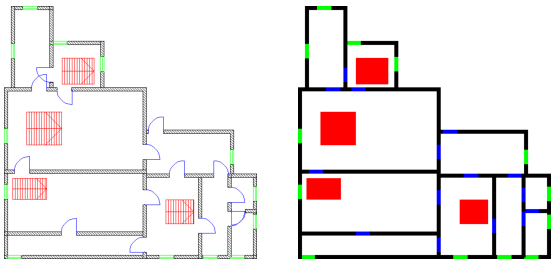
The random building plan generator allows us to generate lines that form a plausible building plan. Within the developed image to 3D building plan pipeline, this approach was initially used to train the classification module to overcome the limitation of having to train and test with the same data source. In addition to that we use the synthesized building plans to test our clustering approach. The architecture of the random building plan generator is split into two parts. At first, we implement a simple random room generator and secondly the room borders are used to place textured windows, walls and doors on them. Additionally, stairs are placed inside these boundaries.

The generation process starts by placing random rectangles inside an image. These rectangles are extended

in width and height as long as they are not touching any other rectangles. Once they hit another rectangle on a specific side, the growing on this side stops and the process is continued with the non-touching sides. The method is applied until the growing of all rectangles has finished. This process can lead to small holes where multiple rooms connect. Nevertheless, this is not an issue as such holes also occur in regular plans for chimneys or air shafts.

After the rooms are created, we start using the lines of the rectangles for further processing. In a first step we determine possible window and wall spots. We evaluate for every line if it is an inside or outside line by checking if it overlaps with a line of any other room. In this case, the overlapping area of one of the lines is removed and both lines are marked as inside lines. The region of the line is then contained in two rectangles and is marked as possible place for a door. On outside lines the entire line is marked as possible position for a window. In the next step the determined positions are used to place randomized doors and windows, as well as one entrance door that is generated at a window position. After that, we have a set of lines for doors and window. All remaining lines are marked as wall lines. For generating the stairs, we randomly add rectangles inside room areas.

In the final step of the building plan generator we apply a texture to all these lines and add a spacing wherever necessary, so that there are no overlapping areas. In the end this results in an automatically generated building plan as shown in figure 5 (a). In this image black lines represent walls, red line are stairs, green lines are windows and blue lines are doors. The final result may contain some unrealistic rooms, that have an odd aspect ratio. Nevertheless, this is sufficient for the purpose of evaluating the clustering process of classified lines. These building plans are then exported as CSV-file containing all the lines and their corresponding type. In addition to the CSV output, we also generate the corresponding contour elements show in figure 5 (b) that are used as a ground truth for the evaluation.

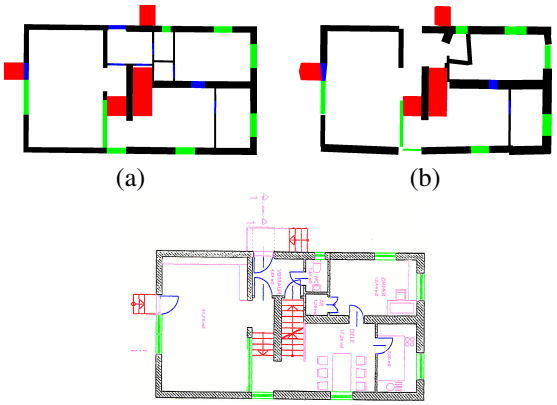


(a) (b)

Figure 5: (a) Sample of an auto generated building plan using Random Building Plan Generator and (b) the corresponding filled contour image, used as ground truth result.

5 EVALUATION

The presented approach is expected to be feasible for clustering single lines. In order to evaluate the correctness of these clusters, they are represented as filled polygons and are compared to our predefined ground truth. Such a comparison is shown in figure 6. Where (a) shows the ground truth that was manually created using an image manipulation program and is compared to (b) the automatically generated result. The evaluation is performed pixel-wise. This means, that we compare the color of each pixel from the automated approach with the ground-truth's counterpart.



(a) (b)

(c)

Figure 6: (a) The manually created ground truth, (b) the transformation result of the real world show in (c). Black segments represent walls, red segments are stairs, blue segments are doors and green segments represent windows.

The described transformation is executed on the classified floor plan shown in figure 6(c) and results in the clustered plan shown in figure 6(b). Together with the manually transformed plan as ground truth (shown in figure 6(a)), the transformation is evaluated. In addition to this real world model, we use the random building plan generator to additionally create 100 equally sized building plans together with their corresponding ground truth contour representation.

Using the ground truth plans and the results of the transformation process the differences are calculated, which is shown for the real world example in figure 7 with figure 6(a) and (b) as input. Table 1 shows the confusion matrix for the various types between the ground truths and our process results for all transformations. It shows, that the transformation works for most types with an accuracy of $> 90\%$. Nevertheless, the transformation fails on doors and cannot transform them. As the table shows, some door lines result in background pixels. The reason for that is, that if the specific transformation step cannot extract the door with a high confidence no door at the desired position is generated, as described in section 4.1.1.

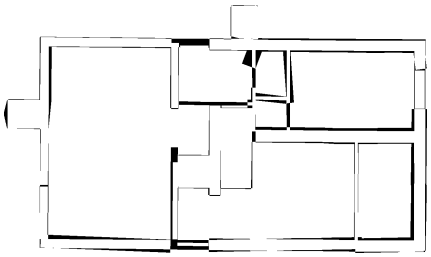


Figure 7: Difference image of the manually transformed plan shown in figure 6 (a) and the automatically transformed plan shown in figure 6 (b). Black pixels representing differences between the compared images and white pixels are equal in the images

		Automatic Transformation				
		Wall	Stair	Window	Door	Background
Ground Truth	Wall	**92.22%**	0.00%	0.63%	0.09%	7.06%
	Stair	0.04%	**98.50%**	0.00%	0.00%	1.46%
	Window	0.25%	0.00%	**99.75%**	0.00%	0.00%
	Door	0.79%	0.00%	0.00%	**7.90%**	91.31%
	Background	1.43%	0.00%	0.10%	0.04%	**98.43%**

Table 1: Confusion matrix for the evaluation of the transformation with normalized values for each type.

6 RESULTS

6.1 Accuracy

The accuracy of the single classes can be seen in the normalized confusion matrix in table 1. Beside the background, which has a very high accuracy due to the total amount of background pixels, the stairs and windows perform even better. Those are followed by walls, which do have a slightly worse accuracy. With only an accuracy of 7.9% doors are the least correctly transformed category. As described in section 5, this happens due to the fact that the transformation does not create a door if the confidence is too low. Overall we can state that the transformation accuracy is with $> 90\%$ for walls, stair and windows sufficient, only for door with around 8% it is not sophisticated for the tested example.

6.2 Performance

In contrast to the accuracy evaluation that is done on equally sized plans, the performance evaluation uses different sized building plans. For this purpose plans with diverse numbers of lines as well as variant widths and heights are generated using the random building plan generator.

Results of the performance measurements are shown in table 2. The experiments are executed with *10* warmup tests followed by additional *10* measured tests, each. The results in the table are averaged values. The results show that the performance scales with the size of the plan in a nearly linear manner. This means, that if the plan size is doubled, the run-time is around twice the time. This is also proven by the strong *Pearson* correlation between area and run-time of around *.996*, as well as in the run-time chart which can be seen in figure 8.

		Plansize (in pixel)			
Rooms	Lines	width	height	area	run-time [ms]
5	686	603	603	363609	725
6	852	603	603	363609	740
7	1011	603	603	363609	1019
8	1248	803	803	644809	1315
9	1375	803	803	644809	1393
10	1741	1029	1003	1032087	2522
12	2428	1303	1303	1697809	3847
14	2581	1529	1503	2298087	4459
16	3600	1703	1703	2900209	7809
18	2960	2003	2003	4012009	10519
20	3776	2503	2503	6265009	15743

Table 2: Performance of the transformation in ms compared to the number of lines, number of rooms and size of the building plan.

It is also affected by the number of lines in the plan. If the number of lines is increased but the size is kept constant, the run-time also increases. This does not have a big impact on the run-time, which can be seen in the slightly lower correlation of around *.896*.

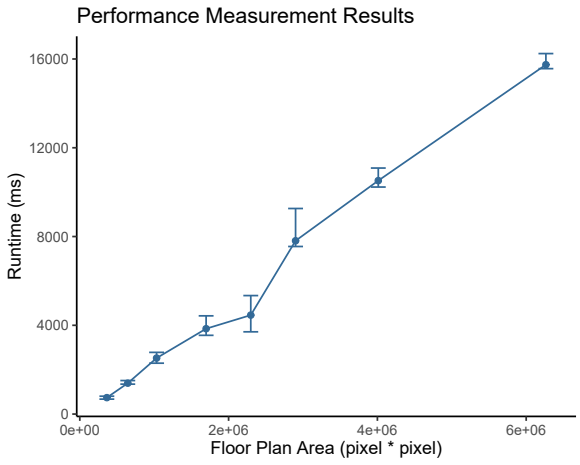


Figure 8: The run-time of the transformation grows linear with the area size of the plan.

7 CONCLUSION

In section 4.1 we answered our research question: How can classified lines be transformed to contour elements using clustering approaches?, and presented an approach on how to automatically cluster lines in the context of a 2D building plan. This concept uses algorithms from the field of computer vision and machine learning.

We evaluated that concept in section 5 using a case study consisting of a single real world floor plan and 100 synthetic building plans.

For this case study, the accuracy was evaluated in section 6.1, which showed an accuracy of $> 90\%$ for window, walls and stairs and around 8% for doors. In addition to that, a performance test was done in section 6.2. The test showed that the average run-time scales linearly with the area of the building plan.

In the future we want to extend the evaluation with a higher amount of real data as well as a higher variety of building plans. Moreover, we aim to improve the transformation accuracy for door elements. This can e.g. be achieved by changing the classification to have separated classes for normal and overlapping doors, which would then give the possibility to address these two tasks differently in the transformation. In addition to that we want to do an evaluation on scaled building plans. As we have seen a near linear scale in run-time compared to the size of the plan this should lead to a increase in performance.

8 REFERENCES

[AhSoon2001] Ah-Soon C. and Tombre K. Architectural symbol recognition using a network of constraints, in Pattern Recognition Letters Vol. 22, pp. 231-248, 2001.

[Arthur2006k] Arthur D. and Vassilvitskii S. k-means++: The advantages of careful seeding, 2006.

[Bresenham1965] Bresenham J. Algorithm for computer control of a digital plotter, in IBM Systems Journal Vol 4, pp. 25-30, 1965.

[Canny1986] Canny J. A Computational Approach to Edge Detection, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, 1986.

[Dosch2000] Dosch P, Tombre K., Ah-Soon C. and Masini G. A complete system for the analysis of architectural drawings, in Int. Journal on Document Analysis and Recognition Vol. 3, pp. 102-116, 2000.

[Duda1972] Duda R.O. and Hart P.E. se of the Hough Transformation to Detect Lines and Curves in Pictures, in Comm. ACM. 15: 11â€“15, 1972.

[Eberly2015] Eberly D. Minimum-area rectangle containing a set of points, 2015.

[Gerstweiler2018] Gerstweiler G., Furlan L., Timofeev M. and Kaufmann H. Extraction of Structural and Semantic Data from 2D Floor Plans for Interactive and Immersive VR Real Estate Exploration, in MDPI AG Vol. 6, 2018.

[Lee2006] Yun-Seok L., Han-Suh K. and Chang-Sung J. A straight line detection using principal component analysis, in Pattern Recognition Letters vol 27(14), 2006.

[Lewis1998] Lewis R. and SÃ©quin C. Generation of 3D building models from 2D architectural plans, in Computer-Aided Design Vol. 30, pp. 765 - 779, 1998.

[Liu2017] Chen L. and Jiajun W. and Pushmeet K. and Yasutaka F.. Raster-to-Vector: Revisiting Floorplan Transformation, in Conf. proc. of IEEE International Conference on Computer Vision (ICCV) 2017.

[Lu2007] Lu T, Yang H. Yang R. Cai S. Automatic analysis and integration of architectural drawings, in Proc. of Int. Journal of Document Analysis and Recognition (IJDAR) Vol. 9, pp. 31 - 47, 2007.

[Macqueen1967] MaqQueen J. Some methods for classification and analysis of multivariate observations, in Proc. of the fifth Berkeley symposium on mathematical statistics and probability Vol. 1., pp. 281-297, 1967.

[Or2008] Or S, Kin-Hong Y. and Ming M. Abstract Highly Automatic Approach to Architectural Floorplan Image Understanding & Model Generation, 2008.

[Pointner2018] Pointner A., Krauss O., Freilinger G., Strieder D. and Zwettler G. A. Model-based image processing approaches for automated person identification and authentication in online banking, in Proc. of the EMSS2018, 2018.

[Preparata1985] Preparata F. and Shamos M. Computational geometry: an introduction, 1985.

[So1998] So C. Baciu G. and Sun H. Reconstruction of 3D virtual buildings from 2D architectural floor plans, in Proc. of the ACM symposium on Virtual reality software and technology, 1998.

[Stojanovic2019] Stojanovic V., Trapp M., Richter R. and Döllner J. Generation of Approximate 2D and 3D Floor Plans from 3D Point Clouds, in Proceedings of VISIGRAPP, 2019.

[VonGioi2012] Von Gioi R., Jakubowic J., Morel J. and Gregory R. LSD: a Line Segment Detector, in Image Processing On Line Vol. 2, pp. 35-55, 2012.

[Yin2009] Yin X., Wonka P. and Razdan A. Generating 3D Building Models from Architectural Drawings: A Survey, in IEEE Computer Graphics and Applications Vol. 29, pp. 20 - 30, 2010.

[Zeng2019] Zeng Z., Li X., Yu Y. and Fu C. Deep Floor Plan Recognition Using a Multi-Task Network With Room-Boundary-Guided Attention, in Proc. of. Int. Conference on Computer Vision (ICCV), 2019.

[Zhang1984] Zhang T. and Suen C. A fast parallel algorithm for thinning digital patterns, in Communications of the ACM Vol. 27, pp. 236-239, 1984.

[Zwettler2010] Zwettler G.A., Backfrieder W., Swoboda R. and Pfeifer F. Fast Medial Axis Extraction on Tubular Large 3D Data by Randomized Erosion, in Springer Lecture Notes CCIS, Springer Vieweg, pp. 97-108, 2010.

Dynamic Sensor Matching for Parallel Point Cloud Data Acquisition

Simone Müller
Leibniz Supercomputing Centre (LRZ), Germany
Boltzmannstrasse 1
85748 Garching bei München
simone.mueller@lrz.de

Dieter Kranzlmüller
Ludwig-Maximilians-Universität (LMU)
MNM-Team
Oettingenstr. 67
80538 München
kranzlmueller@ifi.lmu.de

ABSTRACT

Based on depth perception of individual stereo cameras, spatial structures can be derived as point clouds. The quality of such three-dimensional data is technically restricted by sensor limitations, latency of recording, and insufficient object reconstructions caused by surface illustration. Additionally external physical effects like lighting conditions, material properties, and reflections can lead to deviations between real and virtual object perception. Such physical influences can be seen in rendered point clouds as geometrical imaging errors on surfaces and edges. We propose the simultaneous use of multiple and dynamically arranged cameras. The increased information density leads to more details in surrounding detection and object illustration. During a pre-processing phase the collected data are merged and prepared. Subsequently, a logical analysis part examines and allocates the captured images to three-dimensional space. For this purpose, it is necessary to create a new metadata set consisting of image and localisation data. The post-processing reworks and matches the locally assigned images. As a result, the dynamic moving images become comparable so that a more accurate point cloud can be generated. For evaluation and better comparability we decided to use synthetically generated data sets. Our approach builds the foundation for dynamic and real-time based generation of digital twins with the aid of real sensor data.

Keywords

Multi-Sensor, Dynamic Matching, Stereoscopy, Point Cloud, Real-Time, Data Acquisition, Computer Vision

1 INTRODUCTION

From the fields of autonomous driving [Tow19] and robotics [Liv12] to many other applications, a large amount of sensor data is needed to ensure the accurate and valid sensing of environmental space. Stereo cameras record distances through the synchronous recording of stereoscopic images, which allows to compute 3D points that form of a point cloud. Sensors like light detection and ranging (LiDAR) or stereoscopic systems record and process valid digital representations near real time through visual simultaneous localisation and mapping (SLAM), visual detection and tracking, as well as visual classification and recognition [You13]. The quality of such digitisations is severely restricted by sensor limitations such as range, depth resolution and image accuracy as well as different sampling rates of each sensor [Kad14]. These constraints lead to measurement fluctuations, outliers, asynchronous sensor adjustment and consequently imaging errors, which are particularly noticeable on inhomogeneous surfaces and edges. Figure 1 shows the result of a stereoscopic room acquisition by using ZED2 from Stereolabs [Ste20]. The manufacturer's software of the ZED2 already displays a real-time based point cloud as shown in the illustration. The flat walls and ceilings of digitised space seem uneven and distorted so that the entire point cloud appears inhomogeneous. In the frame on the left and right side of figure 1, the irregular arrangement of the coloured grey 3D points is clearly visible. The pattern of the orange line in the middle of the illustration shows how many measurement cycles and sequences are usually necessary to digitise a room in this form. In addition, the movement of the stereoscopic sensor leads to temporal latency and higher data processing as well as error rates so that localisation errors consequently occur [Sid03]. Regardless whether the sensor or

object is moving, the real-time based visualisation of environmental space is severely affected by movement, especially because further physical effects like lighting conditions, material properties and reflections have an influence.

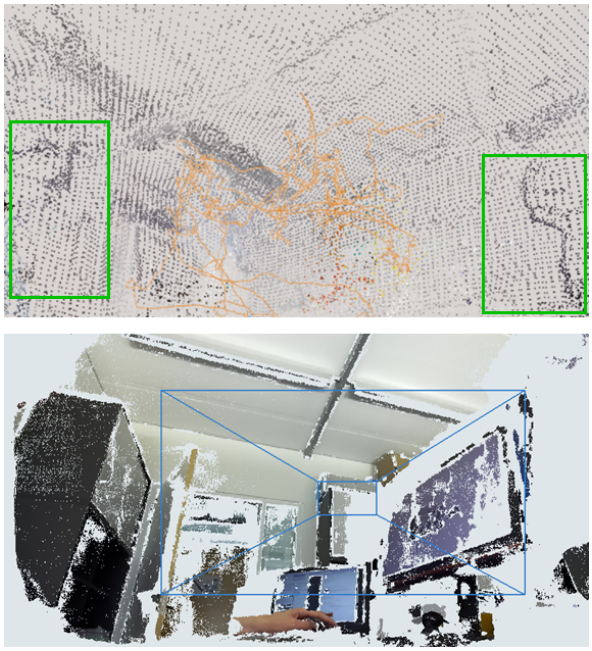


Figure 1: Point cloud visualisation captured by ZED2 stereo camera from Stereolabs [Ste20].

During the real-time acquisition of the environment, each captured frame is rendered individually. There the point cloud shows a rather high fluctuation of the 3D points due to aforementioned effects. Especially small and complex surfaces as well as moving objects can only be reproduced to a limited extent. The use of multiple stereoscopic sensors increases the density of surrounding spatial informations so that errors can be reduced by data matching. Often several statically arranged cameras are used to capture the environment [Tow19].

Our research employs the simultaneous and dynamic use of multiple camera systems. In addition to dynamic sensor implementation, we propose the real-time based generation of digital twins in three-dimensional space. We investigate the characteristics of such dynamic sensor systems and whether inhomogeneous errors can be reduced by this approach.

The paper is organised according to a fixed structure consisting of related work, fundamental concept of dynamic multiple sensors, methodology and experimental setup as well as finally results and discussion.

2 RELATED WORK

In this section we discuss the stereoscopic foundations and continue with details about the related works and our resulting motivation.

Due to the practical features and the number of integrated sensors, stereo cameras are usually used for digital environmental sensing [Ste20]. Figure 2 describes the composition of such a stereo camera. This consists of two static cameras C_L and C_R whose image orientation $\vec{v}$ is parallel to each other (for simplification of figure 2, the cameras are not drawn parallel). With the knowledge of defined distance D between the static fixed cameras C_R and C_L, it is possible to create synchronised image captures. The distance itself is a constant value that usually corresponds to the human interpupillary distance of approximately 6.5 cm [Ste20]. To avoid asynchronous sampling rates, the same settings and properties like image resolution, fps and sensitivity apply to all cameras.

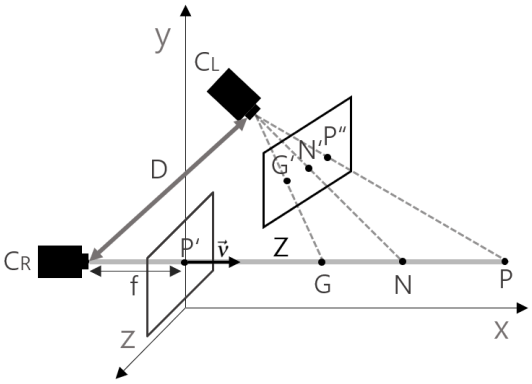


Figure 2: Schematic representation of stereoscopic depth perception. There are different pixels (G,N,P) on the optical beam Z. The three-dimensional point P and the focal points P' and P'', which can be determined by the focal length f, span the epipolar plane [Zha16]. The distance D between right camera C_R and left camera C_L is a static value.

In three-dimensional space the movement of a stereo camera is defined by translation and intrinsic rotation. A basic distinction is made between the intrinsic sensor and image rotation. While sensor rotations refer to the position of the inertial measurement unit (IMU), the image rotations relate to the orientation of the camera picture. For figure 2, only the image orientation is relevant. The intrinsic rotation matrix is defined as $\mathbf{R} \in \mathbb{R}^{3x3}$ where the product of individual rotations is called Yaw(R_ψ), Roll(R_ϕ) and Pitch(R_θ). The rotations correspond to the orientation around their respective axes [Ayk18]:

Rotation around x-axis R_ϕ:

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix} \quad (1)$$

Rotation around y-axis R_ψ:

$$R_\psi = \begin{bmatrix} cos\psi & 0 & sin\psi \\ 0 & 1 & 0 \\ -sin\psi & 0 & cos\psi \end{bmatrix} \quad (2)$$

Rotation around z-axis R_θ:

$$R_\theta = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

The camera position is expressed by $T \in \mathbb{T}^{3x3}$ where the translation directions are defined as T_X, T_Y, T_Z [Ike14], [Ska20]:

$$T = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \quad (4)$$

To create a spatial image, depth perception D is required by using disparity d in three-dimensional space $\mathbf{X} \in \mathbb{X}^{3x3}$. Since the pixel series of the camera pictures are identical, the distance between object P' and camera $|P' - P''|$ can be determined by using the epipolar geometry [Zha16].

$$d : |P' - P''| = \frac{D \cdot f}{Z} \quad (5)$$

The direction vector $\vec{v}$ of the optical axis, which is perpendicular to the camera image, depends on rotation R_ϕ, R_ψ and focal length f.

$$\vec{v} = R_\phi \cdot R_\psi \cdot \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} \quad (6)$$

In order to reproduce objects, the individual camera images must be matched. Since alignment and distance between the two cameras are static and defined, the individual rows of pixels can be compared with each other. The spherical structure of the front view of a camera creates an inhomogeneous and distorted grid. This geometric unevenness is illustrated in figure 3.

Image curvature is particularly noticeable at the edges. By using special computer vision algorithms, such as semi-global matching (SGM) [Hir11, Rob20], the pixels can be matched despite curved image edges. SGM estimates the density of the disparity map from the rectified stereo image pair. For this purpose, the maximum allowed disparity shift $(C(p,D_P))$ and the regularisation cost $PT[|D_P - D| \geq 1]$ are migrated by

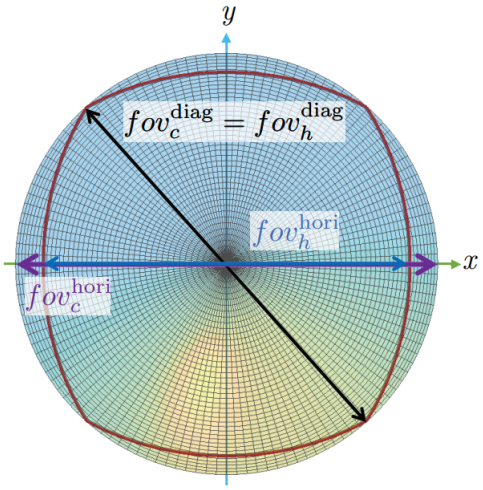


Figure 3: Front view of the camera image surface and the borders of the image content [Ayk18].

summation.

$$E_{(D)} = \sum(C(p,D_P)) + \sum(PT[|D_P - D| \geq 1]) \quad (7)$$

Depth mapping allows us to visualise the disparity as a point cloud in which the individual pixels are in temporal relation to the stereo camera. The resulting point cloud is considered as one object. There is no delimitation or allocation that would directly help to identify objects and to perceive the occluded back side of the object. In addition, only a part of the object front is visible, which allows a spatial contour. Another aspect is the quality of such point clouds.

With O'Riordan's study of colour analysis for reflective surface recognition, challenges such as image matching, false boundary as well as problems due to surface texture and obstacle detection were named [Ori18]. Marton et al. also recognised these problems and worked on a rapid surface reconstruction methodology [Mar09]. They analysed large and noisy point cloud data sets. Although they achieved good results, they also saw a need for further optimisation with regard to the high level of noise.

Chang et al. named the sparse stereo correspondences as well as their disparity limitations and have presented an efficient warping-based method for stereoscopic image retargeting [Cha11]. They have formulated these constraints as an energy minimisation problem for obtaining optimal warping fields for the images and have extended interactive stereoscopic image processing with their approach.

Another approach is provided by Zienkiewicz et al. with the method for incremental surface reconstruction from a moving camera [Zie16]. They use the concept of dynamic level where they adaptively select the best resolution of the model and fuse measurements in an

efficient multi-scale mesh representation. As a result, they declared obvious limitations in the use of height map. By optimising the three-dimensional settings and developing a more flexible multi-scale fusion method, they saw the possibility to reduce the limitations.

Due to the limited field of view afforded by a single stereo camera, synchronous and real-time 360 degrees sensing are difficult to implement. Therefore, multiple sensors like LiDAR or stereocopic systems are usually used to perceive 360 degrees.

With an autonomous vehicle plattform, Siddharth et al. worked on temporally synchronised perception and navigation data [Sid03]. The stereo cameras and LiDAR systems were statically mounted on the vehicle. From the sensor data, the environment was captured into a 3D point cloud, ground reflection map and ground truth pose of the hosted vehicle. The data sets recorded by using SLAM algorithms were also the first dynamically recorded multi data. Siddharth showed that synchronous 360 degree sensing is promising but the sensors are highly dependent on weather and light conditions.

Piatkowska et al. worked on the problem of asynchronous stereo vision for dynamic image sensors [Pia13]. They extended existing methods for event-based processing through the cooperative approach, which enables spatio-temporal and asynchronous three-dimensional reconstruction. They proved that dynamic, asynchronous and cooperative implementation is possible using a specific algorithm.

We question how the named challenges of faulty surface textures, physical influences such as light conditions, reflection, surface textures and also weather conditions can be improved. We come to the conclusion that multiple sensors can not only provide sensing but also the possibility of different perspective views. Therefore, we propose the hypothesis that synchronous dynamic multi-sensors can be used to improve real-time based point cloud visualisation. With perspective change only the relevant images would be used for evaluation. In this way, influences such as direct sunlight or limitations of depth perception could be reduced as far as possible.

3 FUNDAMENTAL CONCEPT OF DYNAMIC MULTIPLE SENSORS

This section introduces the underlying concept of multiple cameras, which are able to move dynamically through three-dimensional space and perceive the environment in the process.

Complex multi-sensor systems like IMU [Spa17] allow motion profiles of cameras to be recorded and used as a direct reference for determining position. Through detection of the sensor orientation and spatial position, images can be aligned accordingly.

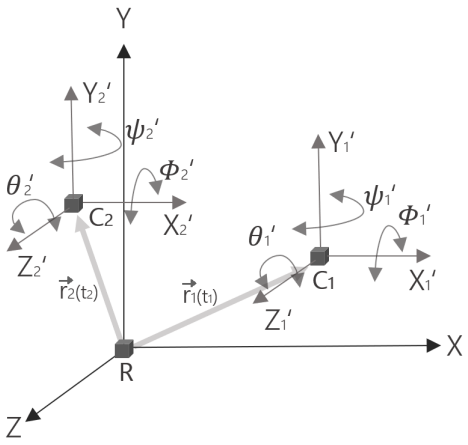


Figure 4: Schematic representation of the spatial coordinates of the stereoscopic sensors (C_1,C_2) in relation to an arbitrary reference point R: Translation directions (X,Y,Z) and intrinsic rotation (R_ψ, R_ϕ, R_θ)

In contrast to a single stereo camera, multisensor systems require an additional reference. Figure 4 illustrates the dynamic relation of the cameras to the reference, which are moving in a time-dependent vector field $\vec{r}_{1(t1)}$, $\vec{r}_{2(t2)}$. The velocity υ_1, υ_2 and acceleration a_1, a_2 of the respective position vectors can be used to determine the position at which the individual images were taken in space [Chr16].

$$\upsilon_t = \frac{\partial r_t}{dt} = (\frac{\partial X_t}{\partial t}, \frac{\partial Y_t}{\partial t}, \frac{\partial Z_t}{\partial t}) \tag{8}$$

$$a_t = \frac{\partial \upsilon(t)}{\partial t} = (\frac{\partial^2 X_t}{\partial t^2}, \frac{\partial^2 Y_t}{\partial t^2}, \frac{\partial^2 Z_t}{\partial t^2}) \tag{9}$$

Figure 5 shows the schematic reference related movement of a stereo camera that corresponds to the position recording of an IMU. In addition to the progression line (dotted line) on which the stereo camera moves, the vectorial viewing direction (arrows) can also be seen.

4 METHODOLOGY AND EXPERIMENTAL SETUP

The methodological approach consists of data initialisation and object recognition in three-dimensional space, generation and allocation of metadata records, point cloud matching, as well as rendering.

To ensure that the collected image data is correctly mapped in the point cloud, the references and stereo cameras must be initialised. Reference R is usually an object that forms the origin of the coordinate system with the parameters $T_{\text{Ref}}(0,0,0)$, $R_{\text{Ref}}(0,0,0)$. The parameters can change as soon as the object is fixed

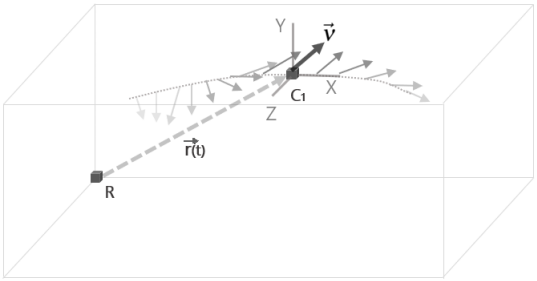


Figure 5: Dynamic movement of a vector field $r_{(t)}$ based camera C_1. The direction vector $\vec{v}$ shows the orientation based on camera C_1. The arrows show the time-dependent snapshots of the gaze alignment. The darker the arrows, the closer is the alignment to the present view. While the lighter arrows on the left side of the illustration belong to the past, the lighter arrows on the right side represent the future.

to a human being, for example, who is also moving. The generated points of the cloud are static if no adjustment is necessary. The offset positon of the coordinate system corresponds to the distance between the rendered 3D point cloud position of a single frame and the current position of the capturing stereo camera. If the relations between the reference and cameras are not initialised, the newly created point cloud of individual stereo cameras will find itself at a different location. This shifts the already recorded point clouds of the individual stereo cameras by the delta contribution of the position coordinates Δ_1, Δ_2.

$$\Delta_1 = T_{\mathrm{Ref(X,Y,Z)}} - C_{1(\mathrm{X,Y,Z})} \tag{10}$$

$$\Delta_2 = T_{\mathrm{Ref(X,Y,Z)}} - C_{2(\mathrm{X,Y,Z})} \tag{11}$$

To ensure that there is no shift in sensor position, the stereo cameras have to initialise themselves on known geometries. Initialisation always takes place as soon as a recognised geometry is recognised. To avoid this, a continuous position initialisation is necessary. One possible approach is to initialise geometries with an Artifical Intelligence (AI) based object recognition. A previously trained AI contains a three-dimensional geometric understanding of objects. If the AI recognises a known object, markers are set and stored in a map. These markings comprise localisation information of the own position and other marked geometries. A correction factor can be determined by comparing the position data of the map and the stereo cameras.

Figure 6 illustrates a possible geometric shape for initialisation. For this purpose, an identifiable geometry is provided with significant points by using computer vision algorithms. The retrievable spatial coordinates of the marker can be captured by both stereo cameras. The experimental set-up takes place in a games engine (Unreal Engine 4 [Ue21]). This has the advantage

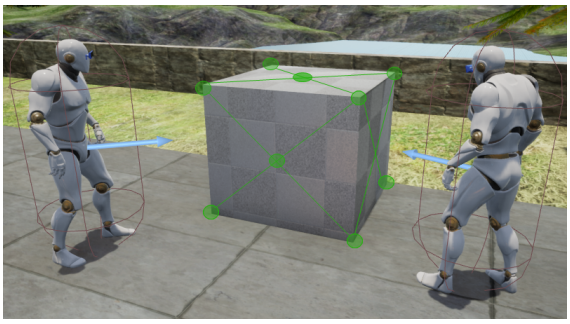


Figure 6: Initialisation of the coordinate systems of C_1 (figure left) and C_2 (figrue right) by using geometrically arranged points on an object.

that a realistic environment can be created where all parameters and physical settings are traceable. In addition, static and dynamic objects can be used. The movement sequences of the stereo cameras can also be specified in the form of avatars. The user himself serves as a reference. Since the captured images of the cameras are position and time-dependent, a new metadata record must be created. Table 1 shows the relevant content of such metadata to reproduce the temporally related informations. Thus, the connected data are momentary images that help to reconstruct the point cloud depending on its position. In addition to sensor parameters, the already detected geometric inital points in the point cloud are also collected. The time stamp enables the synchronous and asynchronous use of the cameras. In the metadata set, a further distinction is made between the rotational orientation of the image and the rotation as well as translation of the entire stereo camera.

Image data	**Sensor IMU**	**Point cloud**
Timestamp	Timestamp	Timestamp
Resolution	Translation	Initial Points
Rotation	Rotation	3D Coordinates
-	Delta	Geometric Type

Table 1: Reference-dependent metadata set. Such a metadata record is created for each sensor and captured data. In addition to image and localisation data, it also contains information about the initialisation of the coordinates.

The images of the stereo camera itself are already aligned due to their static arrangement and therefore do not need to be taken into account. For images with dynamic spacing, the perspective of the image acquisition changes. As a result, the pixel rows are no longer comparable using the SGM algorithm. Therefore, a methodology must be considered for comparing the individual images.

Description	CPU(i5-7300HQ@2.5GHz)	GPU 0(31,9GB)	GPU 1(35,9GB)	RAM(64GB)
No-load Operation	4 % - 1.64 GHz	0 %	0 %	10 %
Simulation - Mean	44% - 2.86 GHz	1 %	38 %	12 %
Simulation - Maximum	75% - 2.86 GHz	11 %	41 %	13 %
Simulation - Minimum	9 % - 1.39 GHz	0 %	11 %	12 %
Real - Mean	54% - 3.14 GHz	0 %	12 %	13 %
Real - Maximum	78% - 3.09 GHz	7 %	12 %	13 %
Real - Minimum	54% - 3.14 GHz	0 %	12 %	13 %

Table 2: Performance comparison between real and simulated multisensor data during real time operation. The no-load operation of computer is shown as a reference.

Figure 7 illustrates the dynamic distance relationship between sensor C_1, C_2 and reference R. It shows that depth perception can also be captured from stereo cameras where the field of view appears larger. Since the correct coordinates of the stereo cameras are known through the initialisation, the dynamic distance C_{12} between C_1 and C_2 can be calculated. If the image orientation of the vector field $\vec{v}$ is corrected, the SGM algorithm can be applied again and a depth image can be generated. Using the generated position-fixed depth images, a point cloud can be reconstructed that has a wider field of view. By superimposing the depth images, the localisation errors are also superimposed.

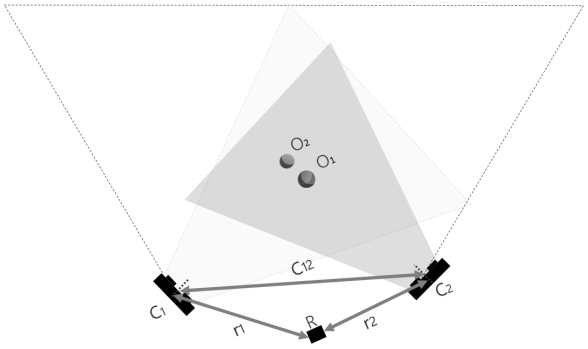


Figure 7: Two-dimensional representation of dynamically arranged stereo cameras C_1, C_2 in relation to reference R. While camera C_2 can only perceive object O_1, camera C_1 captures both objects O_1, O_2 from its perspective. In contrast to a single stereo camera, the distance r_1,r_2 between reference R and cameras C_1, C_2, respectively, are now non-static. The sensors move time-dependently through the environment. By matching C_1 and C_2, the objects can be spatially captured in a larger area.

5 RESULTS AND DISCUSSION

For the evaluation, we delineate an area on which two avatars moved randomly. The periodic vibrations during walking and randomly looking around in the environment are also simulated. Both avatars are equipped with a virtual stereo camera, which made video recordings during the entire measurement.

During this time the user acts as the reference and could move freely around the terrain. Figure 8 shows the marked movement area. It can also be seen that various objects have been installed at different locations around the simulation.

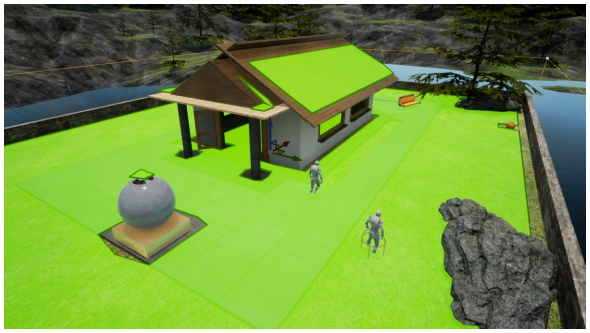


Figure 8: Marked green movement area of the avatars

Care was taken to use different sizes, different textures and reflective, transparent and rough surfaces. For the simulation we used an Intel Core i5 processor (@2.5GHz), 64GB RAM, external GPU1 Nvidia GeForce GTX 1050 Ti and the onboard GPU0 of Intel HD Graphics 630. Since real sensor data differs from synthetic data in its performance, we characterised the acquisition of real-time and dynamic based sensor data and arrived the measurement result in table 2. For measuring the real data we used two stereo cameras of the type ZED2 from Stereolabs and the manufacturer software of ZED2. For the simulation, we recorded test images of both stereo cameras, which contained position data and image data. During the process, the depth images and 3D points were calculated. In comparison to the simulation, we recorded the image and position data for the both ZED2 and also calculated the depth perception and 3D points. On average, the performance between real and synthetic data was not far apart. While the simulated data required a higher GPU (38%), the CPU load for real sensor data was somewhat higher (54%). It was noticeable that real multisensors have a higher sensitivity. In the static state, there is only a slight fluctuation in power. As soon as the stereo cameras were moved or rapid object movement occurred, the CPU load increased up to 78%.

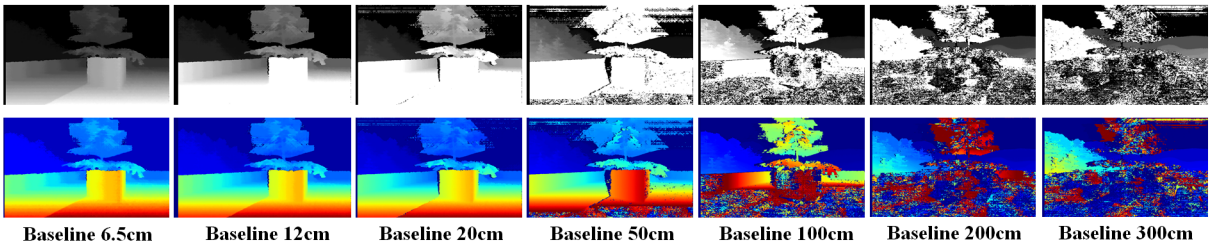


Figure 9: Display of the disparity by increasing the distance between the cameras.

Simulating the motion sequences of synthetic data has shown that the jerky rotations and oscillations caused by the up and down movement influence depth perception.

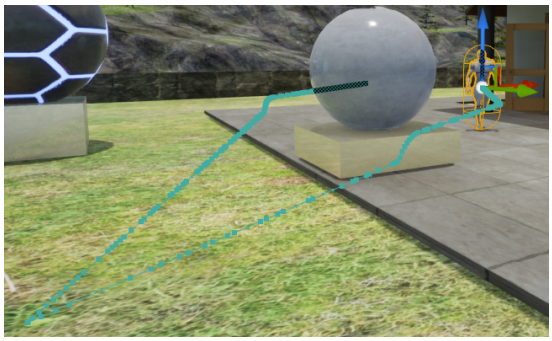


Figure 10: Movement sequence of the stereo camera C_1.

This showed a speed-dependent change in performance. In contrast, the performance of the synthetic data was not influenced by movement sequences, but there was a strong fluctuation in performance due to several processes running simultaneously.
Basically, the technology works in soft real time. The more visualisations are based on real time, the further one moves away from real time. However, there are ways to achieve real-time processing. During the measurements, noticed that the real-time processing of dynamic sensor data depends on a number of factors. The speed of processing depends on the platform. A scientific platform like Python is significantly slower than a low level based platform like C/C++. Furthermore, a better result can be achieved by using other hardware resources. When using embedded systems, this means that a master-slave system could be essential. In the case of a computer, a GPU would be a good choice. Software technical adaptations, such as parallelising work processes as well as compressing and timing algorithms, produce better results in real time processing. In future Work, we want use the methodologies to reach hard real time.

Figure 10 shows a random movement pattern of the stereo cameras where a pointed inlet of the movement line is visible.

We found that the simulated movements were too jerky. It leads to distorted images and consequently to reduced depth perception. Fast movements and natural vibration are also present in real stereo cameras, but the images are already stabilised by firmware. Thus, for the simulation we need to stabilise the cameras or dampen the movements. Due to the random movements, the cameras were only a few moments in right position for the matching.

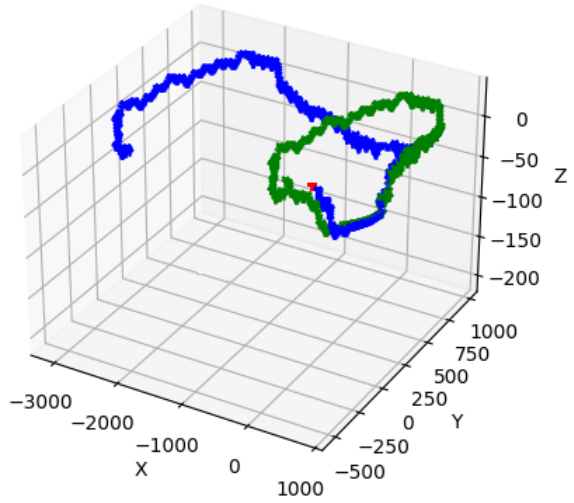


Figure 11: Movement sequence of the stereo cameras (C_1 - Blue) and (C_2 - Green) in dependency of reference (R - Red).

In addition, we analysed the disparity when the distance between the cameras was increased. Figure 9 shows the results of the changed baseline between the cameras. For better illustration we show the disparity in grey (upper row) as well as in colour (bottom row). We increased the distance from 6.5 cm to 300 cm. To ensure comparability, the cameras were arranged parallel to each other. The figure shows that the increase in distance leads to a higher dispersion. From a baseline of 50 cm, a scattering in the near range becomes visible. With the further increase in distance, the dispersion increases from near to far range. Since we kept the parameters constant, it is unclear whether parameter adjustment would lead to a better result.

It can also be seen that the far range in the disparity becomes more visible with increasing distance. There is a proportionality between the baseline and depth visibility of the 3D points.

We conclude that the position matching worked well where no incorrect positions were transmitted. The result of the movement pattern of the both stereo cameras and the reference is shown in figure 11. The illustration shows that the stereo cameras are only initially close to each other. Since there is no equally identifiable geometry with increasing distance, we come to the following consideration:

The synchronous implementation of a dynamic multi-sensor systems is only possible for a limited distance. As soon as the sensors are too far apart, we lose the connection to the object. An expanding approach is to implement the asynchronous matching of dynamic multi-systems. For this implementation it is necessary to store relevant images with the timestamp and position in a map. By using an AI, the best positioned images could be retrieved and matched with the images from the stereo camera.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented the concept of the synchronous and dynamic use of multiple stereo cameras. Our motivation was to improve challenges such as faulty surfaces and textures, physical influences such as light conditions, reflection, surface textures and also weather conditions with the help of the expansion of sensors. For this purpose, we use a simulation environment that contained both static and dynamic geometries as well as different textures and physical properties. We used stereo cameras as test sensors, each fixed to avatars from the simulation environment.

Dynamic matching of sensors is a promising technique that can be use to optimise point clouds. We have discovered that the synchronous implementation can only be realised for limited distances because the objects are no longer visible as the distance increases. Since we consider the approach of asynchronous dynamic sensor matching, we will continue our research work in this direction. It had already been shown that there is a working range in which the distances can be varied without getting a worse result. However, the working range can be adjusted by modifying the parameters of point cloud processing. In our future work we will build on this knowledge and carry out the parameter modification. For this purpose, we will use an AI that recognises geometries for initialisation as well as stored images of different postions. We will further optimise the simulation and make adjustments in the area of data storage and data processing. In addition to AI, we will use an intelligent database to map images and geometries. We want to identify the minimum and maximum dynamic distance for sensory perception of objects. Our measurement has already shown that objects become smaller or no longer visible as the distance increases. From this conclusion, we want to research the dynamic systems behave with overproportionally large objects.

7 REFERENCES

[Ayk18] Aykut T., Burgmair C., Karimi M., Xu J., Steinbach E., Delay Compensation for Actuated Stereoscopic 360 Degree Telepresence Systems with Probabilistic Head Motion Prediction. 2018 IEEE Workshop on Applications of Computer Vision (WACV), https://ieeexplore.ieee.org/document/8354326

[Cha11] Chang C.H., Liang, C.K. and Chuang, Y.Y., Content-aware display adaptation and interactive editing for stereoscopic images. 2011 IEEE Transactions on Multimedia, 13(4) pp. 589-601., DOI: 10.1109/TMM.2011.2116775

[Chr16] Christian B., Pucker N., Mathematische Methoden in der Physik. 2016 Springer, DOI: 10.1007/978-3-662-49313-7

[Hir11] Hirschmüller H., Semi-Global Matching Motivation, Developments and Applications. 2011 Photogrammetric Week 11 pp. 173-184., https://elib.dlr.de/73119/

[Ike14] Ikeuchi K., Computer Vision. 2014 Springer. p 249., DOI: 978-0-387-30771-8, https://www.springer.com/de/book/9780387307718

[Kad14] Kadambi A., Bhandari A., Raskar R., 3d depth cameras in vision: Benefits and Limitations of the Hardware With an Emphasis on the First and Secon Generation Kinect Models. 2014 Springer pp. 3-26, https://web.media.mit.edu/ achoo/tr/3d_benefits_limits.pdf

[Liv12] Livatino S.,Banno F., Muscato G., 3-D Integration of Robot Vision and Laser Data With Semiautomatic Calibration in Augmented Reality Stereoscopic Visual Interface. 2012 IEEE Trans. on Industrial Informatics, https://ieeexplore.ieee.org/abstract/document/6062673

[Mar09] Marton Z.C., Rusu R.B., Beetz M., On Fast Surface Reconstruction Methods for Large and Noisy Point Clouds. 2009 IEEE international conference on robotics and automation pp. 3218-3223, https://ieeexplore.ieee.org/document/5152628

[Ori18] O′Riordan A., Newe T., Dooly G., Stereo Vision Sensing: Review of existing systems. 2018 12th International Conference on Sensing Technology (ICST), https://ieeexplore.ieee.org/document/8603605

[Pia13] Piatkowska E., Belbachir A.N., Gelautz M., Asynchronous stereo vision for event-driven dynamic stereo sensor using an adaptive cooperative approach. 2013 International Conference on Computer Vision Workshops (ICCV Workshops) pp. 45-50, https://ieeexplore.ieee.org/document/6755878

[Rob20] Roboception GmbH, Stereo-Matching. https://doc.rc-visard.com https://roboception.com/de, last visited January 13th 2021

[Sid03] Siddharth A., Ankit V., Gaurav P., Wayne W., Kourous H., McBride J., Ford Multi-AV Seasonal Dataset. 2003 Cornell University, http://arxiv.org/pdf/2003.07969v1

[Ska20] Skala V., Karim S.A.A, Kadir E.A., Scientific Computing and Computer Graphics with GPU: Application of Projective Geometry and Principle of Duality, International Journal of Mathematics and Computer Science. 2020 International Journal of Mathematics and Computer Science, Vol.15, No.3, pp.769-777, http://afrodita.zcu.cz/ skala/publications.htm

[Spa17] You Z. - Space Microsystems, Micro/nano Satellites, Inertial Measurement. 2017 Sciencedirect, https://www.sciencedirect.com/topics/engineering/inertial-measurement, last visited January 13th 2021

[Ste20] Stereolabs - ZED2, https://www.stereolabs.com/zed-2, last visited January 13th 2021

[Tow19] Fernandes A., Maheshkumar H., Yang K., Sijo V.M., Vinayagam T., 3D Object Detection for Autonomous Vehicles. 2019, https://towardsdatascience.com/3d-object-detection-for-autonomous-vehicles-b5f480e40856, last visited January 13th 2021

[Ue21] Epic Games Inc. - Unreal Engine, https://www.unrealengine.com/en-US, last visited January 13th 2021

[You13] You L., Ruichek Y., Cappelle C., Optimal Extrinsic Calibration Between a Stereoscopic System and a LIDAR. 2019 IEEE Trans. on Instrumentation and Measurement 62(8) pp. 2258-2269., DOI: 10.1109/TIM.2013.2258241

[Zha16] Zhang Z., Epipolar Geometry. 2016 Springer, https://doi.org/10.1007/978-0-387-31439-6_128

[Zie16] Zienkiewicz J., Tsiotsios A., Leutenegger S., Davison A., Monocular, Real-Time Surface Reconstruction using Dynamic Level of Detail. 2016 IEEE Fourth International Conference on 3D Vision (3DV) pp. 37-46. (3DIM-PVT), https://ieeexplore.ieee.org/document/7785075

Deep Light Direction Reconstruction from single RGB images

Markus Miller
Dept. of Computer Science and Mathematics
University of Applied Sciences Munich
Lothstr. 64
D-80335, Munich, Bavaria
markus.miller@hm.edu

Alfred Nischwitz
Dept. of Computer Science and Mathematics
University of Applied Sciences Munich
Lothstr. 64
D-80335, Munich, Bavaria
nischwitz@cs.hm.edu

Rüdiger Westermann
Chair of Computer Graphics and Visualization
Technical University Munich
Boltzmannstr. 3/II
D-85748, Garching, Bavaria
westermann@tum.de

ABSTRACT

In augmented reality applications, consistent illumination between virtual and real objects is important for creating an immersive user experience. Consistent illumination can be achieved by appropriate parameterisation of the virtual illumination model, that is consistent with real-world lighting conditions. In this study, we developed a method to reconstruct the general light direction from red-green-blue (RGB) images of real-world scenes using a modified VGG-16 neural network. We reconstructed the general light direction as azimuth and elevation angles. To avoid inaccurate results caused by coordinate uncertainty occurring at steep elevation angles, we further introduced stereographically projected coordinates. Unlike recent deep-learning-based approaches for reconstructing the light source direction, our approach does not require depth information and thus does not rely on special red-green-blue-depth (RGB-D) images as input.

Keywords

Light, source, direction, estimation, reconstruction, RGB, deep learning.

1 INTRODUCTION

In the past decade, augmented reality (AR)-capable hardware and virtual reality (VR) devices have become increasingly available. Successful AR applications should create an immersive user experience, as immersion in AR is important to prevent a barrier between the virtual world and real world that can impede user's acceptance of AR. To minimise this barrier, it is important to avoid mismatches between virtual and real objects, such as illumination deviations. To achieve consistent illumination between virtual and real objects, virtual illumination in an AR application must adapt to the real illumination conditions.

Depending on the illumination model used to render virtual objects, virtual illumination may consist of an emissive and reflective light term, known as the rendering equation (Kajiya, 1986). The reflective term is an integral over the entire positive hemisphere above a given point, containing the bidirectional reflectance distribution function (BRDF) and incident light. To begin with a simple scenario, we restrict our approach to situations in which only one infinite point light source illuminates the scene. Thus, we neglect the emissive light term of the rendering equation, more complex lighting situations (e.g. extended, textured and multiple light sources) and indirect illumination from other surfaces. However, with an infinite point light source, we can obtain illumination conditions that contain the most important elements for three-dimensional spatial perception. In our study it is thus sufficient to focus on the light direction to be reconstructed to achieve virtual illumination while providing consistency for an immersive AR experience. We further disregard the intensity of the light, as it does not affect the illumination unless it exceeds a certain range, which constitutes a special case to be addressed in future work.

Illuminated scenes, as perceived by humans, are the result of the interaction of several parameters present in a scene. By observing the shading and light reflection of an object's surface, the human brain can estimate the characteristics and shape of the surface. The human

brain can further estimate the origin of the main light source illuminating a scene, which is usually the sun. Deep neural networks (DNNs) can also learn to estimate a light's origin from visual input (Section 2) and can even reformat this information to be used as a parameter in virtual illumination models. Immersive and responsive AR applications may require updated illumination parameters for each frame; as a result, approaches with less complex network architectures are preferable. In addition, classification approaches require a large number of classes to reconstruct a light's origin with satisfactory resolution; therefore, creating suitable training datasets is a cumbersome task. In contrast, regression approaches yield continuous output values and therefore only require as many output neurons as the number of parameters to be reconstructed.

Considering the aforementioned requirements, we propose a deep-learning-based regression approach to reconstruct the light direction in a scene given as azimuth and elevation angles (ϕ, θ) from RGB input images using a modified VGG-16 convolutional neural network (CNN). In addition, we introduce stereographic coordinates (s_x, s_y) instead of angular coordinates (ϕ, θ) to reduce inaccuracies due to the coordinate uncertainty occurring at steep elevation angles. Our approach shows that it is plausible to reconstruct the light direction in a scene from a single RGB image of the illuminated scene using a CNN. Our approach does not require additional depth information and thus can be used on devices without special depth sensors.

The main contributions of our proposed approach can be summarised as follows:

- Reconstruction of the dominant light source direction is possible using only RGB input images, and does not require special RGB-D information.
- A stereographic coordinate representation significantly improves the estimation results.

Our approach exceeds existing learning-based approaches for inferring light direction from images (Kán and Kaufmann, 2019) in that it enables reconstruction from RGB images. This is achieved by increasing the amount of training data and using pre-trained model weights, which has been successfully applied to a similar problem (Marques et al., 2018), and inferring the light direction in a stereographic coordinate representation.

2 STATE OF THE ART

Classical image processing approaches reconstruct the location of a visible light source within an image (Laskowski, 2007) or determine the light direction with additional user input (Lopez-Moreno et al., 2009). However, this is impractical for immersive AR applications.

More recent approaches utilise DNNs to reconstruct light source parameters, such as the direction, location or intensity of the light. Using the association between an object and its shadow, it is possible to determine the screen space light direction (Wang et al., 2020), which can be converted into world space coordinates with additional camera parameters. In addition, generative adversarial networks are useful tools to directly create artificial content. They also can add missing shadows to marked virtual shadowless objects in real scene images (Liu et al., 2020) and output a complete AR scene image. AR content can also be created with image-based lighting (IBL) renderers, which utilise environment textures containing the illumination details of a scene. Real scene environment textures can be derived from RGB input images using regression (Gardner et al., 2017) and continuously loaded into video memory to match illumination changes in the real scene. LeGendre et al. (2019) extended this concept by deriving more detailed textures for IBL renderers from RGB images using an encoder-decoder network. Garon et al. (2019) introduced a DNN that uses an RGB image and an image location to estimate a fifth order spherical harmonic representation of the lighting, which can be used to illuminate virtual objects placed at this location. A more straightforward approach to create AR content, is to simply place a virtual light source in the virtual scene that is combined with the real scene. To determine where to place the virtual light source to match the real illumination conditions, either the location or direction of the light source is required. The light direction can be reconstructed by classifying real-scene RGB images into directional classes using a neural network (Pemasiri et al., 2015).

Marques et al. (2018) used a similar concept in their approach to overlay a VR scene with an image of a user's hand pose. Because the real illumination conditions were embedded in the hand pose image, it was more practical to simply adjust the virtual illumination to match the real conditions. To achieve this, the authors trained a residual CNN (ResNet) starting with initial model weights pre-trained on the ImageNet and COCO datasets to classify the point light source in the VR scene that was most suitable for producing similar illumination. With 100 possible point light sources available in the virtual scene, their network achieved a top 1 accuracy of approximately 82% in classifying the correct point light source.

Kán and Kaufmann (2019) proposed a regression model to reconstruct continuous azimuth and elevation angle values indicating the dominant light direction in real scenes using RGB-D input images for their ResNet. They also conducted experiments using the RGB com-

ponents of their RGB-D dataset; however, their network failed when it relied solely on these three colour channels. They integrated their network into an AR application and achieved a mean angular error of approximately 28 °and an inference time of approximately 380 ms while running on a central processing unit.

Both approaches used ResNet to avoid exploding or vanishing gradients and both trained their networks with synthetic training data. In contrast, Marques et al. (2018) generated images of hand poses that were illuminated from different angles with the Unreal Engine, Kán and Kaufmann (2019) rendered five different objects illuminated from different angles using a Monte Carlo path tracing renderer. Kán and Kaufmann (2019) further attempted to include real training images; however, their network performed best when trained purely with synthetic data. Apart from the additional image channel and generation process, the main difference between the training data used in these approaches was the dataset size. Kán and Kaufmann (2019) used a small synthetic training set of 23,111 images. In addition, they had 5,650 real images available that were omitted for training. Marques et al. (2018) had a synthetic dataset of 83,799 images, from which they used 54,471 images as training data for their pre-trained ResNet.

The rendering equation (Section 1) suggests all required information to be present in an RGB image to reconstruct the illumination origin in a scene, however, given the unsuccessful experiment of Kán and Kaufmann (2019) using only RGB information, we begin with a simple scenario to investigate if RGB images provide enough information to reconstruct the dominant light direction. Considering the different dataset sizes and training strategies of Kán and Kaufmann (2019) and Marques et al. (2018), a DNN with model weights pre-trained on a large dataset as initial values for training and a large number of training examples appears to be a reasonable basis for a regression approach that aims to reconstruct the dominant light direction from RGB images.

3 LIGHT DIRECTION INFERENCE

In our proposed approach, we aim to reconstruct the azimuth and elevation angles (ϕ, θ) of the dominant light direction from RGB input images. This is similar to the approach of Kán and Kaufmann (2019); however, our approach focuses on light direction reconstruction using only RGB information and explicitly omits the depth information. We assume that the network is capable of reconstructing the light direction without additional depth information. For this, we use an ImageNet (Russakovsky et al., 2015) pre-trained VGG-16-like CNN (Simonyan and Zisserman, 2014) as a regression model, and train it with a dataset consisting of synthetic and real images (Fig. 1). Considering the real-time requirements for immersive and responsive AR applications, we decided to use a VGG-16-like network, as this architecture achieved the best performance relative to its complexity in the ImageNet competition (Russakovsky et al., 2015). By greatly increasing the dataset size, we were able to improve the reconstruction performance of the network for (ϕ, θ) using RGB input images.

We start with a network to predict continuous values (ϕ, θ) for azimuth and elevation in the range $(0°, 0°)$ to $(360°, 90°)$. The influence of the azimuth angle ϕ on the prediction error of the network decreases as the elevation angle θ gradually reaches $\theta = 90°$, and ϕ loses any influence on the prediction error, as $\theta = 90°$ denotes the pole of the hemisphere. Thus, ϕ cannot be properly estimated by the network, leading to an increased angular error at steep values of θ, as illustrated in Fig. 7b. To compensate for this, we convert (ϕ, θ) into stereographic coordinates (s_x, s_y) ranging from $(-1, -1)$ to $(1, 1)$, and train a second network predicting the light direction in stereographic coordinates, as they do not suffer from coordinate uncertainty.

3.1 Stereographic Coordinates

A stereographic representation (s_x, s_y) of the angular coordinates (ϕ, θ) is introduced to avoid coordinate uncertainty at $\theta = 90°$. The coordinate uncertainty refers to an undefined value of ϕ at $\theta = 90°$, as any value of ϕ denotes the pole of the sphere, leading to ϕ not able to be properly estimated by the network, as ϕ no longer has any meaningful influence on the prediction error.

A stereographic projection (Fig. 2) projects spherical coordinates onto a circular projection plane E_p. The southern pole S of the spherical coordinate system is the projection centre, and as such, undefined in the stereographic domain. However, this is not a problem, as any light source located in the southern hemisphere would shade the entire scene, not providing any useful information for estimating its direction. From S, a given point A is projected onto E_p resulting in A'. The distance m from the origin O' of E_p to A' computes (s_x, s_y) by scaling $\cos(\phi)$ or $\sin(\phi)$, respectively. The spherical coordinate space can be expressed by its origin O and radius r. By assuming that the domain region is half of a unit sphere, one can assume $r = 0.5$, as it simplifies the computation of (s_x, s_y) in

$$m(\theta) = 2 \cdot r \cdot \tan\left(\frac{90° - \theta}{2}\right) \tag{1}$$

$$= \tan\left(\frac{90° - \theta}{2}\right) \tag{2}$$

$$s_x(\phi, \theta) = m(\theta) \cdot \cos(\phi) \tag{3}$$

$$s_y(\phi, \theta) = m(\theta) \cdot \sin(\phi) \tag{4}$$

To compare the results of Net_{s_x, s_y} using stereographic coordinates to the results of $\text{Net}_{\phi, \theta}$, the predicted stere-

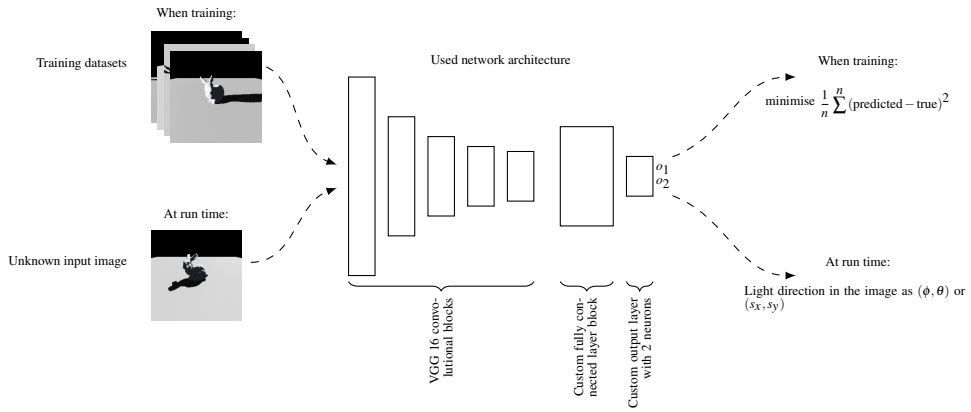


Figure 1: Diagram of network architecture and training and test procedures. The architecture utilises the standard VGG-16 convolutional blocks and a custom fully connected block and output layer with two neurons. Here the light direction is inferred from red-green-blue (RGB) input images as (ϕ,θ) or (s_x,s_y), respectively.

ographic coordinate values (s_x,s_y) can be transformed back into angular values (ϕ,θ) using

$$m(s_x,s_y) = l(s_x,s_y) = \sqrt{s_x^2+s_y^2} \quad \forall\, m \neq 0 \tag{5}$$

$$\theta(s_x,s_y) = 90 - 2\cdot\arctan\left(l(s_x,s_y)\right) \tag{6}$$

$$\phi(s_x,s_y) = \arcsin\left(\frac{s_y}{l(s_x,s_y)}\right) \tag{7}$$

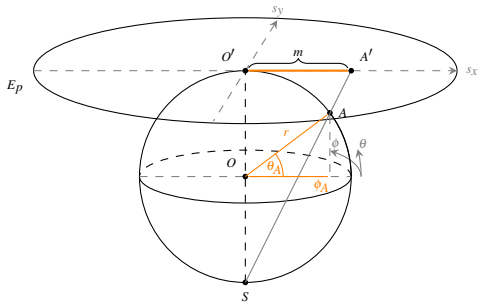


Figure 2: Depiction of stereographic projection using example $A = (\phi_A,\theta_A)$ with $\phi_A = 0°, \theta_A = 37°$, resulting in stereographic coordinates $A' = (s_{xA},s_{y_A}) = (0.4986,0)$.

4 NETWORK ARCHITECTURE AND TRAINING

We start with a pre-trained VGG-16 CNN in Keras and modify the fully connected (FC) layers of the network architecture to meet our requirements. For each of the two light direction representations, one dedicated network is trained and optimised: the first network $\text{Net}_{\phi,\theta}$ is predicting the light direction in angular values of (ϕ,θ) and the second network Net_{s_x,s_y} is predicting stereographic coordinates (s_x,s_y).

Prior to training, well-performing hyperparameter combinations are identified. To achieve this, the FC layers and the soft-max output layer are removed from the standard VGG-16 architecture, and only the five convolutional blocks are retained. The train flag is disabled on these five convolutional blocks during training; thus, all pre-trained weights are kept. After the convolutional blocks, a dynamically generated FC subnetwork is added, which consists of a FC input layer, a variable number of hidden layers and a FC output layer containing two neurons. The dynamically generated subnetwork is created just before training using parameters from a range of hyperparameters. Optimisation is performed using Talos[1]. Talos runs a grid search across the entire specified set of hyperparameter values, returning a list of hyperparameter combinations and trained model weights for each FC subnetwork. Each resulting subnetwork is trained for a single epoch with a uniform learning rate that is automatically adjusted to the used optimiser on a reduced synthetic training dataset of 20,000 images. The reduced dataset is split into a training and validation set with a ratio of 80:20. After the optimisation process, the two hyperparameter combinations and subnetworks with the best estimation performance are selected for further refinement.

To refine the two most accurate networks and determine the most suitable network architectures, the fifth convolutional block is unlocked to be trained, simultaneously reducing the learning rate to $1/1,000$ of the optimised rate. Thus, the fifth convolutional block is able to adapt to the new task while maintaining and improving what was previously learned by the dynamic FC layers. To make the network more robust to input images of varying brightness and representing objects of varying sizes

[1] Autonomio Talos [Computer software]. (2019). Retrieved from http://github.com/autonomio/talos.

Hyperparameter	$\text{Net}_{\phi,\theta}$	Net_{s_x,s_y}
optimiser	Adam	Adam
batch size	32	32
uniform learning rate	2	1
hidden layers	0	0
FC input neurons	4,096	4,096
activation	leaky ReLU	ReLU
dropout	0.25	0.25

Table 1: Hyperparameters of the resulting networks.

and locations, the training data are augmented before use in the training process by shifting the images vertically and horizontally, zooming in and out and varying the brightness. The refinement training is performed for up to 400 epochs, reducing the learning rate by a 10th if the validation mean absolute error (MAE) did not decrease over 13 epochs. If the validation MAE did not decrease over a total of 20 epochs, the training is stopped. All networks use a mean squared error loss function. Each network is trained with three different datasets: one purely synthetic, one purely real and one mixed dataset. The synthetic training dataset contains 100,000 images from the synthetic image dataset, while the real training dataset contains 800 images from the real image dataset[2]. The mixed training dataset is a combination of the real and synthetic image datasets, containing 800 real and 99,200 synthetic images.

Finally, all resulting fully trained network architectures and hyperparameter sets are tested on synthetic and real test sets and the best in each category is selected as the final architecture and hyperparameter combination (Table 1) for $\text{Net}_{\phi,\theta}$ and Net_{s_x,s_y}. The test sets consist of images the networks had not seen before and contain 10,000 synthetic images and 61 real images, respectively. Training and testing are performed on NVidia RTX 2080 Ti and NVidia Titan XP graphics processing units.

4.1 Datasets

Kán and Kaufmann (2019) used 23,111 synthetic images in their training but were unable to achieve a satisfactory reconstruction using only the RGB components of their training data. Marques et al. (2018) used a total of 83,799 synthetic RGB images and achieved satisfactory classification results. Hence, we use a large dataset in our approach, assuming that approximately 150,000 real and synthetic images (one third to be designated as test data and two thirds as training and evaluation data) are sufficient. When creating the datasets, the data are directly labelled using the angular values (ϕ,θ) of the configured camera and light settings. For the stereographic network, the angular labels $\Lambda_{\phi,\theta}$ are converted into stereographic labels Λ_{s_x,s_y} (Section 3.1).

4.1.1 Synthetic Dataset

To create the synthetic image dataset, a simple scene is created using the Unreal Engine that is composed of a single object placed on a base surface illuminated by a directional light source. Due to the wide variety of possible light directions, identifying shadow bias values that do not produce artefacts is cumbersome. Therefore, the RTX shadow capabilities of the NVidia RTX 2080 Ti are used for rendering the shadows. Five models of varying complexity are used as the centre object: a box, a cone, the Stanford bunny, the Stanford Buddha and a sphere (Fig. 3). Suitable physically based rendering (PBR) surface materials (Karis and Epic Games, 2013) are assigned to the base surface and the models to capture the material structure of their real-world counterparts.

A total number of 1,225 different light directions are obtained by illuminating the scene with a directional light source L from angles evenly distributed around the centre object. The direction values of L (ϕ_L,θ_L) range from $(0^\circ,5^\circ)$ to $(360^\circ,90^\circ)$ with a step size Δ_L of $(5^\circ,5^\circ)$. Light directions from below are not considered. Duplicate images at light elevation angles of $\theta_L = 90^\circ$ are omitted. At elevation angles of $\theta_L = 90^\circ$, a single image using an azimuth value of $\phi_L = 0^\circ$ is generated. The camera C is placed in a similar fashion as L at a fixed distance with spherical coordinates (ϕ_C,θ_C), ranging from $(0^\circ,1^\circ)$[3] to $(360^\circ,90^\circ)$ with a step size Δ_C of $(45^\circ,30^\circ)$, resulting in 32 different camera positions. Using the same camera settings for point-symmetric objects, such as the sphere, would result in duplicate images. Therefore, redundant camera settings are omitted, reducing 32 different camera positions to four positions placed around the sphere. Though having an axis-symmetric model structure, the cone object appears different from every angle due to its wrinkled paper surface material and is therefore rendered with the entire set of camera positions.

Combining camera positions and light directions, 39,200 images of the box, cone, Stanford bunny and the Stanford Buddha, and 4,900 images of the sphere are obtained. Therefore, the entire synthetic dataset contains 161,700 images that are directly captured in the required resolution of 224×224 pixels. In addition to capturing the images, we further export the angular labels of the synthetic dataset $\Lambda^s = (\phi_L,\theta_L)$.

4.1.2 Real Dataset

The images in the real dataset are photographed with a Canon EOS 5D Mark II under controlled light condi-

[2] Due to the great effort required to generate real data, we only have 861 images available thus far, 800 for training and 61 for testing. However, we plan to increase our dataset in the future.

[3] 1° instead of 0° on the first step for visibility reasons, the next elevation step is 30°

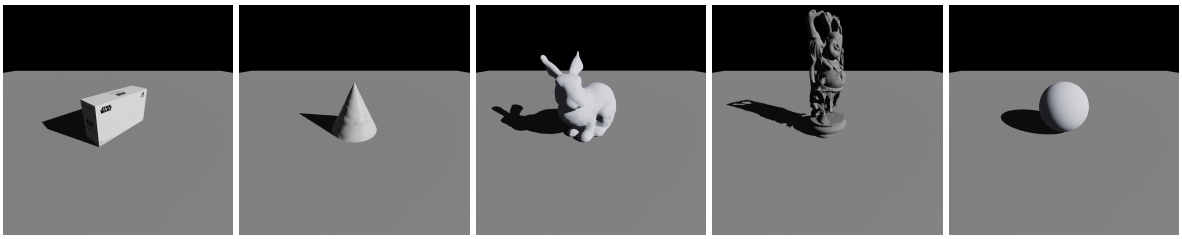


Figure 3: Examples of three-dimensional models used.

tions. The laboratory is completely darkened so that no light source other than the used halogen spotlight would interfere. A box, a three-dimensional print of the Stanford bunny and a ping-pong ball (Fig. 4) as real-world representations of their virtual counterparts are placed on a table with a surface material similar to the base surface material in our synthetic image dataset.

Camera C is placed on a height-adjustable tripod focussing on the centre object O, which is photographed from three different heights, representing elevation angles in the range $[30°, 65°]$. To adjust the view angle of C, O is rotated in the centre of the table, and photographs are taken from seven different angles. To vary the light, the height-adjustable tripod with an attached spotlight L is moved to five different locations[4] around the table, and photographs of the scene illuminated from four different heights are taken.

This way, by the end of the available time frame, a labelled real image dataset is obtained, containing a total of 861 images (i.e. 403 images of the box, 402 images of the Stanford bunny and 56 images of the sphere). To label the images, the vertical and horizontal distances of the scene are measured before taking the photographs, and the angular labels $\Lambda_{\text{real}}(\phi,\theta)$ are computed using the measured distances. For the computation, the measured distances between the following measuring points are required: O and the base of the camera tripod C_b, O and C, C and C_b, a freely chosen reference point R and C_b, O and the base of the light tripod L_b, O and L, L and L_b, and R and L_b (Fig. 5).

With the measured distances, the spherical coordinates of $C = (\Phi_C, \Theta_C)$ and $L = (\Phi_L, \Theta_L)$ can be computed using the law of cosines as

$$f_{\sphericalangle}(a,b,c) = \arccos\left(\frac{a^2+b^2-c^2}{2\cdot a\cdot b}\right) \tag{8}$$

The camera and light azimuth angles Φ_C and Φ_L are computed using $\Phi_C = f_{\sphericalangle}(\overline{OC_b}, \overline{OR}, \overline{RC_b})$ and $\Phi_L = f_{\sphericalangle}(\overline{OL_b}, \overline{OR}, \overline{RL_b})$, respectively. The elevation angle of the camera Θ_C and of the light Θ_L are computed similarly with $\Theta_C = f_{\sphericalangle}(\overline{OC_b}, \overline{OC}, \overline{CC_b})$ and $\Theta_L = f_{\sphericalangle}(\overline{OL_b}, \overline{OL}, \overline{LL_b})$. Both $C = (\Phi_C, \Theta_C)$ and $L = (\Phi_L, \Theta_L)$ are then used to compute the angular labels of the real images $\Lambda^r = (\phi_{lbl}, \theta_{lbl})$ with

$$\phi_{lbl} = \begin{cases} |\Phi_L - \Phi_C| & \text{if } \Phi_L \geq \Phi_C \\ |360 - |\Phi_L - \Phi_C|| & \text{else} \end{cases} \tag{9}$$

$$\theta_{lbl} = \Theta_C + \Theta_L \tag{10}$$

The photographs are taken as RGB images in a resolution of $2,784 \times 1,856$ pixels, cropped to $1,856 \times 1,856$ pixels with the object centred to keep the aspect ratio and then resized to 224×224 pixels using the default Keras ImageDateGenerator function before being used for training.

5 RESULTS

$\text{Net}_{\phi,\theta}$ and Net_{s_x,s_y} were tested by estimating the light direction on the synthetic and real test datasets. To compare the results of the networks (Table 2), the mean angular estimation error was computed as

$$\overline{E}_{\measuredangle} = \frac{1}{n}\sum^{n} \left|\arccos\left({v_p}^T v_t\right)\right| \tag{11}$$

over all test images between the predicted direction v_p and the ground truth direction v_t. Both v_p and v_t were given in spherical coordinates and therefore needed to be converted to Cartesian coordinates.

Train – Test	$\text{Net}_{\phi,\theta}$	Net_{s_x,s_y}
synth – synth	7.8°	3.7°
synth – real	99.2°	25.5°
real – real	12.4°	8.8°
mixed – real	16.8°	7.1°

Table 2: Average angular error $\overline{E}_{\measuredangle}$ on synthetic and real test data.

Furthermore, each network was trained with three different training datasets: a purely synthetic training dataset (Section 4.1.1), an entirely real training dataset (Section 4.1.2), and a mixed training dataset consisting of images from both synthetic and real image datasets.

To determine whether it is theoretically possible to estimate the light direction from images given only RGB information, the synthetically trained $\text{Net}_{\phi,\theta}$ was tested on synthetic test data images, as this test dataset does

[4] Due to space limitations it was not possible to illuminate the scene from $[0°, 360°]$; hence, the scene was illuminated from directions in the range $[\approx 40°, \approx 270°]$.

Figure 4: Examples of real image dataset displaying the real-world models – a box, a Stanford bunny and a sphere.

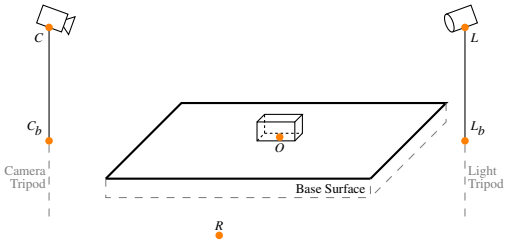


Figure 5: Illustration of measured points in a real scene.

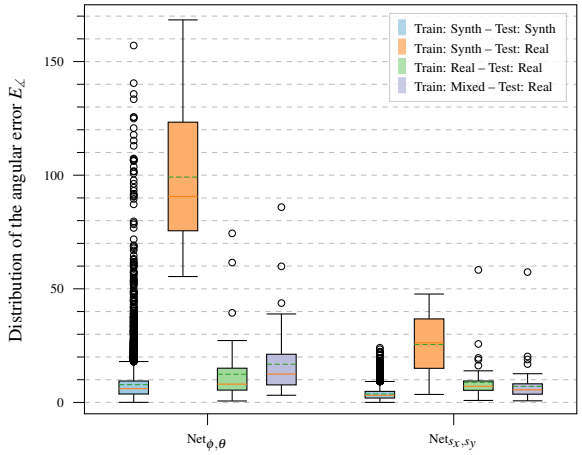


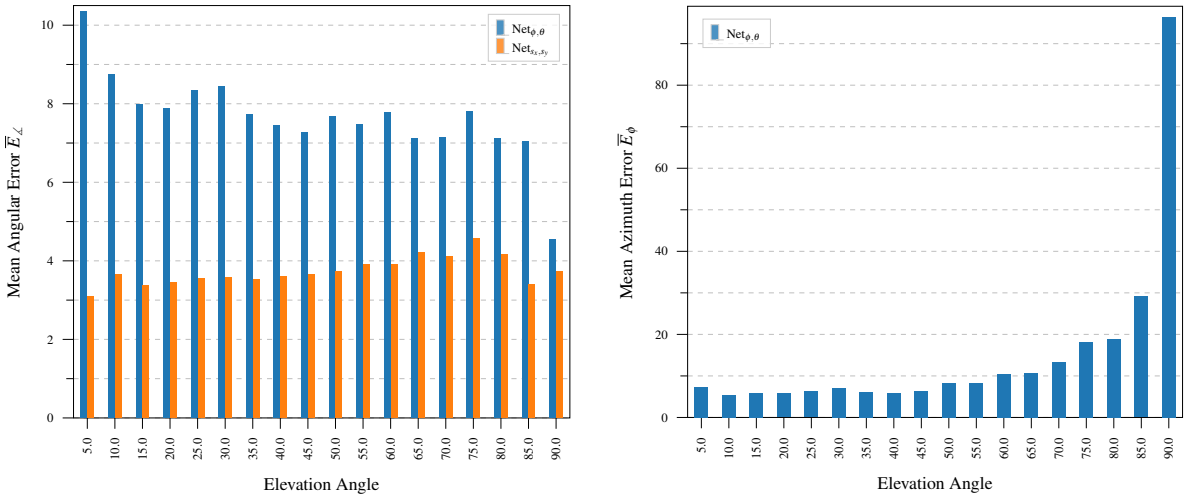
Figure 6: Box-and-whisker diagram of the angular estimation error distribution of $\text{Net}_{\phi,\theta}$ and Net_{s_x,s_y}. The median of the distribution is displayed as a green dashed line, the mean as red line, and outliers as circular marks.

not suffer from noise interference, leading to distorted results. In this test, $\text{Net}_{\phi,\theta}$ achieved a mean angular error of $\overline{E}_{\measuredangle\phi,\theta} = 7.8°$. This result indicates that reconstructing the light direction from only RGB images is reasonable; however, it requires further refinement. The synthetically trained Net_{s_x,s_y} tested on the synthetic test dataset had the best performance, achieving an error of $\overline{E}_{\measuredangle s_x,s_y} = 3.7°$. Considering the angular error distribution of the synthetically trained networks on synthetic test data (Fig. 6, blue graphs), introducing the improvements of Net_{s_x,s_y} not only reduced the estimation error, but also decreased the range of the outlier deviation.

To evaluate the difficulties in predicting the light direction in spherical coordinates (ϕ,θ), the estimation errors of the networks depending on the elevation angle θ (Fig. 7) were investigated. While $\overline{E}_\measuredangle$ of Net_{s_x,s_y} remained fairly constant over all angles of θ, the errors of $\text{Net}_{\phi,\theta}$ did not reveal any clear trend (Fig. 7a). Therefore, the mean error of the estimated azimuth angle $\overline{E}_\phi$ of $\text{Net}_{\phi,\theta}$ (Fig. 7b) was investigated, revealing an increasing error on steep elevations θ, as initially expected. Considering the scale, the estimation of the light direction in stereographic coordinates (s_x,s_y), performed by Net_{s_x,s_y} only revealed a minor deviation (Fig. 7a, green bars). This observed behaviour supports the initial concept of introducing a stereographic representation, as the results indicate that a network using (s_x,s_y) can avoid this inherent error. Because only a small real test dataset could be provided that contained no examples of steeply illuminated objects, this particular case was investigated using only the synthetic test dataset to obtain statistically useful information.

The synthetically trained networks were tested on real image data to investigate whether they would be effective for real data without any necessary adjustment. However, the domain gap between the synthetic and real datasets appears to be large, as $\text{Net}_{\phi,\theta}$ failed completely with a mean angular error of $\overline{E}_{\measuredangle\phi,\theta} = 99.2°$. With an error of $\overline{E}_{\measuredangle s_x,s_y} = 25.5°$ (Fig. 6, orange graphs), Net_{s_x,s_y} achieved reasonable estimation performance. To determine whether the small real image training dataset would be sufficient to obtain reasonable estimation performance, the networks, which were trained with real image data, were tested on the real image dataset. In this test run, $\text{Net}_{\phi,\theta}$ improved its performance with a mean angular error of $\overline{E}_{\measuredangle\phi,\theta} = 12.4°$; Net_{s_x,s_y} outperformed the other network with $\overline{E}_{\measuredangle s_x,s_y} = 8.8°$ (Fig. 6, green graphs). Trained with mixed image data, the networks were tested on real test data to investigate how the small fraction of real image data would benefit from being augmented with the synthetic training set. With a fraction of 0.8% (i.e. a total of 800 real images), the networks achieved a mean angular error of $\overline{E}_{\measuredangle\phi,\theta} = 16.8°$ and $\overline{E}_{\measuredangle s_x,s_y} = 7.1°$ (Fig. 6, purple graphs). Unlike $\text{Net}_{\phi,\theta}$, Net_{s_x,s_y} improved its prediction results by being trained on a mixed dataset as opposed to being trained on a purely real, but small dataset. Augmenting the real dataset with synthetic images appears to worsen the performance in this case, since the synthetically trained $\text{Net}_{\phi,\theta}$ tested on real images performed poorly whereas Net_{s_x,s_y} performed reasonable.

As images of different models were used in the training process, we investigated whether the networks learned



(a) mean angular error $\overline{E}_{\measuredangle}$.

(b) mean azimuth error $\overline{E}_{\phi}$.

Figure 7: Estimation errors of Net$_{\phi,\theta}$ and Net$_{s_x,s_y}$ depending on elevation θ.

a model preference when estimating the light direction. Hence, the mean angular error $\overline{E}_{\measuredangle}$ for each model (Fig. 8) was computed and examined for any significant deviation. Despite having different error amplitudes from each other, the synthetically trained and tested networks only display minor deviations on the model-dependent $\overline{E}_{\measuredangle}$ (Fig. 8a). Trained with mixed image data and tested on the real dataset (Fig. 8b), Net$_{\phi,\theta}$ and Net$_{s_x,s_y}$ display a similar behaviour of the model-dependent $\overline{E}_{\measuredangle}$. Thus, the networks do not appear to favour a certain centre model when estimating the light direction.

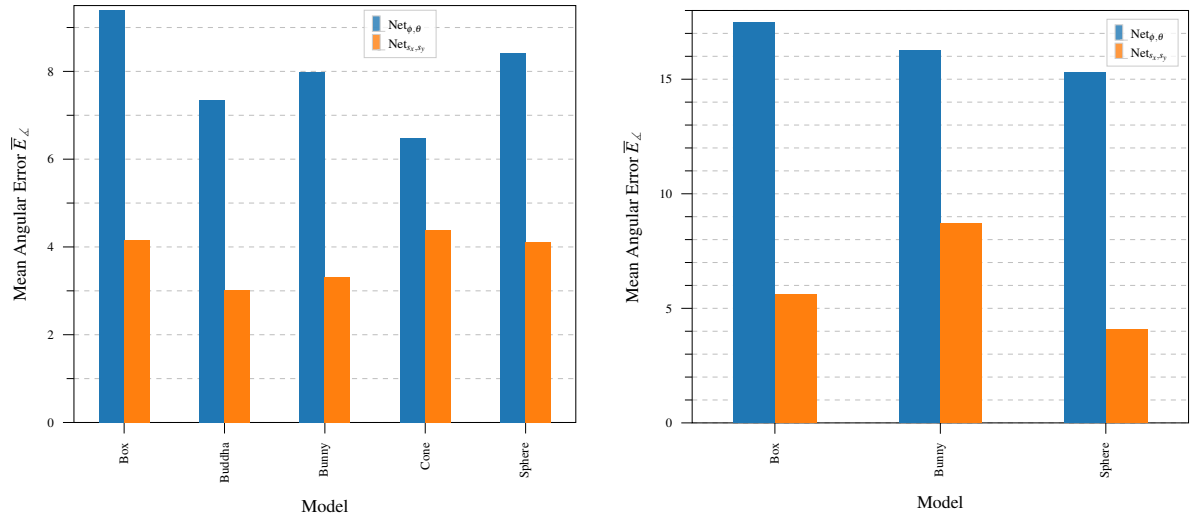
6 DISCUSSION

Considering the performance of the synthetically trained networks on synthetic test data, the results demonstrate that it is possible to reconstruct the dominant light direction (ϕ,θ) in a scene from RGB input images using a VGG-16-like CNN, without requiring additional depth information or relying on special RGB-D images. The light direction reconstruction performance could be further improved by introducing the estimation of stereographic coordinates (s_x,s_y) with Net$_{s_x,s_y}$.

On real test data, the lower prediction accuracy of Net$_{\phi,\theta}$ and Net$_{s_x,s_y}$ are likely to be caused by a domain gap between the synthetic training dataset and the real test dataset, as the prediction performance of the synthetically trained networks on real test data significantly improved by adding even a small fraction of real image data to the training dataset. Considering this domain gap, the prediction performance of Net$_{s_x,s_y}$ on real test data is satisfying, even when synthetically trained.

The small number of real images for testing the networks is a problem, because most of the real images were required for training to improve the prediction performance on real test data, leaving only 61 images available for testing our CNNs, which is critical for obtain statistically useful information. However, the real dataset is sufficient to demonstrate the general feasibility of our proposed approach. Generating the real dataset was a cumbersome task, because to label the images, it was necessary to measure all required distances for each photograph and then prepare the scene for the next scene image, which was time-consuming. Labelling the photos afterwards was not possible, as the labels would have been mere estimates lacking the accuracy necessary for training.

Because we were using a halogen spotlight, which is a spotlight source with small extent in a finite distance in terms of the rendering equation, to illuminate the real scene when taking photographs, there was a structural illumination difference between the directional light used in the synthetic dataset and the extended spotlight in a finite distance in the real dataset. As a directional light source represents a light source, infinitely far away, it is difficult to recreate such a light source in real scenes, as the light source in the scene can not be placed infinitely far away. With increasing distance from an object, however, light from a real source gradually becomes more parallel, transforming into an infinite light source. Considering the distance of the spotlight, which varied between 2 and 3 m, the size of the real models and the input resolution of the CNNs, we considered the difference between the synthetic directional light and the real light source to be negligible. Furthermore, the directional light source in the synthetic dataset did not have a spatial extent, resulting in an umbra without a penumbra, which we attempted to address in our real dataset by using a very small halo-



(a) trained and tested on synthetic image data.

(b) trained with mixed and tested with real image data.

Figure 8: Mean angular errors of $\overline{E}_\angle$ of $\text{Net}_{\phi,\theta}$ and Net_{s_x,s_y} depending on the depicted model.

gen spotlight that was powerful enough to illuminate the scene.

Summarising our contribution to advance the state of the art decribed in Section 2, our findings demonstrate that RGB images are sufficient to estimate the direction of the dominant light source, and that RGB-D images are not required, although they may improve results, which we will investigate in future work. Furthermore, we demonstrated that a stereographic representation of the light direction avoids the coordinate uncertainty of the azimuth angle ϕ at steep elevation angles θ, leading to significantly improved estimation results.

7 FUTURE WORK

We plan to considerably extend our small real dataset to achieve statistically stronger results on real test data. In addition, this extended real dataset can be used to test the proposed approach and additional approaches for different real scene setups. To reduce the effort required to extend the dataset, the generation and labelling process can be automated by a robotic device.

As gathering real image datasets even with an automated device may still require significant effort, we will also pursue the concept of using a semi-real image dataset. We will create such image data by augmenting distinct real scene photographs with synthetic billboards displaying green-screen extractions from images of real objects.

Our long-term goal is illumination reconstruction with DNNs trained on synthetic image datasets that are aided by as little real image data as possible to close the gap between virtual and real scene illumination. Not requiring many real image samples is crucial, as gathering labelled real image data is cumbersome and sometimes not possible. Hence, in the future, we will focus on gaining a better understanding of which details in synthetic training images are important to improve the results achieved by mixed training data. We will first investigate the influence of specific illumination components on the estimation performance, including the surface shading, appearance and presence of a shadow, and indirect lighting. Finally, we will investigate, which DNN architecture and input data in addition to the RGB information can further improve the estimation results.

ACKNOWLEDGEMENTS

We would like to thank the Competence Center Image Processing of the University of Applied Sciences in Munich[5] with all their funding companies[6], as well as the Chair of Computer Graphics and Visualization[7] of the Technical University Munich for allowing us to conduct this research.

We would further like to thank IBM for their support and help. They kindly granted us access to their OpenPOWER[8] deep learning system, which we used for training and optimising our network and other experiments.

Finally, we would like to thank Tobias Kroiss for his assistance in creating the proof of concept, and Bigyan Karki for his effort in creating the real image dataset during his internship here in Munich.

[5] https://www.hm.edu/allgemein/forschung_entwicklung/forschungsfelder/competence_center/bildverarbeitung/index.de.html

[6] https://www.hm.edu/allgemein/forschung_entwicklung/forschungsfelder/competence_center/bildverarbeitung/partner.de.html

[7] https://www.in.tum.de/cg/

[8] https://openpower.ucc.in.tum.de

REFERENCES

Gardner, M.-A., Sunkavalli, K., Yumer, E., Shen, X., Gambaretto, E., Gagné, C., and Lalonde, J.-F. (2017). Learning to predict indoor illumination from a single image. *ACM Trans. Graph.*, 36(6):176:1–176:14.

Garon, M., Sunkavalli, K., Hadap, S., Carr, N., and Lalonde, J.-F. (2019). Fast spatially-varying indoor lighting estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kajiya, J. T. (1986). The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150.

Kán, P. and Kaufmann, H. (2019). Deeplight: light source estimation for augmented reality using deep learning. *The Visual Computer*, 35(6):873–883.

Karis, B. and Epic Games (2013). Real shading in unreal engine 4. In *SIGGRAPH 2013 Course Notes*.

Laskowski, M. (2007). Detection of light sources in digital photographs. *Central European Seminar on Computer Graphics (CESCG)*.

LeGendre, C., Ma, W., Fyffe, G., Flynn, J., Charbonnel, L., Busch, J., and Debevec, P. E. (2019). Deeplight: Learning illumination for unconstrained mobile mixed reality. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, abs/1904.01175.

Liu, D., Long, C., Zhang, H., Yu, H., Dong, X., and Xiao, C. (2020). ARShadowGAN: Shadow generative adversarial network for augmented reality in single light scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lopez-Moreno, J., Hadap, S., E., R., and Gutierrez, D. (2009). Light source detection in photographs. In *Congreso Espanol de Informatica Grafica (CEIG)*, pages 161–168. Other identifier: 2001321.

Marques, B. A. D., Drumond, R. R., Vasconcelos, C. N., and Clua, E. (2018). Deep light source estimation for mixed reality. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP*, pages 303–311. INSTICC, SciTePress.

Pemasiri, A., Wijebandara, C., Wijayarathna, S., Perera, A., and Gamage, C. (2015). An online lighting model estimation using neural networks for augmented reality in handheld devices. In *2015 Fifteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 4–8.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, volume abs/1409.1556.

Wang, T., Hu, X., Wang, Q., Heng, P.-A., and Fu, C.-W. (2020). Instance shadow detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Comparison of 2D vs. 3D U-Net Organ Segmentation in abdominal 3D CT images

Nico Zettler
Aalen University
Beethovenstr. 1
Burren Campus
Germany, 73430, Aalen,
Baden-Württemberg
nico.zettler@hs-aalen.de

Andre Mastmeyer
Aalen University
Beethovenstr. 1
Burren Campus
Germany, 73430, Aalen,
Baden-Württemberg
andre.mastmeyer@hs-aalen.de

ABSTRACT

A two-step concept for 3D segmentation on 5 abdominal organs inside volumetric CT images is presented. First each relevant organ's volume of interest is extracted as bounding box. The extracted volume acts as input for a second stage, wherein two compared U-Nets with different architectural dimensions re-construct an organ segmentation as label mask. In this work, we focus on comparing 2D U-Nets vs. 3D U-Net counterparts. Our initial results indicate Dice improvements of about 6% at maximum. In this study to our surprise, liver and kidneys for instance were tackled significantly better using the faster and GPU-memory saving 2D U-Nets. For other abdominal key organs, there were no significant differences, but we observe highly significant advantages for the 2D U-Net in terms of GPU computational efforts for all organs under study.

Keywords

Organ bounds, U-Net, Architecture, Abdomen, Segmentation, 3D CT images

1 INTRODUCTION

Manual segmentation of liver, kidneys, spleen and the low-contrast pancreas in axial CT slices is very time consuming for radiologists. The automated just-in-time (JIT) segmentation of 3D patient organ models continues to be an unsolved challenge. Automatic and JIT reconstruction procedures of 3D models are highly relevant for surgery planning and navigation. Another important aspect is the saving of computational power and memory, which differ greatly among various Convolutional Neural Network (CNN) architectures for image segmentation. The 2D U-Net architecture[1] showed promising results in 2015, outperforming conventional models on 2D biomedical segmentation problems and being effective on smaller images. Unlike previously known CNN models, the U-Net concept uses down- and up-sampling steps to resample the condensed feature map to the original size of the input image. By incorporating higher resolution feature information in each upsampling step, semantic segmentation of the input images can be achieved very efficiently by skip connections. The resulting U-shaped architecture was extended to a 3D U-Net edition in 2016 by Cicek et al.[2] by replacing 2D operations with their 3D equivalents. The research focus of this paper targets the question, is the 3D U-Net really better suited for 3D data?

In Big Data studies we expect, parallel segmentation of thousands of 3D volumetric images requires high computational time due to the limited amount of processing nodes and sub-processes for parallel processing. We suppose, 3D U-Nets demand even higher GPU computational and memory overhead for 3D image processing. In this work, the focus lies on proposing solutions with costs as low as possible while keeping or even improving segmentation quality. In our concept, we (1) detect and let U-Nets work in local sub-images for each task (liver, kidneys, spleen, pancreas). We (2) compare two U-Net architectures for quality and GPU-performance carrying out semantic segmentations. (3) Based on a detailed statistical analysis, we recommend the more suitable architectural model to the interested reader. Again, subject to our image data base of 80, could the simpler 2D U-Net perform better than complex and theoretically more powerful 3D U-Nets?

2 RECENT SOLUTIONS

3D segmentation is especially relevant for the emerging field of intervention training and planning using Virtual Reality (VR) techniques. Time-variant 4D VR simulations with breathing simulation are readily available for

training and planning of liver interventions[3, 4, 5]. The JIT and high quality reconstruction of all the necessary 3D models remains a main hindrance, i.e. especially abdominal organs are rarely easy to segment. This is due to varying imaging conditions such as contrast agent administration, structure variations and noise.

To this aim in the literature, 3D U-Nets are very often proposed as mightier[6, 7, 8, 9, 10]. However, there are some studies indicating benefits of 2D U-nets in 3D segmentation tasks such as[11, 12, 13]. Nemoto et al.[11] state that low computational burden 2D U-Nets are effective and on par with 3D U-Nets for semantic lung segmentation excluding the trachea and bronchi.

Chang et al.[14] suggested a hybrid approach of 3D and 2D inputs for the evaluation of hemorrhage on head CT scans, which was later adapted by Ushinsky et al.[13] for application to prostate segmentation in MRI images, showing that 2D U-nets are very effective for 3D data. Christ et al.[15] demonstrated a slice-wise application of 2D U-Nets to liver segmentation in combination with 3D random fields. Similarly Meine et al.[12] proposed liver segmentation methods with the assistance of 3D, 2D and three fused 2D U-Net sectional (axial, coronal, sagittal) results in a 2.5D ensemble approach. They find the 2.5D U-Net ensemble results statistically superior for liver segmentation, especially for images with pathologies.

Other recent approaches aim to combine multiple stacked 2D U-Nets[16] and further improve information flow through semantic connections between different components[17]. This concept has also been used in application areas such as colon polyp segmentation[18] and face recognition[19].

In our 2D U-Net setup for the abdominal organs liver, spleen, kidneys and pancreas, we favor axial section training inside of pre-detected organ-specific VOIs. This is for significant radiation dosage savings by high scan pitch. Yet image resolution in axial CT slices is very high.

New aspects in this work are the organ-specific VOI approach for the organs under study such as liver, spleen, kidneys and pancreas. Concluding, an organ-specific architecture dimensionality recommendation for each of the organs is given.

3 PROPOSED SOLUTIONS

Eighty CT-scans and related label data found in various public sources[1] were considered for training and tests of our U-Nets. Their image information differs not only in quality, noise and field of view. But also, the patient-individual volume appearance varies in the amount of slices (64 to 861), pathological lesions, slice width (1 to 5 mm) and applied contrast agent.

The issue relative to data annotation was mainly the lowly contrasted pancreas organ in the CTs, unavailable with some image sets consisting of congruent intensity and label images from the public sources. We coped with that by four eye reviewed manual segmentation of this occasionally missing structure in the label maps. However, most organ reference segmentation were readily available in the public data sources.

Preprocessing

The orientation of images was changed to Right-Anterior-Inferior (RAI) and zero origin (0.0, 0.0, 0.0). As CNNs are not able to interpret voxel spacings natively, isotropic image resampling with 2.0^3 mm as trade-off for varying image xyz-spacings (xy $\leq$ 2 mm, z $\leq$ 5 mm) was performed.

Our concept is composed of two different machine learning approaches, bounding box detection using random regression forests (RRF) and U-Nets for semantic segmentation. The RRFs can be used to detect organ VOIs in CT data. The U-Nets use the detected VOIs and segment the contained organs.

Volume of interest bounding box detection

The ground truth corresponding organ VOI bounding box (BB) - needed in this work's evaluation - of each of the organs can be created by scanning the reference segmentation maps for labeled voxel coordinate extremes. To create a three-dimensional BB vector for each organ, for each coordinate direction (x,y,z), we iterate in orthogonal slices through the organ's label map and save the extreme limits in a 6D BB vector to serve as ground truth BBs.

Alternatively, VOI-based organ extraction[20, 21, 22] is readily available for organ-specific VOI detection of the organ ensemble (liver, still left kidney, right kidney, spleen, pancreas). This step also solves the FoV problem in the scanner z-direction, as CT scans cover variable body portions. Our currently investigated scheme to learn bounding boxes of VOIs using RRFs is summarized in the following.

In our concept, we use Random Regression Forests (RRF) to determine the location and extent of abdominal organs[22]. As seen in Fig. 1, the RRF training step expects scans and ground truth VOIs as input.

A three-dimensional VOI b_c of an organ c can be described by using a 6D vector $b_c = (b_c^{Left}, b_c^{Right}, b_c^{Anterior}, b_c^{Posterior}, b_c^{Head}, b_c^{Foot})$ with coordinates in mm[22]. We run over all voxels $p = (x_p, y_p, z_p)$, which are within a specified radial distance (r = 5 cm) from the scan medial axis. The distance d between such a voxel and each of the VOI walls,

[1] http://visceral.eu,
http://sliver07.org,
http://competitions.codalab.org/competitions/17094

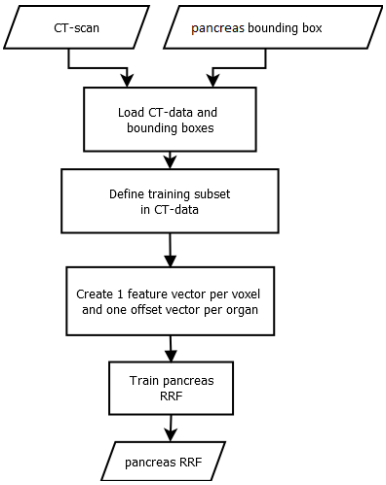


Figure 1: The inputs for the training process are CT scans and ground truth VOIs of a targeted pancreas. We create one feature vector and one mm offset vector for each voxel that is part of a predefined medial cylinder subset in the scan[22]. The trained RRF is able to predict the offset between a voxel and an organ's VOI walls.

can be calculated by using $d(p) = (x_p - x^{Left}, x_p - x^{Right}, y_p - y^{Anterior}, y_p - y^{Posterior}, z_p - z^{Head}, z_p - z^{Foot})$ and is saved as the offset-vector to be learned. In contrast to Criminisi et al.[22], we use only 50 feature boxes, that are evenly distributed on three spheres (r = 5 cm, 2.5 cm, 1.25 cm) to generate the input feature vector. The feature boxes F_j are intended to capture the spatial and intensity context of the current voxel. For this purpose, the mean intensities of the feature boxes are calculated and saved in the feature vector.

Within the first step of our application concept (Fig. 2, left) for every single organ, the RRF locates the VOI for a particular organ as BB.

A detected VOI's information (intensities, labels) is finally resampled to 96x96(x96) as CNN input.

Training and application of 2D and 3D U-Net architectures

Within the 2nd stage, as shown in the bird's eye view in Fig. 2, right, the obtained VOI is either passed to a 2D[1] or a 3D U-Net[2] for semantic labeling.

The training data for our U-Net consists of the expert segmentations and ground truth bounding box contents, as schematized in Figs. 2, 3. The VOIs are then used to locally extract the intensity and label data from the CT scans. As input, a U-Net receives a VOI from the intensity data, while the corresponding label data is connected to the output. We use lightly modified 2D and 3D U-Net architectures vs. Ronneberger et al.[1]. Our architecture consists of four down- and up-scaling steps connected by skip connections and max-pooling while down-sampling. For the 3D U-Net, an additional layer depth dimension is added while all other design elements are kept constant.

The U-Net uses the data contained inside a given VOI bounding box to segment the corresponding organ. The output is a segmentation map of the full target organ in a probability range from 0 to 100%. The threshold for all organs except pancreas was chosen as 50%. For pancreas 30% was empirically found best.

The U-Nets were trained using batches of size 8 over 100 epochs. In addition, Adam optimization and a cross entropy loss function were used. We train one U-Net for each organ, using a ReLU activation function.

Evaluation, metrics and statistics

For fair comparison, in each cross-validation iteration, we use the same 64 training images for both U-Nets, and test with the same 16. This way, we prevent data contamination by separate training and test sets. The difference between the U-Net architectures is input, output and filter kernel layer dimension only. For 2D U-Nets, input and output size is 96x96 and 2D filter kernels are used in the layers. Accordingly for 3D U-Nets, we have 96x96x96 in- and outputs and use 3D convolutions in the architecture layers.

A 5-fold randomized cross-validation using 4:1 splits was used. Five iterations yield 80 quality measures for each organ being used in our statistics (Fig. 1). This means five new models for each organ are trained with randomly selected training data and used in the evaluation. To purely concentrate on the influence associated with U-Net dimensionality, the reference VOIs were utilized in the evaluation of this study. The metric used for analysis is the Dice similarity coefficient:

$$DSC = \frac{2 \cdot |U \cap G|}{|U| + |G|}, \tag{1}$$

where U represents the voxel set from U-Net object segmentation and G the ground truth voxels. A DSC value of 1 indicates perfect segmentation, a value of near 0 poor segmentation. From the DSC results, we calculate means and standard deviations, medians and Inter-Quartile-Ranges (IQR) as measures of accuracy and precision. Statistical assessments using paired T-tests and Wilcoxon-Signed-Rank(WSR)-test were performed with GNU-R 4.0.3. We finally recommend the dimensionality of the U-Net architecture based on accuracy, i.e. greater mean or median, and precision, i.e. smaller standard deviation or IQR and smaller number of outliers. We also keep an eye on the ratio between quality and GPU efforts in time and memory.

4 EXPERIMENTAL RESULTS

Qualitative liver results (Fig. 4) show the 2D U-Net (top) superior with a more evenly distributed segmentation surface coverage. The 3D U-Net (bottom) obviously suffers from under-segmentation as more brown

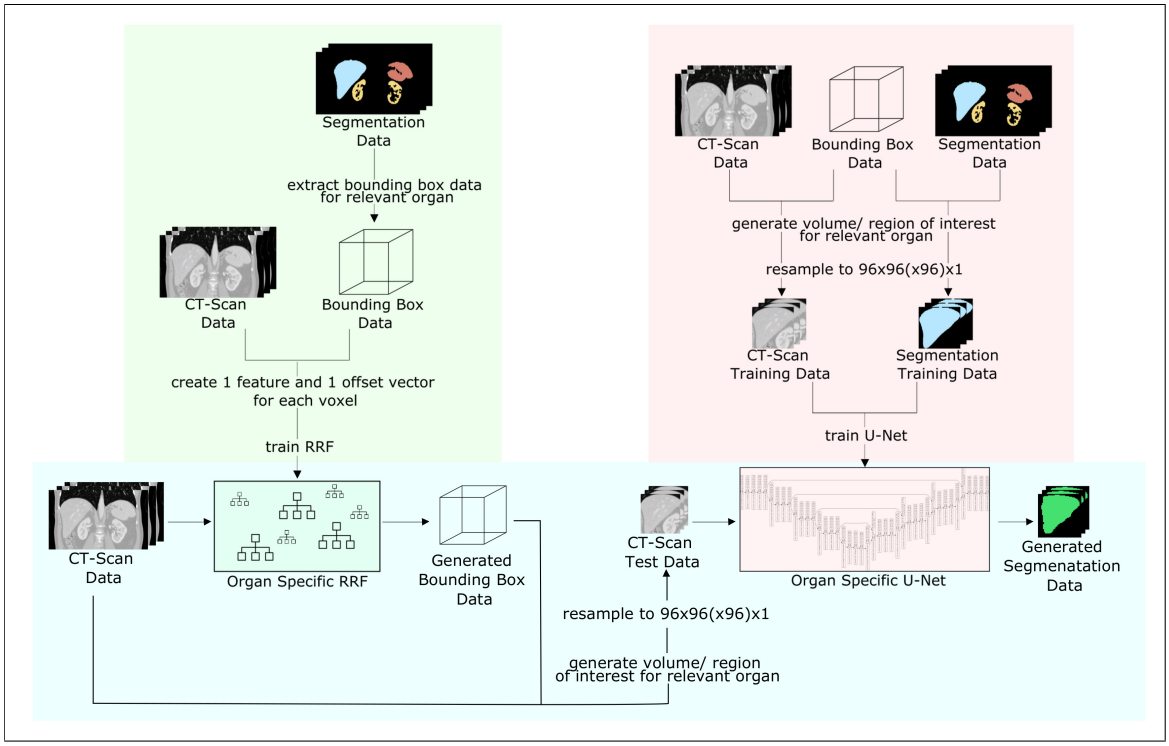


Figure 2: Bird's eye perspective on our current segmentation concept with RRF for VOI detection and subsequent U-Net. Both training (green, red) and application (blue) concepts are shown. In this work, the right part with regards to the U-Nets is focused in the evaluation

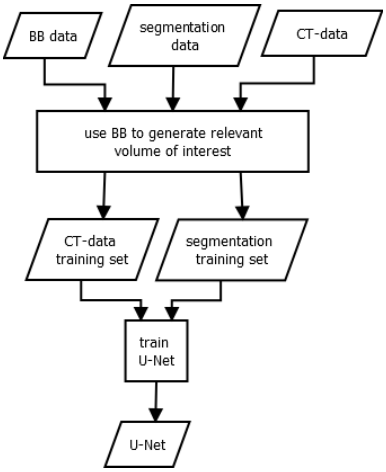


Figure 3: The inputs for the training process are ground truth bounding boxes VOIs, CT scans and their corresponding segmentation maps. The box cuts out the CT- and segmentation data to extract the relevant image region. Inside the organ VOIs, the segmentation is learned. The process results in organ-wise U-Nets

ground truth surface is visible in Fig. 4 (bottom). Referring to Tab. 1, precision is also better for the 2D U-Net due to lower std. deviations and IQRs.

For kidneys, the 2D U-Net is highly significantly better in both T- and WSR-tests. The boxplots with a small x for the mean in Figs. 5, 6 confirm visually: the 2D U-Net is significantly better for liver and highly significantly better for kidneys (Figs. 5(a), 6(a), 5(b), 6(b)).

Spleen segmentation shows an advantage for the 2D U-Net by trend as shown in Figs. 5(c), 6(c)) and Tab. 1.

The 3D U-Net possesses higher accuracy and precision for the low-contrast pancreas. Higher precision is indicated by smaller standard deviations and less outliers in Tab. 1 and Figs. 5(d) and 6(d). With regards to accuracy measured by medians, the 3D U-Net has an edge by trend of 0.02 over the 2D U-Net, while precision measured by IQR is the same.

The GPU-Performance evaluation in Tab. 2 shows the superiority of the 2D U-Net in this study's scale with an 80 images data base. The ratio of quality to GPU resources is always better for the 2D U-Net. In Tab. 2 GPU memory saving is >6-fold in the training stage and >5-fold in the model application, trivially a highly significant result as there are no variations, i.e. no standard deviation and IQR. With respect to GPU calculation times measured in training, the 2D U-Net is precisely 40 seconds or 7% faster on average, again a highly significant mean result. In the U-Net model application, we can observe a weakly significant ($p < 0.1$) advantage for the 2D U-Net of 37% to 75%.

5 CONCLUSION

Surprisingly in this study, 2D U-Nets are favorable regarding the ratio of quality vs. computing costs.

DSCs:	**Mean±Std.**		**Median∼IQR**	
Organ	2D U-Net	3D U-Net	2D U-Net	3D U-Net
Liver	**0.94±0.03***	0.93±0.04	**0.95∼0.02****	0.94∼0.03
R. kidney	**0.91±0.05*****	0.89±0.05	**0.92∼0.03*****	0.90∼0.05
L. kidney	**0.92±0.05*****	0.86±0.14	**0.93∼0.03*****	0.89∼0.08
Spleen	**0.93±0.04**	0.92±0.04	**0.94∼0.03**	0.93∼0.03
Pancreas	0.57±0.19	**0.59±0.15**	0.60∼0.21	**0.61∼0.21**

Table 1: Mean DSCs with standard deviations (Mean±Std.) and Median DSCs with Inter-Quartile-Range (IQR) (Median∼IQR) of 2D and 3D U-Nets from 5-fold randomized cross-validation experiments using 4:1 splits of the 80 images into training and test data.
Legend: We use statistical standard notation for found significances: *; **; ***: $p < 0.05$; $p < 0.01$; $p < 0.001$ from T-tests (paired) and Wilcoxon-Signed-Rank-Tests on the right of the **favorable U-Net result**.

GPU-Performance:	**Memory**				**Time**			
	Training [MiB]		**Application [MiB]**		**Training [min:sec]**		**Application [sec]**	
Organ	2D U-Net	3D U-Net	2D U-Net	3D U-Net	2D U-Net	3D U-Net	2D U-Net	3D U-Net
Liver	1693	10957	1693	9117	9:26	10:07	1.47	3.18
R. kidney	1693	10957	1693	9117	9:28	10:07	0.40	0.55
L. kidney	1693	10957	1693	9117	9:26	10:07	0.40	0.55
Spleen	1693	10957	1693	9117	9:27	10:08	0.40	0.55
Pancreas	1693	10957	1693	9117	9:28	10:08	0.40	0.55
Mean±Std.	**1693±0*****	10957±0	**1693±0*****	9117±0	**9:27±0:01*****	10:07±0:01	**0.61±0.43**	1.07±1.05
Median∼IQR	**1693∼0*****	10957∼0	**1693∼0*****	9117∼0	**9:27∼0:01**$^{+}$	10:07∼0:01	**0.40∼0.003**$^{+}$	0.55∼0.001
Mean Improvement	**647.19%**	N/A	**538.51%**	N/A	**107.13%**	N/A	**175.22%**	N/A
Median Improvement	**647.19%**	N/A	**538.51%**	N/A	**107.07%**	N/A	**137.10%**	N/A

Table 2: GPU-Performance table with memory consumption for training and application to the left and training and application times to the right.
Legend: 1 MiB=1.048581024 MB=1024^2 bytes. We use statistical standard notation for found significances: +; *; **; ***: $p < 0.10$; $p < 0.05$; $p < 0.01$; $p < 0.001$ from T-tests (paired) and Wilcoxon-Signed-Rank-Tests on the right of the **favorable U-Net result**.

The liver and spleen hold the greatest volume in our abdominal organ group. The liver and especially kidney competition is significantly won by the 2D U-Net ($p<0.05$). The liver is a difficult organ often with a variably filled stomach as a neighbor.

Kidneys can be regarded as easy organs lighted by contrast agent and inside fatty tissue with low CT intensity.

A better posed training for the 2D U-Net could be the reason for the 2D U-Nets' better results for liver and kidney tasks. A higher relative number training elements is used, i.e. axial slice pixels, vs. the number of net weights.

The spleen results are in favor of the 2D U-Net regarding the mean and medians by trend. The 2D U-Net is also favorable for a less number of outliers (Fig. 5(c)).

The difficult pancreas does not provide many axial training slices useful for the 2D U-Net, as its elongation is not prominent on the z-axis. We suppose this is the reason, why the 3D U-Net wins here by mean trend in terms of accuracy and precision, i.e. higher mean and lower standard deviation and lower number of outliers (Fig. 5(d)). This win is supported by higher accuracy for the 3D U-Net in terms of medians. However, the race is not decided by significant differences making the 2D U-Net still very attractive for some users with GPU performance and memory concerns.

Abdominal volumetric CT images and key organ segmentation were analyzed. This new study shows interesting results from competing U-Net architectures, especially focusing different dimensionalities of net filter bank kernels and quality vs. GPU performance. The interested reader can now select a particular U-Net architecture, primarily whether to use a computational inexpensive design. Finally in this study's scope, a humble recommendation for the 3D U-Net could be given for the pancreatic organ in terms of better accuracy by trend and smaller standard deviation only. The IQRs as an alternative measure of precision are on par, and regarding the median the 3D U-Net wins just by a small trend. We co-conclude, because of the deeper layering structure and thus more trainable weights, a 3D U-Net needs significantly more training data vs. possible overfitting to outpace 2D U-Nets.

As training volumes are normalized to a square or cube of 96 voxels, the GPU memory consumption in Tab. 2 is always constant. Therefore differences are trivially highly significant, as no varying results occur. We observe consistently lower memory consumption for 2D U-Nets. The memory effort in training is higher for the 3D U-Net including more space for administrative overhead data. Thus, 2D U-Nets can run on affordable 2-4 GB GPUs for 3D CT volume segmentation. 3D U-

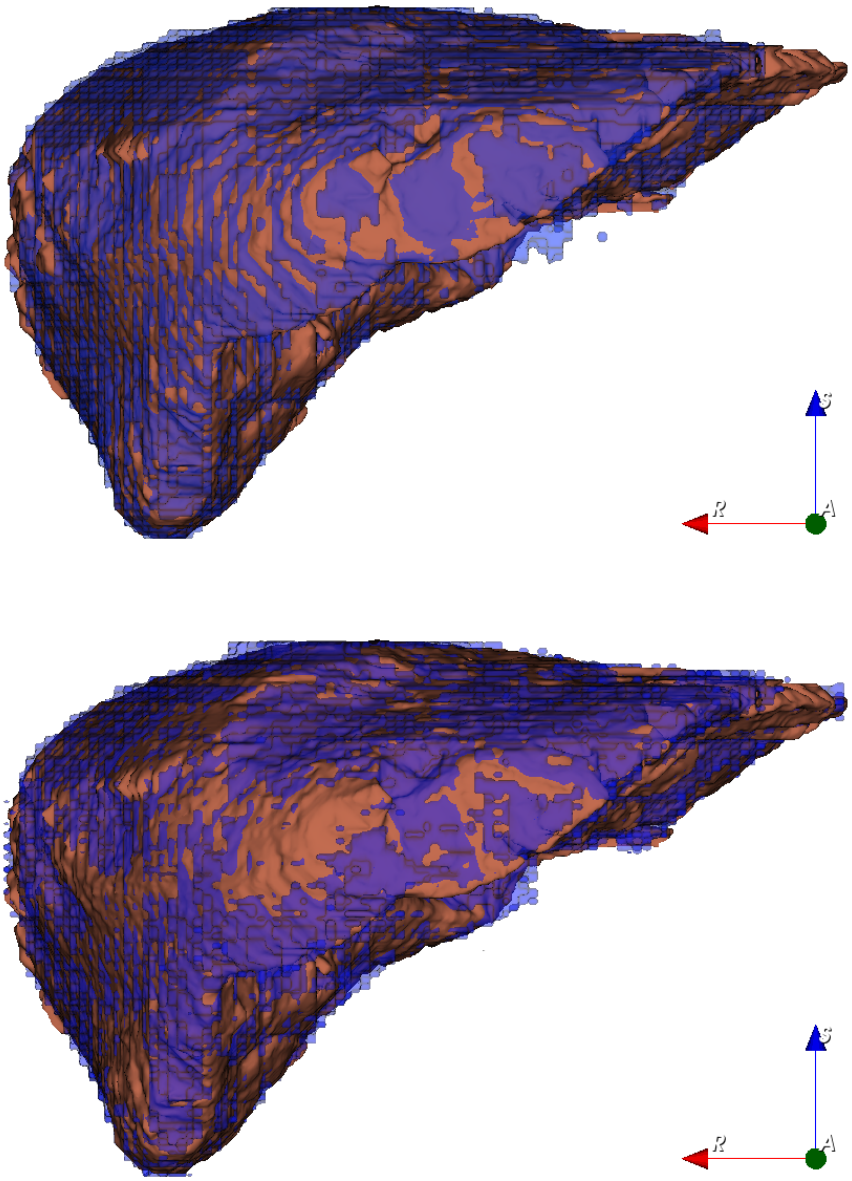
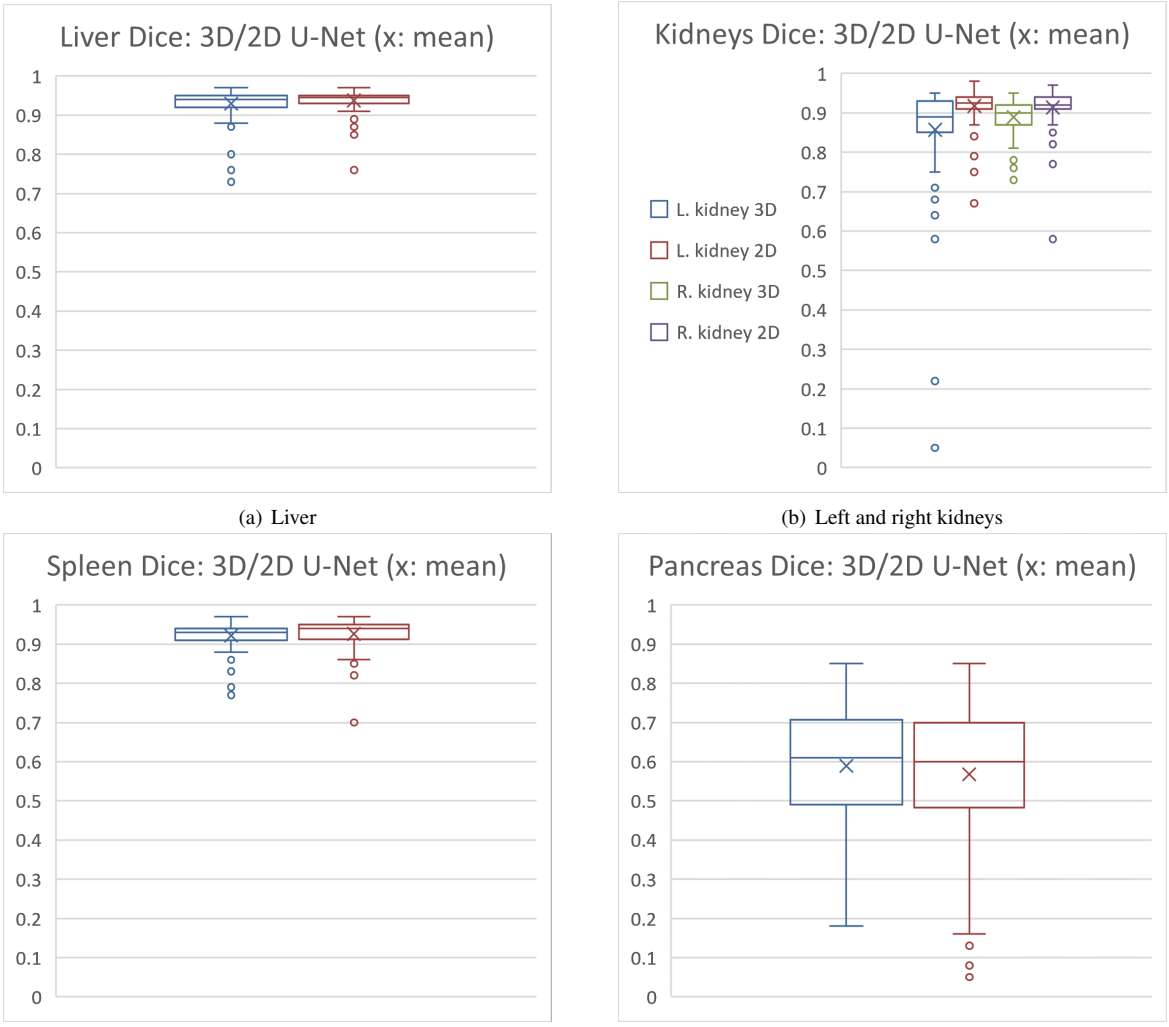


Figure 4: Two liver segmentations (anterior view) with 2D U-Net (top) and 3D U-Net (bottom). The coverage of the 2D U-Net result (top) appears more equally distributed, i.e. more sensitive. Brown: Reference and purple: U-Net CNN. N.B., for the 2D model result on top, the oscillation pattern between reference (brown) and 2D CNN segmentation (purple) is much denser, thus showing out better mean local quality.
Legend: reference (brown) and U-Net segmentation (purple)

Nets definitely need currently totally overpriced 12 GB GPUs.

The timing measures in Tab. 2 clearly speak out for the 2D U-Net. Training is highly significantly 40 seconds or 7% faster on average using the 2D U-Net. However regarding application, in the trained model prediction, the differences are not so striking with a weak significance by median. 37% to 75% improvement can be achieved, however in the range of one second, which is practically unimportant.

As final and bold conclusion regarding our study design and results, we can recommend using the 3D U-Net subject to the amount of data we used here - for pancreas only. The conclusions are justified by statistically significant or by trend quality and GPU-computation performance results for all organs under study. We suppose, 3D U-Nets may overcome in quality when



(a) Liver

(b) Left and right kidneys

(c) Spleen

(d) Pancreas

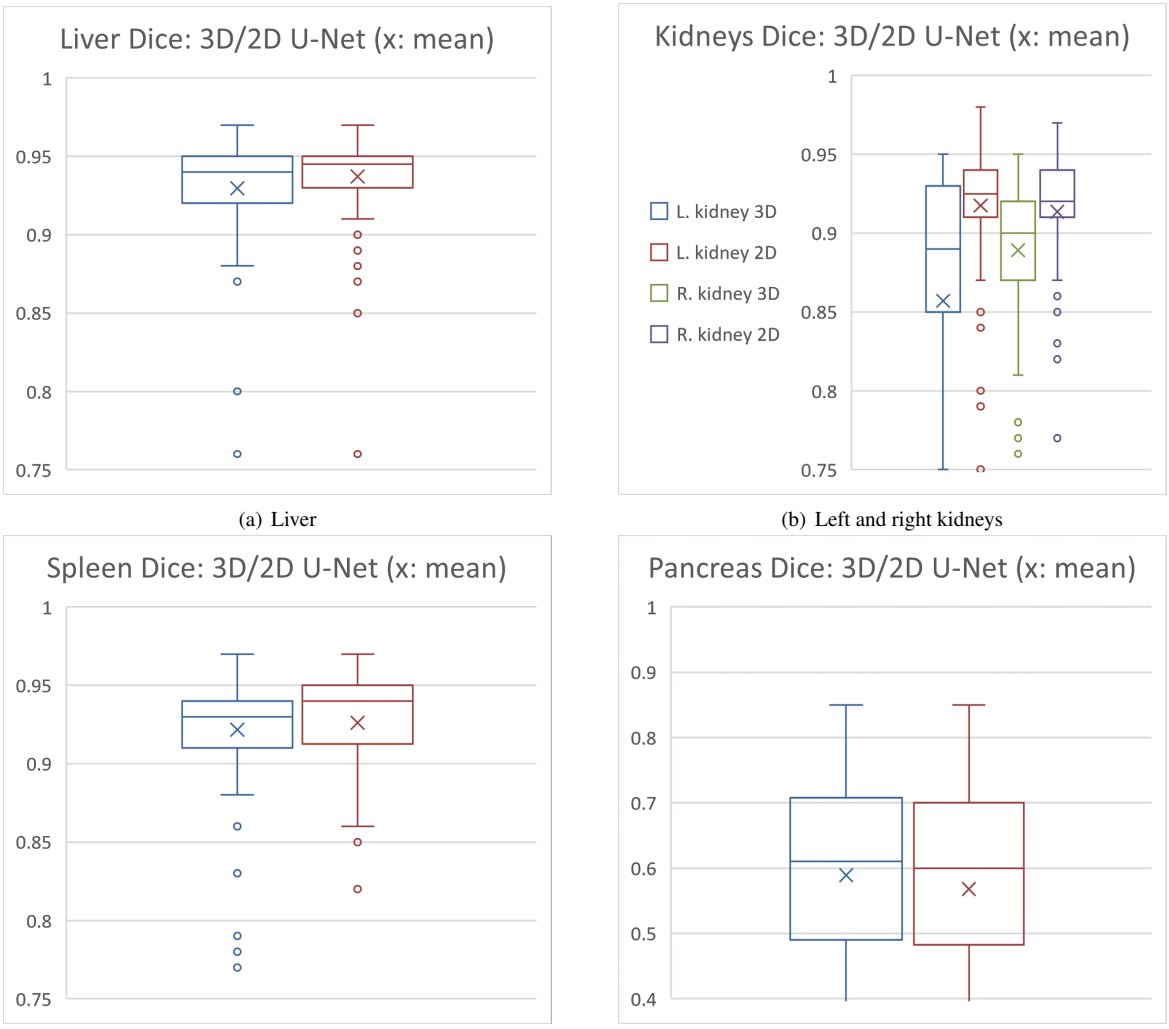
Figure 5: Overview DSC boxplots: 3D (blue, green) and 2D U-Net (brown, purple) on the x-axis vs. DSCs on the y-axis: 2D U-Nets in favor for (a) liver and (b) kidneys (left k. and 3D U-Net: blue, left k. and 2D U-Net: brown; right k. and 3D U-Net: green, right k. and 2D U-Net: purple). (c) Spleen comes out with an edge for the 2D U-Net by trend. (d) Mixed results: 2D U-Net wins the accuracy contest, but loses the precision contest in terms of lower standard deviation (cf. Tab. 1) and regarding less outliers for pancreas.
Legend for (a), (c) and (d): blue boxplots: 3D U-Net; red boxplots: 2D U-Net.

using several hundreds of training images. However, this comes approximately with an order of magnitude higher additional computational burden.

For the first time, an original and significant comparison of U-Net architecture dimension is provided to the reader focusing the key abdominal organs of liver, spleen, kidneys and pancreas. The reader can decide, which approach is appropriate for his concrete target organ, amount of training data and used GPU or cluster nodes, parallel process design, e.g. for atlas-based usage of U-Nets as encountered in multi-classifier fusion[12].

Regarding the difficult pancreas with mixed DSC results in this study, we plan to train 2D U-Nets using a elongation optimized algorithm[23, 24, 25] to provide slices oriented axially along its main central curve, to better reflect its orientation to generate more training slices for 2D U-Nets. On the other hand, more training images shall be used to explore the 3D U-Nets' theoretical advantage under better conditions, as we have discussed, for its training and application.

The scientific explanation of the U-Net methods is given detailed in the original works of Ronneberger and Cicek et al.[2, 1], and we lift these methods here to compare them. The current limitations of this study will manifest as improvements in the future. Development will cover improved bounding box detection. At this state of our research, our current RRF bounding box detection would confound the core message of this



(a) Liver

(b) Left and right kidneys

(c) Spleen

(d) Pancreas

Figure 6: Zoomed DSC boxplots: 3D (blue, green) and 2D U-Net (brown, purple) on the x-axis vs. DSCs on the y-axis: 2D U-Nets in favor for (a) liver and (b) kidneys (left k. and 3D U-Net: blue, left k. and 2D U-Net: brown; right k. and 3D U-Net: green, right k. and 2D U-Net: purple). (c) Spleen comes out with an edge for the 2D U-Net by trend. (d) Mixed results: 2D U-Net wins the accuracy contest, but loses the precision contest in terms of lower standard deviation (cf. Tab. 1) and regarding less outliers for pancreas.
Legend for (a), (c) and (d): blue boxplots: 3D U-Net; red boxplots: 2D U-Net.

paper, the aim of which was to focus purely on U-Net performances.

6 ACKNOWLEDGMENTS

German Research Foundation: DFG MA 6791/1-1;
Nvidia GPU grant;
Foundation Kessler+Co. for Education and Research: EXPLOR19-AM.

REFERENCES

[1] Olaf Ronneberger et al. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: Medical Image Computing and Computer Assisted Intervention - MICCAI 2015, 2015.

[2] Ozgun Cicek et al. “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation”. In: Medical Image Computing and Computer Assisted Intervention - MICCAI 2016, 2016.

[3] Andre Mastmeyer et al. “Population-based respiratory 4D motion atlas construction and its application for VR simulations of liver punctures”. In: *Proc. SPIE Medical Imaging: Image Processing*. Vol. 10574. International Society for Optics and Photonics. 2018, p. 1057417.

[4] Andre Mastmeyer et al. “Interpatient Respiratory Motion Model Transfer for Virtual Reality Simulations of Liver Punctures”. In: *Journal of*

World Society of Computer Graphics - WSCG 25.1 (2017), pp. 1–10.

[5] Andre Mastmeyer et al. “Real-Time Ultrasound Simulation for Training of US-Guided Needle Insertion in Breathing Virtual Patients”. In: *Studies in Health Technology and Informatics*. Vol. 220. IOS Press, 2016, p. 219.

[6] Getao Du et al. “Medical image segmentation based on u-net: A review”. In: *Journal of Imaging Science and Technology* 64.2 (2020), pp. 20508–1.

[7] Nahian Siddique et al. “U-Net and its variants for medical image segmentation: theory and applications”. In: *arXiv preprint arXiv:2011.01118* (2020).

[8] Pavlo Radiuk. “Applying 3D U-Net architecture to the task of multi-organ segmentation in computed tomography”. In: *Applied Computer Systems* 25.1 (2020), pp. 43–50.

[9] Nima Tajbakhsh et al. “Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation”. In: *Medical Image Analysis* 63 (2020), p. 101693.

[10] Geert Litjens et al. “A survey on deep learning in medical image analysis”. In: *Medical image analysis* 42 (2017), pp. 60–88.

[11] Takafumi Nemoto et al. “Efficacy evaluation of 2D, 3D U-Net semantic segmentation and atlas-based segmentation of normal lungs excluding the trachea and main bronchi”. In: *Journal of radiation research* 61.2 (2020), pp. 257–264.

[12] Hans Meine et al. “Comparison of u-net-based convolutional neural networks for liver segmentation in ct”. In: *arXiv:1810.04017* (2018).

[13] Alexander Ushinsky et al. “A 3D-2D Hybrid U-Net Convolutional Neural Network Approach to Prostate Organ Segmentation of Multiparametric MRI”. In: *American Journal of Roentgenology* 216.1 (2021), pp. 111–116.

[14] P.D. Chang et al. “Hybrid 3D/2D Convolutional Neural Network for Hemorrhage Evaluation on Head CT”. In: *American Journal of Neuroradiology* 39 (July 2018). DOI: 10.3174/ajnr.A5742.

[15] Patrick Ferdinand Christ et al. “Automatic Liver and Lesion Segmentation in CT Using Cascaded Fully Convolutional Neural Networks and 3D Conditional Random Fields”. In: *Lecture Notes in Computer Science* (2016), 415â423. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46723-8_48. URL: http://dx.doi.org/10.1007/978-3-319-46723-8_48.

[16] Debesh Jha et al. “DoubleU-Net: A Deep Convolutional Neural Network for Medical Image Segmentation”. In: July 2020, pp. 558–564. DOI: 10.1109/CBMS49503.2020.00111.

[17] Zhiqiang Tang et al. *CU-Net: Coupled U-Nets*. Aug. 2018.

[18] Dinh Viet Sang et al. *AG-CUResNeSt: A Novel Method for Colon Polyp Segmentation*. 2021. arXiv: 2105.00402 [eess.IV].

[19] In Seop Na et al. “Facial UV map completion for pose-invariant face recognition: a novel adversarial approach based on coupled attention residual UNets”. In: *Human-centric Computing and Information Sciences* 10.1 (Nov. 2020). ISSN: 2192-1962. DOI: 10.1186/s13673-020-00250-w. URL: http://dx.doi.org/10.1186/s13673-020-00250-w.

[20] Daria Kern et al. “3D Bounding Box Detection in Volumetric Medical Image Data: A Systematic Literature Review”. In: 2020. arXiv: 2012.05745 [eess.IV].

[21] Andre Mastmeyer et al. “Random forest classification of large volume structures for visuo-haptic rendering in CT images”. In: *Proc. SPIE Medical Imaging: Image Processing*. 2016, 97842H.

[22] Antonio Criminisi et al. *Decision Forests for Computer Vision and Medical Image Analysis*. London: Springer, 2013.

[23] Christian Reinbacher et al. “Variational segmentation of elongated volumetric structures”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 3177–3184.

[24] Maria Kallergi et al. “Automatic Segmentation of Pancreatic Tumors in Computed Tomography”. In: *Handbook of Biomedical Image Analysis*. Springer, 2005, pp. 183–228.

[25] Andre Mastmeyer et al. “Model-based Catheter Segmentation in MRI-Images”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention - MICCAI*. 2015.

An Analysis on Pixel Redundancy Structure in Equirectangular Images

Vazquez, I.
Boise State University
1910 W University Dr.
USA 83725, Boise, ID
ikervazquezlopez@u.boisestate.edu

Cutchin, S.
Boise State University
1910 W University Dr.
USA 83725, Boise, ID
stevencutchin@u.boisestate.edu

ABSTRACT

360° photogrammetry captures the surrounding light from a central point. To process and transmit these types of images over the network to the end user, the most common approach is to project them onto a 2D image using the equirectangular projection to generate a 360° image. However, this projection introduces redundancy into the image, increasing storage and transmission requirements. To address this problem, the standard approach is to use compression algorithms, such as JPEG or PNG, but they do not take full advantage of the visual redundancy produced by the equirectangular projection. In this study of the 360SP dataset (a collection of Google Street View images), we analyze the redundancy in equirectangular images and show how it is structured across the image. Outcomes from our study will support the developing of spherical compression algorithms, improving the immersive experience of Virtual Reality users by reducing loading times and increasing the perceptual image quality.

Keywords

Image compression, Equirectangular projection, Virtual Reality,

1 INTRODUCTION

Virtual Reality (VR) provides real-world scenarios focusing on maximizing the immersive sensation. Images and videos shown in VR devices are known as panoramic, 360° or omnidirectional images. Gaming, art, photography and social media takes advantage of the immersive properties VR provides and has been used in many applications [Gooed, Faced].

360° photogrammetry captures the surrounding light from a central point. To process and transmit this type of image over the network to the end user, the most common approach is to project them onto a 2D image using the equirectangular projection to generate a 360° image. This permits easy image storage and visualization but introduces a major problem. Because the projection reduces the dimensionality from 3D to 2D, it generates topological visual alterations related to the projection stretching samples over multiple pixels and introducing redundant data.

To the best of our knowledge, there is no study about how the redundancy is structured in equirectangular images, nor how much each pixel contribute to the overall image quality. In order to provide a good immersive experience by reducing image loading times and improve the streaming quality, it is important to determine how the redundant data is structured in equirectangular images. With the current trend of resolution increase in visualization devices, images can be adapted accordingly. If we increase the image resolution, the number of redundant pixels will also increase and image file size will be needlessly increased. Redundant pixels do not contribute to the image quality as much as unique pixels, and their impact on the user's immersive experience is often minimal.

In this paper, we present the results of a study we conducted on how perceptual redundancy is structured in equirectangular images. The redundancy is created by how the equirectangular projection stretches the captured samples from the 360° field of view. We determine what is the contribution of each of the projected image pixels to the image's overall perceptual quality. Because of the nature of equirectangular projection the portions of the image at angles farther from the horizon (top and bottom rows respectively) contribute less to the overall image quality than the portions nearest the image horizon. Since equirectangular images capture a 360° field of view, some areas of the image are stretched because the scene is projected creating low

gradient textures (top and bottom rows) while other areas remain unstretched (middle rows). The contribution of this paper is an analysis, using different metrics (row and region based), of how the redundancy is structured in equirectangular images and how each portion in the image contributes differently to the overall image quality.

2 RELATED WORK

360° photogrammetry captures the surrounding light using a camera located at the central point in a scene. There are multiple ways to capture these type of photos, including multi-shot camera rigs, single camera shot and single camera shot with catadioptric lenses. By applying warping and/or stitching techniques to the captures, we can build a 360° photo in which each pixel represents a small portion of the full scene [Sze10]. Conceptually, the photo is a colored spherical point cloud in a 3D space. The density of this point cloud determines the resolution of the capture and therefore the overall quality of the photo. Using higher resolution sensors in cameras results in a direct visual quality increase. However, the memory requirement for 360° photos that provide a good visual quality is high.

For efficient storage and transmission of 360° photos, the photo sphere must be projected from 3D space to 2D to generate a 360° image. This process produces visual alterations in the projected image yielding redundant data. The most common projections are cylindrical, equirectangular and cube-map projections. As *Azevedo et al.* [Aze19] stated, the capturing, processing and delivery of 360° content, introduces visual alterations which do not fully keep perspective projection properties. The study reviewed the most common types of visual alterations, such as geometrical and data alterations, and identified their main causes.

Visual alterations generate redundant data that can be compressed to reduce the storage and transmission requirements because they do not add any improvement to image quality. Despite the good performance of standard codecs for regular images, using them to compress 360° images does not take full advantage of the 360° image's spherical properties. *Sun et al.* [Sun18] conducted a subjective image quality assessment study on how common image codecs influence the 360° image quality. And it has been shown that codecs for 2D images can be adapted to the 360° images by exploiting projection properties [DS16].

Downsampling specific regions in equirectangular images was proposed in some works. Since top and bottom rows of the equirectangular mapping are stretched, Budagavi *et al.* [Bud15] proposed a Gaussian smoothing filtering over those regions. The key point in that work was the exploiting of the characteristics of the projection and give more importance to equatorial regions. Using the same idea, Youvalari *et al.* [You16], split the image in several strips and each one is downsampled differently based on the latitude they are located. Lee *et al.* [Lee17] used a similar approach but in a pixel level, where each image row is individually downsampled and produces a rhomboid shape image. This studies remove most of the redundant data produced by the equirectangular projection, however, they do not fully consider the structure of visual alterations produced in the projected images.

The compression efficiency of the 360° images differs based on the used projection [Jam19, Yu15]. Some studies considered compression approaches by just reprojecting the images and discovered that compression performance is content dependent. Boyce *et al.* [Boy17], detected regions where textures were highly detailed and rotated the panoramas in the spherical domain to relocate those textures close to the equator. In this manner, regions with low gradients will locate in top and bottom rows where the stretching is higher and benefits the compression with standard compression algorithms. Sun *et al.* [Sun18], proposed the addition of deep learning to the process and by designing a Convolutional Neural Network measured the relation between the visual content and compression rate for different rotations.

The most common metrics to measure the effectiveness of the proposed compression approaches are Peak Signal to Noise Ratio (PSNR) and Structural Similarity Metric (SSIM) [Wan04]. These two metrics were designed to approximate the human perception of regular 2D images, however, they may not provide accurate measurements for 360° images. Since the spherical captures are projected onto a 2D plane, redundant pixels emerge and therefore these metrics are biased by the repeated pixels. To address this problem, the WS-PSNR [Yu15] and S-SSIM [Che18] were proposed, where the results of PSNR and SSIM are weighted, giving more weight to elements close to the horizon.

There exists multiple benchmarks for studying 360° images in different areas. Datasets, such as Abreu *et al.* [DA17], Sitzman et al. [Sit18], Gutierrez et al. [Gut18], were presented to study how the users visualize 360° images by tracking their attention, head movement and eye movement. For 360° image visual quality assessment, the most popular dataset are Huang et al. [Hua18], CVIQD [Sun17], CVIQD2018 [Sun18] and OIQA [Dua18]. SUN360 [oTed] and 3D60 [oITed] (which is a mix of the SunCG [Soned] and SceneNet [McCed] datasets) and 360SP [Cha18] are interesting benchmarks too. 360SP contain outdoor images intended to locate the sun position while 3D60 is a synthetic dataset of indoor images with their respective depth maps. Even though these benchmarks were not

intended to analyze the visual alterations of 360° images, they are a good source for analyzing the redundancy structure in equirectangular images.

3 METHODOLOGY

The main idea of this study is to assess how much individual pixels contribute to the overall image quality based on their location in an equirectangular image. Due to the projected visual alteration when projecting a 3D sphere onto a 2D image plane, some of the samples are stretched across multiple pixels and redundant pixels emerge. Those redundant pixels do not contribute as much to the image quality as non-redundant pixels and add extraneous data that is needlessly stored and/or transmitted. Knowing how this data is structured across equirectangular images is key to designing efficient 360° image focused compression algorithms.

In this work, we first analyzed the quality impact of each equirectangular image row. In each row, we remove multiple sets of contiguous pixels and interpolate gaps to demonstrate the ease of their reconstruction. By computing the resulting row quality, this approach shows the redundancy behavior and provides a visual clue to the number of extra pixels in the image. Full redundancy structure, however, is not revealed with a row based approach and so we use region focused approaches for a better revelation of the underlying structure. Instead of computing quality metrics over entire rows, for each pixel in the image, we focus on its neighbouring region and compute its quality after the region interpolation. In both cases, when we compute the quality metric, highly redundant regions yield high quality measures even when pixels are removed while the quality drops in low redundancy regions when pixels are removed.

With these approaches, the study shows some bias under certain image conditions (such as smooth textures being in the same region). To ensure a robust analysis, we discuss this problem and possible solutions in Section 4.

3.1 Row impact

In equirectangular images, portions of the image at angles farther from the horizon have higher pixel redundancy than at the horizon line due to the projection properties. To test how this redundancy is structured along the image rows, we remove sets of pixels in an increasing number, interpolate them and compute a quality metric. This process reveals the behavior of redundancy in equirectangular images.

We extracted each row in the image and processed them individually to compute their contribution to the image quality. As seen in Figure 1, from each row, we removed multiple sets of contiguous pixels and kept some

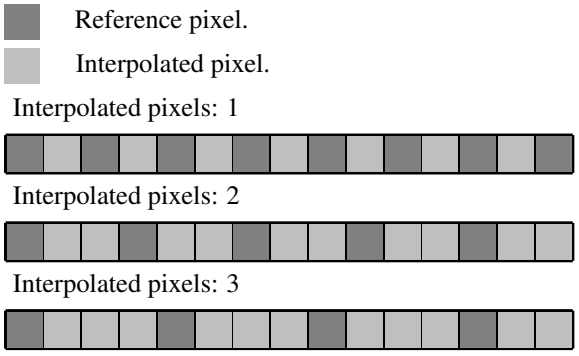


Figure 1: Image row pixel removal and interpolation in an increasing manner.

of the original image pixels as reference. We increase the size of the removed sets in an iterative manner to decrease the number of reference pixels and increase the image error. Then, we reconstruct the missing pixels with the linear interpolation using the reference ones. Since the linear interpolation is a basic interpolation technique and leads to poor results for image reconstruction, it shows how easy it is to reconstruct some regions in equirectangular images. Our thesis is that if, after reconstructing the row with a very basic technique, the resulting row quality is high, the redundancy in that row would be high and therefore most of the pixels in it do not have a high impact on the image quality. Yet, if the resulting quality is low, the redundancy in that row is minimal and most of the pixels in that row would be relevant to the overall equirectangular image quality.

When computing the quality metric, our focus is on the metric behavior and not on its specific value. We wanted to analyze the redundancy behavior in equirectangular images as we move from the top to the bottom rows. Because we processed each row individually, the rest of the image remains untouched.

Computing the row based quality metric over the entire image is unnecessary because it will produce values close to 0 and, based on the image size, the arithmetic precision was seen to affect the results. To address this problem, we computed the row based quality metric only over the modified row and not over the full image. This approach does not have an impact on the study because we are analyzing the error behavior and not the error value. The behavior will be the same even if we compute the row based quality metric of the entire image or just a single row, the only difference will be the resulting error values.

As a result of this process, Algorithm 1, we produce a 2D matrix containing the impact of each row with different removed pixel set sizes. The first dimension of the matrix represents the rows while the second dimension represents the size of removed pixel sets.

Algorithm 1: Row impact on image quality

```
Function Row_Impact(img) : matrix is
    q = matrix[img.rows, MAX_SET_SIZE];
    foreach row ∈ [0..img.rows] do
        foreach ss ∈ [1..MAX_SET_SIZE] do
            r = remove_pixels(img[row], ss);
            r' = interpolate_pixels(r, ss);
            q[row, ss] = quality(img[row], r');
        end
    end
    return q
end
```

3.1.1 Metrics

The quality measurement of the reconstructed rows can be computed using different metrics. In this work, for standard image comparison, we use PSNR and SSIM to compare the equirectangular images. However, these metrics do not take into account the properties of 360° images and therefore, the visual from projecting the 3D sphere onto a 2D plane may affect the results.

$$w(r) = cos\frac{\left(r+0.5-\frac{N}{2}\right)\pi}{N} \tag{1}$$

$$WMSE(r) = \frac{\sum_{i=0}^{cols}(x_i-y_i)^2 \cdot w(r)}{\sum_{j=0}^{rows} w(j)} \tag{2}$$

$$WPSNR(r) = 10log\left(\frac{MAX_I^2}{WMSE(r)}\right) \tag{3}$$

where:

r = image row.
N = image height.
x = original image.
y = reconstructed image.
MAX_I = maximum image intensity value.

In addition to the standard metrics, we use their weighted versions: W-PSNR and S-SSIM. These two metrics weigh the resulting quality using a row based cosine weight, they lower values from top and bottom rows while they rise values in the middle rows. We slightly modified them to only compute the values from single rows instead of over the full image (Equation 3 for $WPSNR_{row}$ and Equation 5 for S_SSIM_{row}).

$$SSIM(r) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \tag{4}$$

where:

μ = image row mean.
σ_{xy} = x and y image row covariance.
σ^2 = image row variance.
$c_1 = (0.01 \cdot MAX_I)^2$
$c_2 = (0.03 \cdot MAX_I)^2$

$$S\_SSIM(r) = \frac{\sum_{i=0}^{cols} SSIM(r) \cdot w(r)}{\sum_{j=0}^{rows} w(j)} \tag{5}$$

3.2 Pixel impact

Studying how the rows impact the equirectangular image quality provides an overall idea about how pixels are relevant in the projected image. A limitation to this approach is that it does not provide information about pixel-wise impact or how the redundancy is structured throughout equirectangular images. Because the equirectangular projection maps the meridians to straight lines the rows are stretched and redundancy emerges perpendicular to the meridians. In order to conduct a study about how each pixel impacts the image quality, we use the ROAD [Gar05] metric and localized SSIM.

In the pixel impact analysis, we do not use the PSNR metric because it is resolution dependant. PSNR requires the computation of the Mean Squared Error, and in small regions, a small difference highly impacted the resulting perceptual quality, therefore, the variance of PSNR results over multiple image is too large to be useful when averaging them.

3.2.1 ROAD metric

The ROAD metric is defined for each pixel (x,y) as Equation 6, where r is the sorted difference of a pixel respect it's neighborhood ones in an ascending order, and $2 \leq m \leq 7$. This metric was presented as a measurement to detect impulse errors when transmitting images over the network. It is a metric that measures the gradient of a pixel respect to its most similar neighborhood ones in a single image. Since the projection generates visual effects in a similar manner on all equirectangular images and stretches pixels across the rows, this metric provides insights about regions with low gradients. Therefore, we only computed the ROAD metric over the original equirectangular image and not over the interpolated image.

$$ROAD_m(x,y) = \sum_{i=0}^{m} r_i(x,y) \tag{6}$$

There are two possible reasons for the low gradient regions: smooth textures and projection's visual alterations. With the ROAD metric, it is not possible to discern, within a single image, whether or not the low gradient regions belong to smooth textures or to visual alterations. However, by computing the ROAD metric

over a number of images and averaging the results, we introduce variability into the equation and therefore we can eliminate smooth texture effects and only the projection's visual alterations remain.

3.2.2 Localized SSIM

In the standard approach, when computing the SSIM metric, the entire image is used to generate one single value. This value represents the quality of a reconstructed image with respect to the original one. Because in our case we want to measure the impact of a pixel in its local region after reconstructing the image, we created a modified SSIM to only be computed in local regions.

For each pixel in the image, we cropped the 3×3 neighborhood from both images (original and reconstructed) and treated them as full images to compute the SSIM. This operation reveals how much that region impacts in the overall image quality. When generating the results over the entire image, we reveal the redundancy structure in the image, as explained in Section 5.

4 EVALUATION

One of the main problems we found for this work was the configuration of the captured 360° photos. Usually, for better visualization, the horizon line of the scene is mapped to the equirectangular image's middle rows. This orientation provides a good overall idea about how the image will look in a 360° viewer when the content is projected onto a sphere. Because of that, the sky and ground tend to be located in the top and bottom rows. Those regions tend to be smooth textures with very little change and therefore, the redundancy structure analysis would be biased, resulting in highly redundant regions when analyzing them. In order to address this, as stated in Section 3.2, we introduce variability by vertically rotating (pitch) the 360° images in the dataset using 22 degree intervals. To compute the vertical rotation, we transform the equirectangular image to the 3D sphere, rotate it in the x axis and project it back to the 2D image using the equirectangular projection. This operation moves all regions in the original images to different locations in the image, which ensures that smooth regions will be processed at different locations, ensuring a robust redundancy analysis.

In our analysis, variability is important and therefore, we require a large non-synthetic 360° image dataset. Synthetic datasets tend to contain large amounts of smooth textures which will bias our results by not providing enough variability. The 360SP dataset is the perfect fit for our needs. It is composed of ≈15000 outdoor images collected from Google Street view at 3328×1664 resolution. After our rotations, the number of images we generated from the dataset was ≈ 170000, which makes a large enough dataset that provides the needed variability to our analysis.

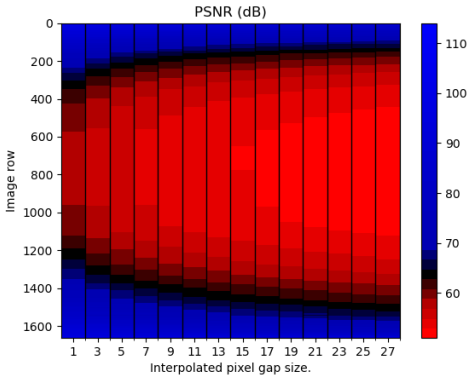


Figure 2: PSNR analysis results.

5 RESULTS

In this section, we show and discuss the results of computing the metrics presented in Section 4. It is important to keep in mind that our results were not intended to compare any methods specifically for their effectiveness. Our goal with this analysis was to study the metric behavior and not the actual values. We ignored the results in terms of their value and focused on how each metric behaved across the equirectangular image. By studying how the metric changed for different pixel gap sizes in each row and in each pixel's neighboring region, we revealed the redundancy structure in equirectangular images.

In Figure 2, we can see that the perceptual impact is minimal in top and bottom rows. It is interesting to point out that PSNR does not seem to care that much about the structure of the equirectangular image, it is sensitive to noise in highly redundant regions. As it is shown in Figure 2 at image rows $[200, 400]$, a small change in a smooth region highly impacts the image's PSNR.

SSIM metric shows more robust results than PSNR for highly redundant regions. In Figure 3, we can see what is the impact in the images for the SSIM metric. It is worth noticing that, for the top and bottom rows of the equirectangular image, the number of pixels that can be removed with minimal information loss is significant. It is possible to remove very large gaps of pixels in those regions and reconstruct them with minimal quality loss.

We colored both plots in Figure 2 and 3 to reveal how the metrics transition from high redundancy to low redundancy regions. To do so, we used the histogram of each metric's data (Figure 4a and 4b) to decide the range of the transition data. In the PSNR case we selected 65dB to be black color to differentiate the two regions, but in the SSIM case, there is no clear boundary and the transition is smoother.

The results from the WPSNR metric (Figure 5) shows a similar behavior as the PSNR with a similar histogram (Figure 7a). The S-SSIM, however, shows a completely

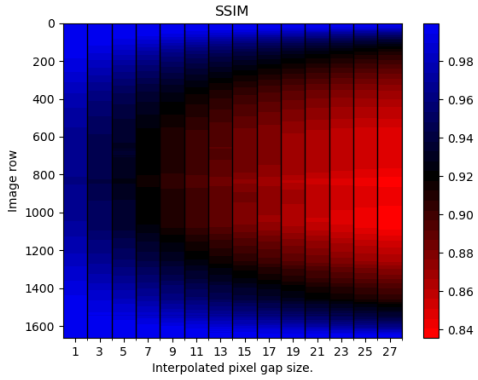
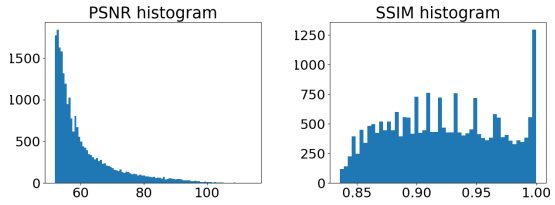


Figure 3: SSIM results.



(a) (b)

Figure 4: (a) PSNR histogram and (b) SSIM histogram.

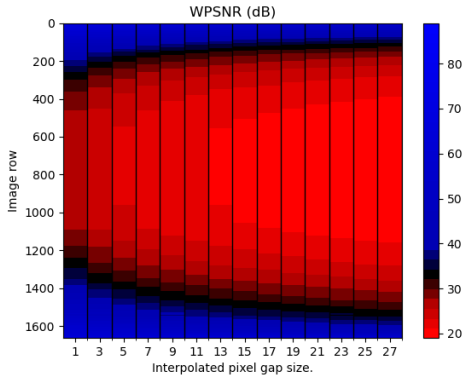


Figure 5: WPSNR results.

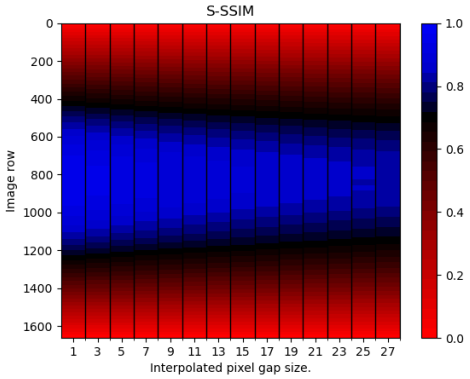
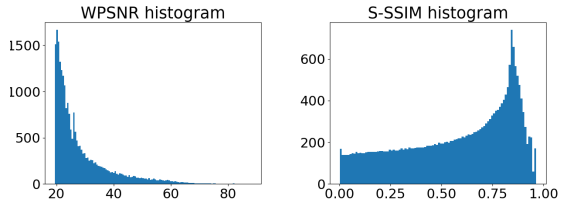


Figure 6: S-SSIM results.



(a) (b)

Figure 7: (a) WPSNR histogram and (b) S-SSIM histogram.

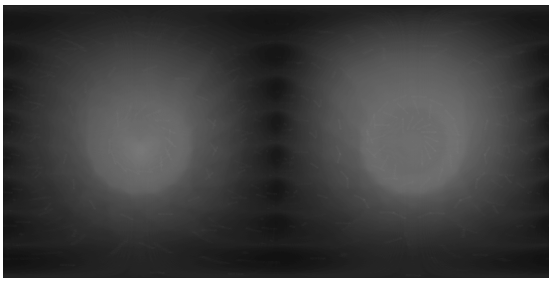


Figure 8: ROAD results, the values are scaled for better visualization.

different data from the standard SSIM. Highly redundant regions in SSIM vanish in S-SSIM due to the cosine factor of the weighting (Equation 1). However, it is possible to notice in Figure 6 a linear perceptual quality decrease towards the middle rows as long as the size of the interpolated pixel gap increases.

As expected from the equirectangular projection, these results show how the visual alterations of these type of images have a consine behavior of relevant pixels. However, these results do not show how the relevant pixels are structured across the image.

The pixel impact analysis produced interesting results on how the redundancy is scattered across equirectangular images. The computation of the localized SSIM metric shows, in Figure 9, the redundancy structure when projecting a 360° photo using the equirectangular projection. In this resulting image, dark pixels represent highly redundant regions and light pixel regions where the redundancy is minimum. The results reveal two circular regions centered in the longitudes $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ of the sphere that contain low redundancy and fade as we move further from their centers.

It is important to notice the existence of vertical dark lines; they represent the pixels used as reference for the interpolation and therefore they have no error when computing the metric values. The behaviour of these lines at the top and bottom rows of the image is interesting too. They get thicker as long as they get closer to the boundaries of the image due to the massive pixel stretching of the equirectangular projection in those regions.

The results from the ROAD metric (Figure 8) share a similar behavior as the SSIM. The difference of these two metrics is that the ROAD metric, as stated in Section 3.2.1, measures the gradients of a 3×3 region without any interpolation. This reveals the same circular

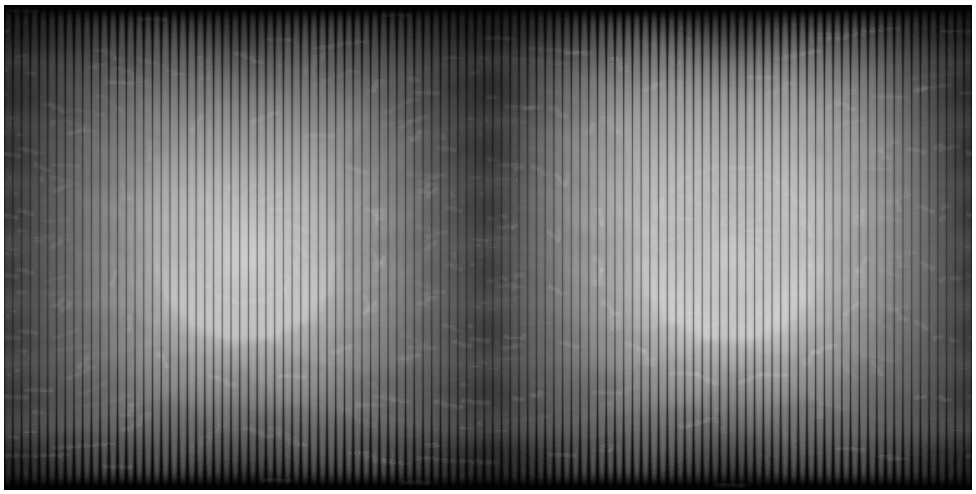


Figure 9: SSIM region results for 25 pixel gap size. Notice that the vertical black lines show the reference pixels from the interpolation where the error is non existent. Close to the top and bottom regions, the lines get thicker due to the highly redundant regions. It is important to mention that the images contain watermarks and are visible in the results.

low redundancy regions centered at longitudes $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. This means that those regions are the less distorted ones when we project the 3D sphere to the 2D plane using the equirectangular projection.

6 CONCLUSIONS

In this paper, we presented an analysis of pixel redundancy in equirectangular images. We analyzed the behavior of the visual quality assessment using standard metrics for regular 2D images and equirectangular images (PSNR and SSIM). In addition to that, we used the 360° image focused versions to complete the analysis and produce meaningful results. After removing several sets of pixels from rows and reconstructing them, we have shown how each metric behaves in a row-wise manner. The results showed consistency with previous statements of diverse authors and provided a visual demonstration of how the redundancy is increased at angles farther from the horizon line in equirectangular images. We revealed the redundancy structure of equirectangular images by using localized versions of SSIM and ROAD metrics, showing how sample values are stretched across pixels in images that use equirectangular projection. These results set a support for future research work in 360° image compression by providing how each portion of the equirectangular image contributes to the image quality. The knowledge of the redundancy structure in equirectangular images is useful to improve state of the art compression algorithms and improve the visual quality and immersion sensation in VR devices.

7 REFERENCES

[Aze19] Roberto Gerson Azevedo. Visual distortions in 360-degree videos. *IEEE Trans. Circuits Syst. Video Technol.*, PP(99), July 2019.

[Boy17] Jill Boyce. Spherical rotation orientation indication for HEVC and JEM coding of 360 degree video. In *Applications of Digital Image Processing XL*, volume 10396, page 103960I. International Society for Optics and Photonics, September 2017.

[Bud15] M Budagavi. 360 degrees video coding using region adaptive smoothing. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 750–754. ieeexplore.ieee.org, September 2015.

[Cha18] Shih-Hsiu Chang. Generating 360 outdoor panorama dataset with reliable sun position estimation. In *SIGGRAPH Asia 2018 Posters*, number Article 22 in SA '18, pages 1–2, New York, NY, USA, December 2018. Association for Computing Machinery.

[Che18] S Chen. Spherical structural similarity index for objective omnidirectional video quality assessment. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. ieeexplore.ieee.org, July 2018.

[DA17] A De Abreu. Look around you: Saliency maps for omnidirectional images in VR applications. In *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. ieeexplore.ieee.org, May 2017.

[DS16] F De Simone. Geometry-driven quantization for omnidirectional image coding.

In *2016 Picture Coding Symposium (PCS)*, pages 1–5. ieeexplore.ieee.org, December 2016.

[Dua18] H Duan. Perceptual quality assessment of omnidirectional images. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. ieeexplore.ieee.org, May 2018.

[Faced] Facebook. *Facebook 360 video*, 2021 (last accessed).

[Gar05] Roman Garnett. A universal noise removal algorithm with an impulse detector. *IEEE Trans. Image Process.*, 14(11):1747–1754, November 2005.

[Gooed] Google. *Tilt Brush by Google*, 2021 (last accessed).

[Gut18] Jesús Gutiérrez. Toolbox and dataset for the development of saliency and scanpath models for omnidirectional/360 still images. *Signal Processing: Image Communication*, 69:35–42, 2018.

[Hua18] Mingkai Huang. Modeling the perceptual quality of immersive images rendered on head mounted displays: Resolution and compression. *IEEE Trans. Image Process.*, August 2018.

[Jam19] M Jamali. Comparison of 3D 360-degree video compression performance using different projections. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–6. ieeexplore.ieee.org, May 2019.

[Lee17] S-H Lee. Omnidirectional video coding using latitude adaptive down-sampling and pixel rearrangement. *Electron. Lett.*, 53(10):655–657, April 2017.

[McCed] John McCormac. *SceneNet, https://robotvault.bitbucket.io/scenenet-rgbd.html*, 2020 (last accessed).

[oITed] Visual Computing Lab Institute of Information Technology. *3D60 Dataset, https://vcl.iti.gr/360-dataset/*, 2020 (last accessed).

[oTed] Massachusetts Institute of Technology. *SUN 360 dataset, http://people.csail.mit.edu/jxiao/SUN360/*, 2020 (last accessed).

[Sit18] Vincent Sitzmann. Saliency in VR: How do people explore virtual environments? *IEEE Trans. Vis. Comput. Graph.*, 24(4):1633–1642, April 2018.

[Soned] Shuran Song. *SUNCG, https://sscnet.cs.princeton.edu/*, 2020 (last accessed).

[Sun17] W Sun. CVIQD: Subjective quality evaluation of compressed virtual reality images. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3450–3454. ieeexplore.ieee.org, September 2017.

[Sun18] W Sun. A Large-Scale compressed 360-degree spherical image database: From subjective quality evaluation to objective model comparison. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, August 2018.

[Sze10] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[Wan04] Zhou Wang. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, April 2004.

[You16] R G Youvalari. Analysis of regional down-sampling methods for coding of omnidirectional video. In *2016 Picture Coding Symposium (PCS)*, pages 1–5. ieeexplore.ieee.org, December 2016.

[Yu15] M Yu. A framework to evaluate omnidirectional video coding schemes. In *2015 IEEE International Symposium on Mixed and Augmented Reality*, pages 31–36. ieeexplore.ieee.org, September 2015.

Data Visualisation for Supporting Linguists in the Analysis of Toxic Messages

Ecem Kavaz[1], Anna Puig[1], Inmaculada Rodriguez[1], Mariona Taule[2], and Montserrat Nofre[1]

[1]Department of Mathematics and Computer Science, University of Barcelona
[2]Department of Catalan Philology and General Linguistics, University of Barcelona
Barcelona, Spain
ekavazka27@alumnes.ub.edu

Keywords: Data visualisation, Corpus Annotation, Toxicity, Hate Speech

Abstract: The goal of this research is to provide linguists with visualisations for analysing the results of their hate speech annotation. These visualisations consist of a set of interactive graphs for analysing the global distribution of annotated messages, finding relationships between features, and detecting inconsistencies in the annotation. We used a corpus that includes 1,262 comments posted in response to different Spanish online new articles. The comments were annotated with features such as sarcasm, mockery, insult, improper language, constructivity and argumentation, as well as with level of toxicity ('not-toxic', 'mildly toxic', 'toxic' or 'very toxic'). We evaluated the selected visualisations with users to assess the graphs' comprehensibility, interpretability and attractiveness. One of the lessons learned from the study is the usefulness of mixed visualisations that include simple graphs (Bar, Heat map) - to facilitate the familiarisation with the results of the annotated corpus together with more complex ones (Sankey, Spider or Chord) - to explore and identify relationships between features and to find inconsistencies.

1 INTRODUCTION

Social media have become a powerful tool for many people for self-expression as they can share their voices and opinions freely even anonymously if desired. These platforms let people to use their freedom of speech very actively and effortlessly from the comfort of their homes and at any time (Paschalides et al., 2019). As the use of social media has increased, the multimedia data available has also increased, which provides researchers with greater opportunities to examine these data. On a daily basis, millions of messages are created and shared on different online platforms (Chen et al., 2017). The news comment section offered by some online newspapers is one of the possible spaces in which readers can express their opinions, although sometimes these opinions can be conveyed in an aggressive, offensive or inappropriate manner, especially when they are given anonymously or under a false name. This offensive, abusive or toxic language can be labelled as hate speech. It can be simply described as a kind of speech that attacks a person or a group based on characteristics such as race, religion, ethnic origin, national origin, gender, disability, sexual orientation, or gender identity (Gagliardone et al., 2015). In this context, frameworks and tools for automatically classifying messages are becoming ever more essential for detecting the trends and spreading patterns that will help to identify anomalous behaviour and hate speech (Florio et al., 2020).

In recent years, methods for the automatic classification of hate speech messages have been widely studied in different social networks and areas (Paschalides et al., 2019), (Grosman et al., 2020). The quality of these frameworks depends greatly on the algorithms used in NLP (Natural Language Processing), but also on having access to a sufficiently large corpus of annotated messages in the training steps of these algorithms (Frénay and Verleysen, 2013). This training dataset usually consists of messages (including tweets and comments), which are manually annotated by humans. Indeed, the quality of this man-

ual annotation is a key point to ensure the success of the whole process. Annotation involves processing a large number of messages and tends to become a difficult and time-consuming task plagued by errors and inconsistencies. Linguists usually follow a well-controlled methodology in which a single message is annotated by several annotators (preferably experts). Afterwards, agreement must reached for all the annotations. Detecting errors, trends and inconsistencies efficiently in the individual and the agreed-upon annotations can be helpful to speed up and to guarantee the quality and reliability of the final annotation.

Considering the aforementioned aspects, which make annotation a complex and challenging task, data visualisation can be a helpful method that allows linguists to analyse the results. Viewing data as mere numbers conveys little meaning, whereas data visualisation helps people to process information more easily(Knaflic, 2015). Well-designed interactive data visualisations can appeal to people effortlessly (Wu et al., 2016). However, the design of the most suitable data visualisation in a particular context is not an easy task.

The goal of this research is to provide annotators with a set of visualisations for analysing the results of their hate speech annotation. We use the NewsCom-TOX corpus, which consists of comments posted in response to different Spanish online news articles annotated with toxicity. Concretely, we contribute with: (1) a set of interactive graphs to allow the annotators to see the global distribution of comments, find relationships between features and detect possible errors and inconsistencies in the annotation, and (2) the lessons learned from a preliminary user evaluation of the proposed visualisations to detect the most useful graphs for linguists.

2 RELATED WORK

In this section we first present research works aimed at using data visualisation for the monitoring of automatic annotation systems. We place the focus on the visualisation techniques used by the authors. We then consider works that are aimed at supporting linguistic annotators through meaningful visualisations.

2.1 Visualisations for monitoring automatic annotation systems

Visual analytics for automatic annotation systems aims to identify valuable information in social data. Concretely, the following research works analyse anomalous user behaviour, anomalous information spread and the use of toxic language. (Shi et al., 2019) carried out a survey on the visual analytics of anomalous user behaviours. The survey revealed four types of user behaviours, including social interaction, travel, network communication and transactions. For each of the four types of user behaviours, the authors analysed trends in common data types, anomaly detection techniques, visualisation techniques and interaction methods. Our research is focused on the first type of user behaviour, social interaction, i.e. the communication of ideas and opinions between people in online newspapers.

Fluxflow (Zhao et al., 2014) is an interactive data visualisation system designed to display and evaluate anomalous information spread (rumours and misinformation) on Twitter. The novel visualisation design consisted of packed circles (retweets) arranged along a timeline showing how an original message spreads among people over time. The size of the circles symbolised the power of the influence of a user and the colour represented an anomaly score. Similarly to FluxFlow, Episogram (Cao et al., 2015) was designed to analyse retweeting behaviours on Twitter. It showed the activity of each person separately and every message from each person separately in the form of single lines on a timeline. Nevertheless, this visualisation caused cluttering, making it hard to understand at first sight.

RumorLens (Resnick et al., 2014) was created to help journalists to detect the diffusion of rumours on online social media and, once again, Twitter was the source of data for this research. The authors used a Sankey diagram to effectively summarise the diffusion of a rumour since it makes it very simple to follow and see people's decisions regarding posting or not posting the rumour.

Mandola (Paschalides et al., 2019) was designed with NLP and ML techniques to monitor, and detect online-hate speech on online social media. The Mandola dashboard included the so called Hate-map and Hotspot Map visualisations, both showing findings on a world map. While Hate-Map displays hate data with heat spots in certain countries, Hotspot Maps have a colour scale representing six levels of hate speech. There was also a Heat map that displays hate speech in five topic areas, separated by years. Mandola is close to our research because of its analysis of hate speech. The difference resides in that our goal is to use visualisations to support linguists in the annotation process, whereas Mandola aimed to use visualisations for the monitoring of an automatic annotation system.

Overall, previous research focused on novel approaches for monitoring automatic annotation sys-

tems using different visualisation techniques, such as Sankey diagrams and Heat maps. In this paper, we use these and other types of graphs to help linguists to visualise the results of their annotations.

2.2 Visualisations for supporting linguist annotators

Several tools and platforms (i.e. set of tools) are available to support the task of annotating a corpus. All of them provide basic functionality for data annotation. Nevertheless, some of them do not support more advanced functionalities, such as the management of the inter-annotator agreement. The inter-annotator agreement is a measure of how well two (or more) annotators can make the same annotation decision for a certain feature. This measure may impact the quality and efficacy of the annotation process.

For instance, Brat (Stenetorp et al., 2012) is a mainstream annotation tool that does not allow for several annotators, with a consequent non-support of inter-annotator agreement. Another tool is MAT, which supports the annotation and the management of multiple annotators through a web interface (MAT, 2020). IBM Watson Knowledge Studio, which is integrated in the well-known Watson platform, includes an annotation tool, for creating a training corpus that is well designed and documented (Watson, 2020).

The above-mentioned tools provide support to the annotators but they do not use data visualisations. In contrast, Eras (Grosman et al., 2020) and WebAnno (Yimam et al., 2013), use data visualisation to show, for example, the results of annotators agreement through a Heat map. The visualisations we present in this paper are in line with these two frameworks since we also aim to facilitate and improve annotators' work by means of new, meaningful visualisations.

3 USED DATA

In this section we describe the NewsCom-TOX corpus, the dataset used for developing the visualisations, and the annotation tagset used.

The NewsCom-TOX corpus consists of 1,262 comments posted in response to different articles extracted from Spanish online newspapers from August 2017 to May 2019 annotated with toxicity. The articles selected cover four different topics -economy, politics, religion and immigration- and the comments were selected in the same order in which they appear in the time thread in the web. Those comments that were duplicated were removed. Table 1 shows the distribution of comments per topic and the corresponding newspaper from which they were obtained.

Topic	Comments	Newspaper
Economy	309	La Información, El País
Politics	239	Huffpost, La Vanguardia
Religion	298	Xataca Ciencia
Inmigration	416	El Confidencial
Total	1262	

Table 1: Distribution of comments per topic

In order to have a balanced representation of comments per topic, two different news articles were needed in the case of economy and immigration-related topics. Articles were selected to potentially lead to controversy with the aim of finding comments with opposing opinions and examples of toxic language. Toxicity is difficult to define, possibly because it can be expressed at different levels and in different ways (Ross et al., 2017), (Davidson et al., 2017), (Fortuna and Nunes, 2018). In order to reflect this diversity in the expression of toxicity, the proposal is to assign different levels of toxicity, indicating whether the comment is 'not toxic', 'mildly toxic', 'toxic' or 'very toxic'. With the aim of reducing the subjectivity in the annotation and, therefore, also the disagreement between annotators, we propose first annotating different linguistic features such as sarcasm, mockery, insult, improper language, constructivity and argumentation. These binary features allow us to discriminate the level of toxicity of the comments. Furthermore, some of these features can be correlated, for instance argumentation and constructivity, insult and improper language, and these correlations are also useful when assigning the level of toxicity. Our hypothesis is that the combination of these features helps to determine the level of toxicity in a more objective way. The tagset used for the annotation of comments with toxicity is the following:

- <**argumentation**>: indicates that the comment gives arguments or reasoned explanations or grounds opinions with evidences.
- <**constructivity**>: a comment is constructive when it is respectful and polite (regardless of whether it is in favour or against the content of the article or of another comment)[1], when it intends to create an enriching and useful dialogue, when it contributes with new knowledge, ideas and pro-

[1]We can find two types of comments, those that comment on some specific or general aspect of the article, or those that are responses to another comment.

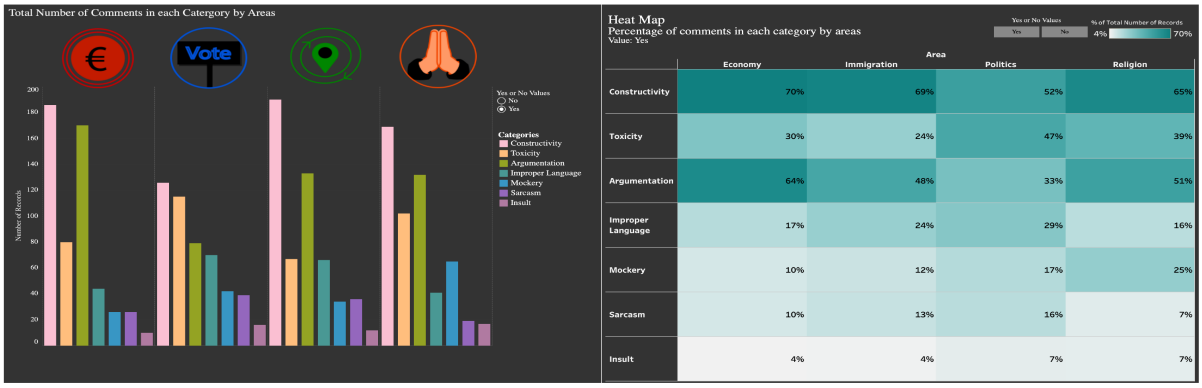


Figure 1: Visualisations of Global Data Distribution. Bar chart and Heat map.

posals and offers new perspectives and insights to approach the subject.

- <**sarcasm**>: a comment is sarcastic when the content is ironic -that is, when the writer uses words that mean the opposite of what he really wants to say- and when it is accompanied by a harsh, sharp and negative criticism and made in bad faith. Ironic comments without intention to cause pain (without a negative load) are not considered toxic and are tagged as <sarcasm=no>.
- <**mockery**>: indicates that the comment ridicules, mocks or humiliates a person or group.
- <**insult**>: indicates that the comment contains one or more insults or slurs with the intention to offend a person or group.
- <**improper language**>: indicates that the comment contains language not consider to be proper or that is vulgar and impolite and/or which includes rude words.
- <**toxicity**>: a comment is toxic when it attacks, denigrates or disqualifies a person or group on the basis of certain characteristics such as race, ethnicity, nationality, religion, gender and sexual orientation, among others. This attack can be expressed in different ways -directly (through insult, mockery and inappropriate humour) or indirectly (for instance through sarcasm)- and at different levels of intensity, that is at different levels of toxicity (the most aggressive being those comments that incite hate or even physical violence).

It should be noted that all these tags have binary values (value= yes/no) except the toxicity tag, which has four values (<1= non-toxic>; <2= mildly toxic>; <3= toxic> and <4: very toxic>). The level of toxicity is determined by the presence and combination of the features presented above. In fact, these features are different ways or mechanisms to express the toxicity and, therefore, they also help to define what is meant by toxicity. The more negative features appear in the comment, the higher the level of toxicity. For instance, we tag as 'mildly toxic' comments in which only one feature appears, the most frequent being <sarcasm>, <mockery> and <improper language>, whereas in comments tagged as <very toxic> the combination of features is higher than two, an especially frequent combination is <improper language>, <mockery> and <insult>. This annotation allows us to establish fine grained criteria for analysing and better defining what can be considered a comment with toxic language or hate speech.

4 GRAPHS TO VISUALISE TOXIC MESSAGES

We have used multiple types of graphs and diagrams to visualise the annotated data, which allow to analyse and measure these kind of data. We have used Tableau[2] (an interactive data visualisation software), DisplayR[3] (an online visualisation tool) and lastly SankeyMatic[4] (an online Sankey diagram builder) to create data visualisations.

4.1 Visualisations of global distributions.

All the visualisations are combined in a Tableau story, which offers an individual page for each visualisation. The first page of our Tableau project is an introduction including four icons representing the four topics (economy, politics, immigration and religion), a brief explanation that provides an overview of the dataset

[2]https://www.tableau.com
[3]https://www.displayr.com
[4]http://www.sankeymatic.com

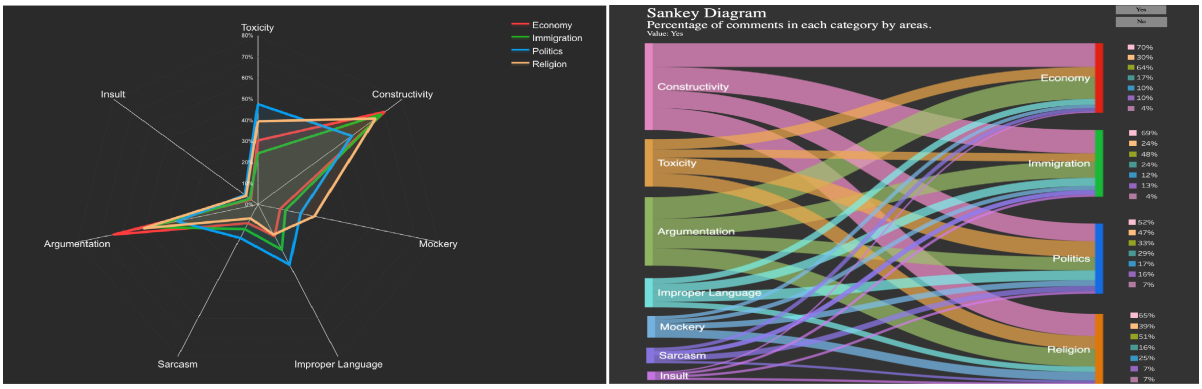


Figure 2: Visualisations of Global Data Distribution. Spider Chart and Sankey diagram.

(such as the origin of the messages and the questions to be answered using the visualisations) and the text visualisation of the seven features used in the annotation of toxicity. Each feature is represented by size, depending on the total number of 'yes' comments they have. On the tooltips, there is an example comment about the feature and the number of comments annotated as 'yes' for that feature. This text visualisation is a summary of upcoming visualisations.

Created graphs are then separated into three groups: (1) visualisations to analyse the global distributions of messages by topics (2) visualisations of the relationships between features, and (3) visualisations by features.

Global data distribution visualisations aim to explore data in terms of topics and features of the messages (see Figures 1 and 2). The first graph in this section is a simple Bar chart. The Bar chart has four sections represented by icons for four topics. Features are colour coded and displayed in each topic individually. The Bar chart displays features in terms of their total numbers. There is a filter that allows users to select values ('yes' or 'no'). Please note that every message has been annotated with tags 'yes or no' for every feature. For example, a comment tagged with 'yes' in the argumentation, mockery and sarcasm features, can be tagged as 'no' in constructivity and insult Secondly, a Heat map[5] is created to show data as a whole and in a simple way without other complex design elements. The Heat map illustrates the features for each topic including a filter to choose 'yes' or 'no'.

The Spider[6] graph is created to present the global distribution of toxic messages for each topic. The Spider web has seven sides representing the seven features that appear in the annotated messages. Topics are represented by their assigned colours. On the Spider chart, each topic has a unique shape made up of points combined together with lines. Each point shows the selected percentage value ('yes' or 'no') percentage of a feature and lines are used to combine the points to form the shape. In the page of the Spider graph, there is a toxicity symbol which displays the mean level of toxicity for each topic.

Another approach for examining the global distribution of data is to create a Sankey diagram[7]. In this case, features are presented on the left side, splitting to form areas on the right side. The Sankey diagram also allows users to choose 'yes' or 'no' values. Next to the topics, there are keys that illustrates the percentage of each feature in terms of the selected value.

4.2 Visualisations of relationships between features

Two visualisations have been created to observe relationship between features (see Figure 3). These two visualisations aim to show a comparison between the features annotated in the comments, which will mainly allow annotators/linguists to analyse their hypothesis.One of them is a Scatter plot with icons for topics, representing one feature on the x-axis and one on the y-axis. Each axis shows the possible values of the feature represented on that axis (yes-no). There is a filter where users can choose the desired feature for each axis to compare with each other. Thus, with this graph, users can examine the number of occurrences of each combination of values

('no-no', 'no-yes', 'yes-no', and 'yes-yes') in selected features. For example, the combination 'no-no'

[5]A heat map displays data values with colour, usually by intensity or hue.

[6]A spider graph displays multivariate data with three or more quantitative variables represented on axes starting from the same point.

[7]Sankey diagram is a type of flow diagram where the width of the links are proportional to the size of the data.

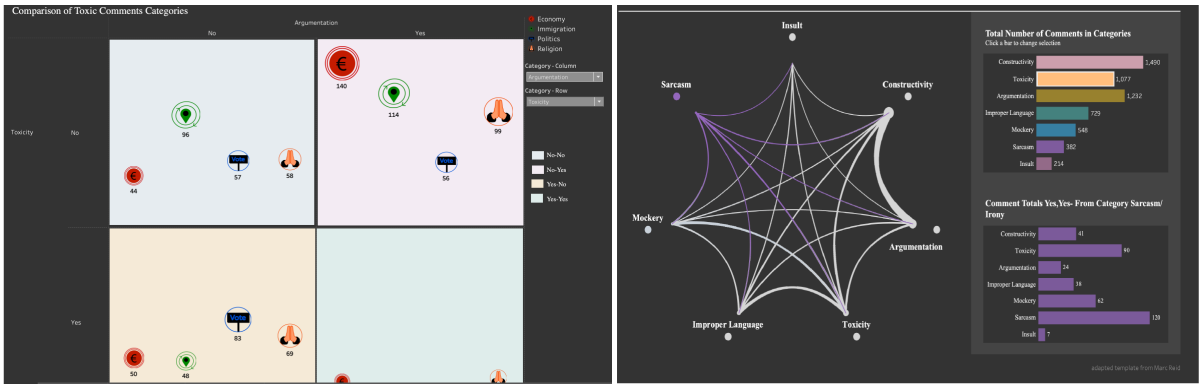


Figure 3: visualisations of relationships between features. Scatter plot and Chord diagram.

for constructivity and argumentation refers to messages tagged as 'no' for the constructive feature and 'no' for the argumentation feature.

The other visualisation is a Chord diagram[8] (Reid, 2020). While a Scatter plot analyses data in topics and features, a Chord diagram does not separate data into topics but rather analyses the data as a whole in the toxic comment features. The Chord diagram also has value selection buttons for 'no-no', 'no-yes', 'yes-no' and 'yes-yes' comments in features. The Chord diagram lets users select a feature and then highlights the arcs of that selected feature, which allows users to examine features individually in relation to the other features. The thickness of the arc that connects two features represents the number of messages annotated by those two features with the value selection. The total number of comments in the selected feature and others are also displayed on a separate Bar chart, which displays the total amount of comments in each feature.

4.3 Visualisations by features

One of the main objectives of this study is to measure the level of toxicity. There are four levels of toxicity, namely, non-toxic (1), mildly toxic (2), toxic (3) and very toxic (4). The aim of these graphs include, analysing annotated corpus, hypotheses and most importantly to identify inconsistencies. Although the mean of toxicity of each topic is illustrated on the Spider, and the Sankey diagram, three additional graphs are created to analyse the level of toxicity in features. Firstly, in the left part of Figure 4 there is a Bar chart in which each level of toxicity is represented with a separate bar for each feature. In this graph, the 'yes' and 'no' values are displayed separately, and can be selected from the list, while a tooltip displays the total number of messages at each level of toxicity.

A Sankey diagram is proposed to show the level of toxicity for each feature (see the right hand side of Figure 4). In this diagram, the data come from the level of toxicity (non-toxic to very toxic) and go into features with lines in which the thickness of the lines represents the total number of comments in each level and its connected feature. The same toxicity icon is used to show the total number of messages on the tooltip for each features at each level of toxicity. Lastly, in Figure 4 there is a Treemap in which the level of toxicity is represented for a selected feature. The Treemap is more complex as it has many components to explore and is highly interactive. There are four different maps for each topic. All levels of toxicity (1 to 4) are divided into 'yes' and 'no' are represented by colours blue to red. The darkest blue represents the lowest toxicity and the darkest red shows the highest toxicity. For example, if the chosen feature is constructive, we can see the number of messages at the level of toxicity 2 separately as 'yes' or 'no' in the economy topic.

There is another small Treemap which represents the total number of messages by topic and, when the topic is chosen, the Treemap shows the total number of messages at each level of toxicity without dividing them into 'yes' or 'no'. There is a filter to change the range of levels. Treemap allows users to compare the levels of toxicity in terms of features and also topics. There is another Treemap that represents the total number of messages in a topic and, when the topic is chosen, the Treemap shows the total number of messages at each level without dividing them into 'yes' or 'no'. There is a filter to change the range of levels. In this graph 'yes' or 'no' values are displayed separately and they can be selected from the list, while the tooltip displays the total number of messages at each level of toxicity.

[8] A chord diagram shows the connection between features. Thickness of the links represents the data size.

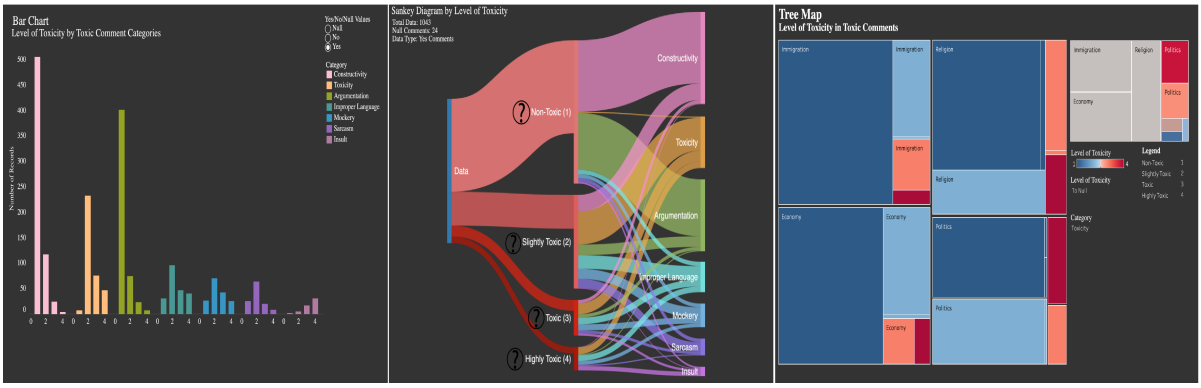


Figure 4: Visualisations by features. Left to right, Bar chart, Sankey diagram and Treemap.

5 EVALUATION

5.1 Methodology

The goal of the evaluation was to assess comprehensibility (how well the graph communicates the information to the user), interpretability (how well the user can extract meaning from the visualisation), attractiveness (to what extent the visualisation is visually appealing for the user), and to gather users' opinions.

The evaluation was exploratory, aimed at obtaining participants' perceptions. It was unmoderated, performed through an online questionnaire. A total number of eight participants were recruited, including five females and three males. The questionnaire consisted of demographic questions followed by three visualisation tasks lasting up to 30-35 minutes. Demographic questions included, gender, age, and two questions related to the participants' prior degree of experience in message annotation and visual analytics. Most of the participants had prior experience in data visualisation and more than half of the participants had expertise in message annotation.

The visualisation tasks intertwined links to our Tableau visualisations and questions related to them:

- Task 1: Please, follow the tableau link of "visualisations of global distribution" (Bar Chart, Heat map, Spider Chart, and Sankey Diagram).
  - *Q1-Comprehensibility*: "Score from 1 (the most negative) to 5 (to most positive) how well the graph communicates the global distribution." Please, justify the best and the worse scores.
  - *Q2-Interpretability*: "Score from 1 (very difficult) to 5 (very easy) how easy is to interpret the graph."
  - *Q3-Attractiveness*: "Score from 1 (very bad) to 5 (very good) the visual appeal of the graph."
- Task 2: Please, follow the tableau link of "visualisations of relationships between features" (Scatter Chart and Chord Diagram).Same questions as TASK1: Q1-Q3.
- Task 3: Please, follow the tableau link of "visualisations by features" (Bar Chart, Sankey Diagram, and Treemap). Same questions as TASK1: Q1-Q3.

The questions were chosen to explore the three dimensions of visualisations including: the communication of information (how well the visualisations communicate), the interpretation of graphs (how easy or difficult it is to understand the graphs, and the attractiveness (how appealing the visualisations). There were nine closed-questions (i.e. score 1 to 5), three of which were followed by three open-questions (i.e. justify your answer) presented in the questionnaire. Data were collected anonymously.

5.2 Results

For each task, we first analysed answers to questions (Q1-Q3), then we presented the qualitative data arising from users' comments.

Task 1 : visualisations of global distribution

According to the results in Figure 5, the Heat map received the highest scores in *Q1-Comprehensibility*, followed by the Bar chart. The Spider graph received relatively good scores in this question, whereas the Sankey diagram was the least favourite graph. The results show that the Bar chart is the easiest to understand in *Q2-Interpretability*, with 75 percentage of participants scoring it as very easy and the Sankey diagram as the hardest to understand, with relatively low scores. The Spider chart and the Sankey diagram scored the highest in *Q3-Attractiveness*.

In the first task, the Bar chart was the most favoured graph in the dimensions of comprehensibility and attractiveness. The Bar chart was the second

favourite graph in the communication of information dimension, scoring slightly lower than the Heat map. The Bar chart was described by multiple participants

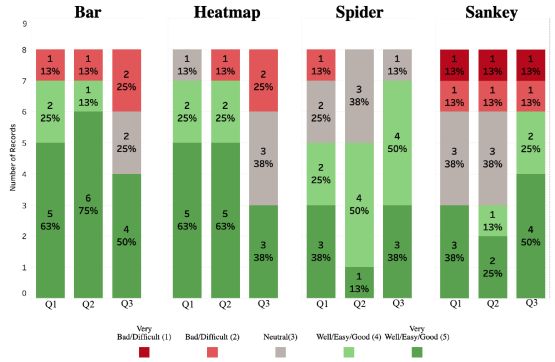


Figure 5: Stacked bar chart displaying the results in Task 1. Q1-Comprehensibility, Q2-Interpretability and Q3-Attractiveness.

as very easy to understand and use. A couple of participants commented that they favoured the Bar chart as they are used to analysing this type of graph. Another positive comment was that the colours of the bars were both appealing and made the visualisation clearer. However, various participants commented that the Bar chart should have included percentages along with the actual total number of comments in the features. The Heat map received the highest scores for *Q1-Comprehensibility*. The results also show that the Heat map was easy to understand and communicates information well, however, visually it was not as attractive as the other three visualisations.

The Spider graph received mixed reviews from users. One participant stated that *"I think the Spider gives a very clear idea of the distribution of attributes by features, with its isolated and superimposed surfaces, it reflects very clearly what has been annotated"*. Another participant stated that *"The Spider chart is also a good way of displaying the data for a general comparison across topics. However, the comparison among features within a specific topic is less clear when only one axis contains the percentage indicator."* Another participant agreed with this comment by stating that comparisons between the features were a little bit difficult when the values matched. The results suggested that the Spider graph was visually very appealing but slightly more difficult to understand than the Bar chart and the Heat map.

The Sankey diagram was the least favoured of all the graphs, even though it received high scores in the visual appeal section. The majority of the participants commented that the shape of the graph was confusing, difficult to understand and that it was difficult to compare features of messages and topics. One participant stated that *"The Sankey diagram seems rather chaotic compared to the other ones"*. Some participants commented that they needed to pay extra attention to the Sankey diagram because of its complexity.

Multiple participants commented that to improve communication, all of the graphs should have included both percentage values and actual total numbers. The interactivity of the graphs was an important element as results show that participants preferred the Bar chart since it was very interactive while they did not like the fact that the Sankey diagram was static. The comments suggest that participants favoured graphs on which they could spend the least possible time as they did not want to waste time trying to understand the graphs.

Task 2: visualisations of relationships between features

In the *Q1-Comprehensibility*, (see Figure 6), the Scatter plot and the Chord diagram achieved similar scores. The Scatter plot is favoured slightly more by the participants. The Scatter plot achieved higher scores in the *Q2-Interpretability* than the chord diagram, with a total of seven high scores, while the Chord diagram received only two high scores. For the *Q3-Attractiveness*, the Chord diagram was considered visually more attractive than the Scatter plot and it obtained significantly high scores.

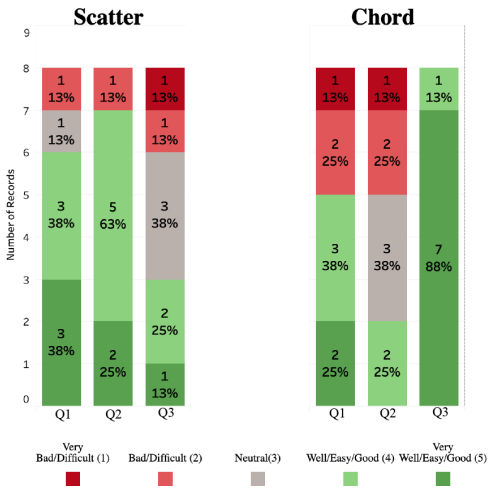


Figure 6: Stacked bar chart displaying the results in Task 2. Q1-Comprehensibility, Q2-Interpretability and Q3-Attractiveness.

In the second task, most of the participants agreed that the Scatter plot was easier to understand than the Chord diagram, and the Chord diagram was visually more appealing than the Scatter plot. There were mostly positive views about both graphs in terms of their ability to communicate the information and

the Scatter plot was described as more intuitive than the Chord diagram. Participants commented that the Scatter plot provided a clear view for comparing topics and features, and that it was also very engaging thanks to its interactivity. A participant commented that *"with the Scatter plot it is easier to understand the correlation between the different features, an explanatory legend appears that also includes an example, the relationship is quickly associated with the number of examples, it displays the relationships between features according to the topic. It is more intuitive"*. On the other hand, some participants have found the Chord diagram, relatively easy to understand and useful, especially in terms of the overall view of the data. One participant stated that *"Chord Diagram helps to understand the different relationships between features very well and allows interaction by focusing on various elements and their intersections, providing very valuable information."*

Task3: visualisations by features

The Bar chart received the highest scores in the *Q1-Comprehensibility*, (see Figure 7). The Sankey diagram and the Treemap obtained similar scores. In the *Q2-Interpretability*, the Bar chart was considered to be the easiest and the Treemap the hardest to understand. As in task 1, the Sankey diagram given the highest scores in the *Q3-Attractiveness*, followed by the Bar chart which received the second highest scores in Task 3. The Treemap was the least preferred graph in this task. In the Task 3 (see Figure 7), the

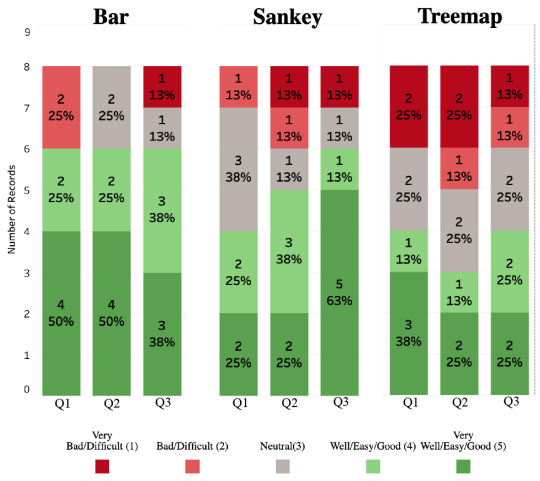


Figure 7: Stacked bar chart displaying the results in Task 3. Q1-Comprehensibility, Q2-Interpretability and Q3-Attractiveness.

most preferred graph was the Bar chart in the dimensions of comprehensibility and interpretability. This is in line with other comments since the Bar chart is very commonly used, it is easier to interpret, and comparing the features with each other was therefore easier. Also analysing a Bar chart does not require prior knowledge of the visual analytics, which was another reason for the popularity of the Bar chart.

The Sankey diagram had the highest score in the visual appeal section. There were mixed observations about the Sankey diagram as some participants found it very clear while some did not. One participant stated that *"The Sankey Diagram illustrates very well the distribution of features according to the level of toxicity and vice versa, it is very easy to understand and provides a lot of information, it would be useful if the information could be isolated interactively"*. Various participants agreed with the statement and they have described the Sankey diagram as being rather difficult to understand at first but afterwards it was clear as the information was globally displayed. Having more interactive elements could solve the problems of interpretation as users can filter down to explore features separately. On the other hand, some participants described it as chaotic and confusing.

Lastly, the results showed that the Treemap was the least liked graph in this task. An interesting finding was that participants described the Treemap as difficult to follow and understand though it stored more information than the Sankey diagram and the Bar chart, such as comparisons in topics and comparisons of 'yes' and 'no' comments together. For example, a participant commented that *"The Treemap, which at first glance seems more unpleasant, when you look at it closely, it gives very interesting information, which is when we find the same attribute labelled two different ways, helping then in the finding of inconsistencies in the annotation. For example, in the Economy topic, toxicity level 2 is represented by annotated comments such as Toxic = Yes and Toxic = No"*. However, most of the participants commented that they did not understand the Treemap.

Overall, results show that participants were attracted by the graphs that were easier to understand. The appeal of the graphs was also an important element, though, not as important than the ease of use. Many participants were not attracted to the graphs they did not understand and did not want to spend time on them, even though they liked their appearance more. Another finding suggests that participants would have benefited from greater guidance in the complex visualisations with various elements, which would have facilitated the comprehension, thereby proving more attractive. An idea is to combine simple graphics with more complex ones to create simple graphics like the Bar chart to guide graphs like the Treemap.

6 CONCLUSIONS AND FUTURE WORK

This paper presents a study related to data visualisation of hate speech (toxicity) annotations. To do so, we proposed various data visualisations, that includes different diagrams to assist annotators in the detection of inconsistent annotations, the analysis of the global data distribution and the discovery of relationships between features. We used a complex corpus composed by comments posted in different Spanish online new articles. All the comments were annotated using a new tagset that combines several features to determine the level of toxicity in a more objective way. The challenge in the proposed data visualisations was providing annotators with a wide spectrum of diagrams that highlight trends and relationships in an easy and comprehensible way. We proposed a diversity of visualisations (from those that were well-known to some others that could be new or unfamiliar for annotators). We conducted a preliminary evaluation of the proposed visualisations from collected qualitative and quantitative data obtained from a small but representative group of annotators. That is, they have different degree of expertise on visual analytic tools, and different knowledge in the annotation of corpus. We evaluated several dimensions of the visualisation experience (comprehensibility, interpretability and attractiveness).

In the following, we share our lessons learned, including design recommendations useful for future visualisation studies on corpus annotation. Regarding the comprehension of visualisations, the first consideration is that the participants mostly prefer the simple graphics (such as the Bar chart or Heat map), probably because annotators already acquainted with them on their daily annotation. However, the more expert participants in using visual analysis the more they valued visualisations that show more complex details (like the Sankey, the Spider or the Chord Diagram). Moreover, we also observed that in some cases the perception of the same diagram differed depending on which task and when it was visualised. For example, participants who rated the Sankey diagram as difficult to understand in Task 1, they found it easier in Task 3. Keeping this understanding in mind, the use of mixed visualisations that include easiest graphs -to facilitate "the landing" in the annotated corpus-, together with the more complex ones - to explore more complex relations - could help to enhance the visualisation effectiveness. In relation to the interpretation of the visualisations, we found that participants appreciated having redundant information in the graphs (for instance, percentages and absolute values), and also they highly valued the interaction offered by some graphs, particularly in the ones that were more complex to understand (such as the Scatter Plot and the Tree Map). The use of interactivity by prioritising the most important attributes such as features, topics, Yes/No values, etc. to select the data to be shown is also an important fact to remind take into account in future visualisations. Last but not least, participants gave rather positive comments in open questions in those graphs which they scored high in the attractiveness dimension (such the Scatter Plot and the Spider Diagram). Making visualisations attractive and clear, using suitable colours and icons, is engaging and, more importantly, instructive and enlightening.

In the future, the first step is to design novel visualisations tailored according to linguists' needs with the findings of this paper and validate them by a larger group of annotators. Moreover, we plan to integrate the visualisations in earlier stages of the annotation process. Visualisation will serve as a tool to lead the annotation, letting users examine corpus as well as finding and editing inconsistent data. Additionally, we plan to introduce more valuable information related to the context in which the comments are obtained. Collecting information such as, where the message is post, who is the user, location and timestamps, could also help annotators to detect diffusion of hate speech.

ACKNOWLEDGEMENTS

To MISMIS-Language (PGC2018-096212-B-C33), CI-SUSTAIN (PID2019-104156GB-I00) and Accio COMRDI18-1-0010.

REFERENCES

Cao, N., Lin, Y.-R., Du, F., and Wang, D. (2015). Episogram: Visual summarization of egocentric social interactions. *IEEE Computer Graphics and Applications*, 36(5):72–81.

Chen, S., Lijing, L., and Yuan, X. (2017). Social media visual analytics. *Computer Graphics Forum*, 36(3):563–587.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

Florio, K., Basile, V., Polignano, M., Basile, P., and Patti, V. (2020). Time of your hate: The challenge of time in hate speech detection on social media. *Applied Sciences*, 10(12):4180.

Fortuna, P. and Nunes, S. (2018). A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):1–30.

Frénay, B. and Verleysen, M. (2013). Classification in the presence of label noise: a survey. *IEEE trans. on neural networks and learning systems*, 25(5):845–869.

Gagliardone, I., Gal, D., Alves, T., and Martinez, G. (2015). *Countering online hate speech*. Unesco Publishing.

Grosman, J. S., Furtado, P. H., Rodrigues, A. M., Schardong, G. G., Barbosa, S. D., and Lopes, H. C. (2020). Eras: Improving the quality control in the annotation process for natural language processing tasks. *Information Systems*, page 101553.

Knaflic, C. N. (2015). *Storytelling with data: A data visualization guide for business professionals*. John Wiley & Sons.

MAT (2020). The MITRE annotation toolkit. [Online; accessed 2-sep-2020].

Paschalides, D., Stephanidis, D., Andreou, A., Orphanou, K., Pallis, G., Dikaiakos, M. D., and Markatos, E. (2019). Mandola: A big-data processing and visualisation platform for monitoring and detecting online hate speech. *ACM Trans. on Internet Technology*, 37(4):1–21.

Reid, M. (2020). Creating a Chord Diagram with Tableau Prep and Desktop. [Online; accessed 13-sep-2020].

Resnick, P., Carton, S., Park, S., Shen, Y., and Zeffer, N. (2014). Rumorlens: A system for analyzing the impact of rumors and corrections in social media.

Ross, B., Rist, M., Carbonell, G., Cabrera, B., Kurowsky, N., and Wojatzki, M. (2017). Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

Shi, Y., Liu, Y., Tong, H., He, J., Yan, G., and Cao, N. (2019). Visual analytics of anomalous user behaviors: a survey. *arxiv:1905.06720v2*.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). Brat: a web-based tool for nlp-assisted text annotation.

Watson (2020). Watson knowledge studio. [Online; accessed 2-sep-2020].

Wu, Y., Cao, N., Gotz, D., Tan, Y.-P., and , Keim, D.-A. (2016). A survey on visual analytics of social media data. *IEEE trans. on multimedia*, 18(11):2135–2148.

Yimam, S. M., Gurevych, I., de Castilho, R. E., and Biemann, C. (2013). Webanno: A flexible, web-based and visually supported system for distributed annotations.

Zhao, J., Cao, N., Wen, Z., Song, Y., Lin, Y.-R., and Collins, C. (2014). fluxflow: Visual analysis of anomalous information spreading on social media. *IEEE trans. on Visualization and Computer Graphics*, 20(12):1773–1782.

A Deep CNN Model for Skin Cancer Detection and Classification

Masum Shah Junayed
Bahcesehir University
Istanbul 34439, Turkey
masumshahjunayed@gmail.com

Nipa Anjum, Abu Noman Md Sakib
Khulna University of Engineering & Tech.
Khulna, Bangladesh

Md Baharul Islam
American University of Malta
Bormla 1013, Malta
bislam.eng@gmail.com

ABSTRACT

Skin cancer is one of the most dangerous types of cancers that affect millions of people every year. The detection of skin cancer in the early stages is an expensive and challenging process. In recent studies, machine learning-based methods help dermatologists in classifying medical images. This paper proposes a deep learning-based model to detect and classify skin cancer using the concept of deep Convolution Neural Network (CNN). Initially, we collected a dataset that includes four skin cancer image data before applying them in augmentation techniques to increase the accumulated dataset size. Then, we designed a deep CNN model to train our dataset. On the test data, our model receives 95.98% accuracy that exceeds the two pre-train models, GoogleNet by 1.76% and MobileNet by 1.12%, respectively. The proposed deep CNN model also beats other contemporaneous models while being computationally comparable.

Keywords

Skin Cancer, Dataset, Data Augmentation, Deep CNN, Medical Image, Computer Vision.

1 INTRODUCTION

An uncontrollable growth of abnormal cells that appears on our skin is called skin cancer. It happens when some unusual DNA damage activates mutations [1] that helps skin cells to increase very fast, and this forms malignant tumours. Some major skin cancer types are basal cell carcinoma, squamous cell carcinoma, actinic keratosis and malignant melanoma. Basal cell carcinoma is a nonmelanoma type cancer that starts with different sized nodules [1]. The second type of skin cancer is squamous cell carcinoma that creates scaly red marks, open inflammations, uneven clots or lumps in the skin. In the U.S., more than 1 million cases of this type are diagnosed each year [2]. A different kind of skin cancer is actinic keratosis that is the initial stage of squamous cell carcinoma [3]. The last one is a serious one among others which is malignant melanoma. It occurs when pigment cells grow without any control. Without any indication, 90% of melanomas are diagnosed as primary tumours [4].

Ultraviolet radiation triggers these tumours to increase. These cancer cells are seen mostly on sun-exposed areas like ear, lips, face, around the eye, scalp, neck, hand. American Cancer Society [5] estimated the death rate for melanoma skin cancer to be 6850, where new cases of this estimation are 100,350 where 60,190 people are male, and 40160 are female. They also estimated that the death rate is higher in male than female. A specialist do visual analysis by analysing the pigmented lesions by changing size, irregular shape and colour. Then histopathologic diagnosis is another way to examine the cancer cell. Here experienced dermatopathologist help to determine the melanoma. The earlier cancer detection can be useful to save many lives. Vocaturo et al. [6] have shown different machine learning techniques for automatically detecting melanoma. These techniques help to research more on skin cancers [7]. However, we need systems that can accurately predict skin cancer or not and its type.

Non-pigmented lesions like basal cell carcinoma, squamous cell carcinoma, actinic keratosis and pigmented lesion like malignant melanoma have some clinical differences: bleeding, pain, and itching. But the appearance of these lesions on human body parts is very similar. So it is difficult to distinguish any of these skin cancers. Besides visual analysis, an automated system can help a physician find skin cancer types faster and easier and reduce patient life risk. Classification algorithms such as K-nearest neighbour, decision tree, deep learning, [8] logistic regression, support vector machine etc.

are used to describe how well these classifiers perform on dermoscopic images. In [9], K-mean clustering is used to segment skin cancer, and then features are extracted using grey level co-occurrence matrix. The support vector machine is used as a classifier. For faster and better prediction, A classification technique can also help the patient life that relies on the detection result [11] [12] [13] [10]. It motivates us to implement a deep learning approach to build a CNN model that can classify four different skin cancer types.

This paper proposes a deep convolutional neural network model that automatically detects and classifies actinic keratosis, basal cell carcinoma, malignant melanoma, and squamous cell carcinoma. To test our model, we collect a dataset and train it in our model with other two pre-train models. Our model provides satisfying accuracy compare to the state of the arts. The summary of our contribution is four folded.

- We collect a skin cancer dataset, size 800 images of four skin cancer classes that are, Actinic Keratosis, Basal Cell Carcinoma, Malignant Melanoma and Squamous Cell Carcinoma. We use data augmentation techniques for increasing the dataset at 5600 images that were split into training and test set for deep CNN models.
- A deep CNN model has been proposed that has convolution layers for extracting features and fully connected layers for classifying the cancer type. Regularization techniques like batch-normalization and dropout helped to reduce the overfitting.
- We present a comparative study of two pre-trained models that shows how our model differs from architecture and accuracy.

Rest of the paper is discussed as follows: Section 2 most related works, Section 3 our proposed deep CNN model, Section 4 experimental results and discussion, and we conclude our work with further research direction in Section 5.

2 RELATED WORKS

This section narrates some most related works on Skin Cancer using different techniques for detecting and classifying.

2.1 Learning based Approaches

Garg et al. [14] proposed a way to detect melanoma skin cancer by using some image processing techniques. They have implemented an ABCD rule called Asymmetry, Border Irregularity, Color, and Diameter. An illumination correction used before the skin lesion segmentation. They received the accuracy up to 91.6%. Tammineni et al. [15] proposed a melanoma segmentation technique for early detection of skin cancer. A melanoma segmentation technique used based on Gradient and Feature Adaptive Contour (GFAC) model. They experimented on PH2 dataset consisting of 200 images. The achieved accuracy for the proposed segmentation techniques is 98.64%. The acquired accuracy is well enough, but it is limited to a low sized dataset.

2.2 CNN based Approaches

Pham et al. [16] classified skin lesions with Deep CNN with Data Augmentation. They combined images from different sources such as ISBI Challenge, ISIC Archive, PH2 dataset to prepare the dataset. InceptionV4 was used as the model architecture, and the outcomes compared by using Support Vector Machine (SVM), Random Forest (RF) and Neural Network (NN) as classifiers. They achieved an overall accuracy of 89%. Even though the model architecture was InceptionV4, the gained accuracy was not satisfactory. Jordan et al. [17] proposed a method to classify multimodal skin lesion using deep learning. They have composed their dataset with 2917 cases from five classes. They have used modified ResNet-50 as the model architecture. As a modification, they removed the softmax and fully connected layer from the end. They used 2048-dimensional image feature vector as the flattened output, which they referred to as an image feature extraction network. They received 85.8% average accuracy in single image classification, and the highest multimodal network accuracy was 86.6%. The multimodal network fails to acquire higher accuracy. Serban et al. [18] implemented a convolutional neural network to facilitate automatic diagnosis of skin cancer. The dataset consists of 1000 images collected from International Skin Imaging Collaboration and PH2 databases. There are two classes, benign tumours and skin malignant lesions, each having 500 images. Then the dataset was fed to the neural network. The proposed solution acquired 80.52% accuracy on the dataset.The dataset has only two classes and the accuracy can be improved.

Hosny et. al [19] classified three different types of skin cancer that are melanoma, common nevus and atypical nevus. They modified the AlexNet architecture. When the model was trained with original images, they got 80% accuracy. The accuracy was 98.61% after using augmented images. They used 200 images from ph2 dataset which became 11000 after applying augmentation. However, there are 11000 images after augmentation while the original dataset has three classes with 200 images in total. Ensaf et al. [20] have used transfer learning as a part of deep convolutional networks to classify enhanced skin lesions. They used HAM10000 dataset. In the pre-processing step, the

dataset was cleaned, downsampled, split and then augmented. As model architecture, DenseNet-121 and MobileNet architecture was used. Both CNN models were pre-trained. After evaluation, DenseNet-121 and MobileNet achieved 71.9% and 82.6% testing accuracy, respectively on the unbalanced dataset. Whereas, they reached 92.7% and 91.2% accuracy, respectively on the balanced dataset. It was an impressive improvement.

Nahata et. al [21] proposed skin cancer detection model using deep learning that can classify the types of skin cancer. They used similar seven classes from ISIC 2018 and 2019 dataset. Transfer learning was used to train the CNN. For this, InceptionV3, ResNet50, VGG16, MobileNet and InceptionResnet was used. Among all these models, InceptionResnet achieved 91% accuracy. The dataset had 35,348 images with seven classes. Improvement in the model architecture can give more accurate prediction result. Ulzii et al. [22] proposed a way to classify skin cancer by using ECOC SVM classifier with pre-trained AlexNet architecture. They collected the dataset through searching on different search engines available online. The dataset consists of 3753 images with 4 classes Actinic Keratoses (897), Melanoma (958), Squamous cell carcinoma (977), Basal cell carcinoma (921). For the model architecture, pre-trained AlexNet model was used to extract the training features. The classifier was Error-Correcting Output Codes (ECOC) SVM. After evaluation, they were able to acquire 94.2% accuracy on multiclass classification. Further improvement can increase the accuracy of their model.

The earlier research was mostly focused on melanoma, but research focusing on various skin cancer types is scarce. Our focus is on improving the dataset and building a model that can classify different types of skin cancer.

3 METHODOLOGY

In this section, we have described the overall workflow of our proposed method. At first, we have collected a dataset then we preprocessed the dataset with the augmentation method. Finally, we proposed a deep learning-based model to detect skin cancer, as shown in Figure 2.

3.1 Dataset

Our data collection process was a challenging task. We collect 21 images from the Department of Dermatology, Dhaka Medical College (DMC), Dhaka, Bangladesh. Other 779 images acquired from `www.dermnet.com`, which is the largest independent photo dermatology source dedicated to online medical education. Our dataset contains four different classes of Skin Cancer, including Actinic Keratosis (AK), Basal Cell Carcinoma (BCC), Malignant Melanoma (MM) and Squamous Cell Carcinoma (SCC). Figure 1 shows a sample of our dataset on skin cancer. After collecting the dataset, we resized our dataset because all the images were not the same size. So, the images were pre-processed by resizing them into 224×224 pixel as it gives a better result. For our deep learning approach, the number of images was not enough for the model's learning.

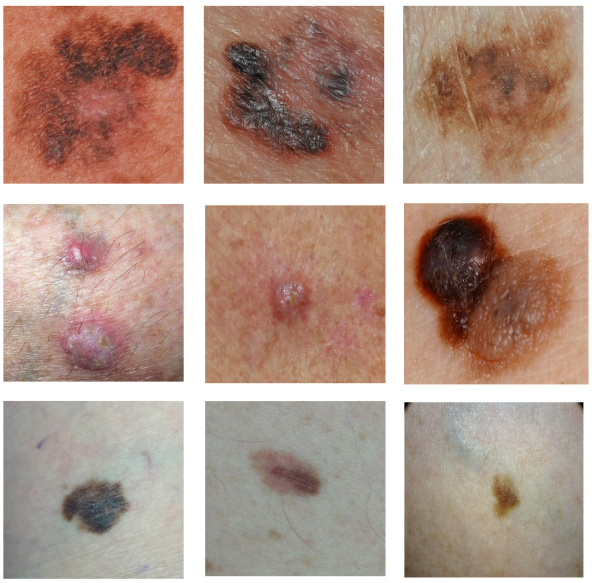


Figure 1: A sample of our collected dataset.

Table 1: Image Augmentation Parameters.

Transformation Types	Descriptions
Rotation	Rotate an angle ranging from 0 to 360 degrees at random.
Flipping	0 (Without) or 1 (With) flipping
Re-scaling	With a scale factor ranging from 0.3 to 0.8, at random
Shading	From the middle to the edges, the image brightness is reduced.
Translation	Adding a given value to the x-axis and y-axis shifts the image in coordinate space.
Shearing	Shifting one portion of the image in the manner of a parallelogram.

3.2 Pre-processing and Augmentation

We used different augmentation techniques to improve the dataset size using rotation, flipping, shading, translation, shearing and scaling. We applied 0.2 as shifting, shear and zoom range. Rotation transforms a source image by rotating clockwise or counterclockwise by some number of degrees randomly. Flipping means the rotation of an image in horizontal or vertical axis. Shading is done by different hue values denoting the shade of colors in the image. Translation means moving the

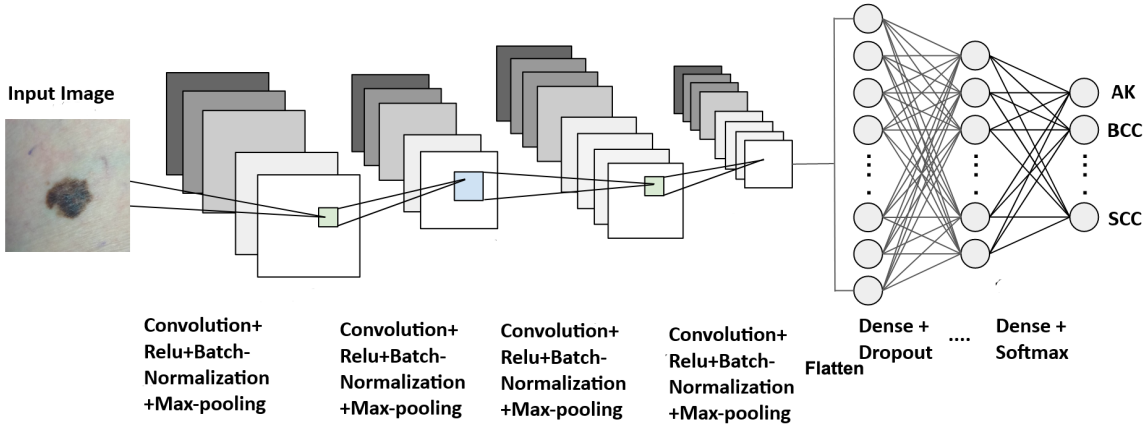


Figure 2: The architecture of propose deep CNN Model for Skin Cancer detection and classification.

image along to either x-axis or y-axis. Shearing means shifting one part of an image to any parallel side. Resizing an image to a given size is scaling. Table 1 contains a summary of the augmentation parameters. After applying all the transformations, each image normalized. After augmenting our dataset, we received in a total of 5600 images. Then we split it into training and testing sets. In our training phase, 4480 images (80% of the dataset) used for model learning means training and validation. We kept 1120 images (20% of the dataset) for testing and evaluating the model.

3.3 Proposed Model

Different types of deep learning architectures are commonly used for image and video classification, while recent methods focusing on deep CNN models for identifying and classifying 2D and 3D images [23] [24] [25] [26]. Due to memory limitations, it is computationally expensive to learn many trainable parameters and arithmetic operations [27]. It motives us to work on a deep learning-based method to detect and classify skin cancer. Our proposed model used an input layer corresponding to 224×224 input image with three input channels. In this architecture, we used the first convolutional layer from the input image with two and used ReLU activation function to activate every convolution layer. We used batch normalization, which normalizes the feature extraction and reduces data variance. It is used after ReLU (1) activation function and before max-pooling. It reduces the spatial dimensionality of the extracted feature maps. Then we used the output filter size as 16 with 7×7 kernel size followed by 2×2 size max-pooling layer with stride 2. Then added three convolutional blocks, stacked over the first block, each having a 3×3 kernel size with 32, 64, 128 and filters sequentially with ReLU activation function with corresponding batch-normalization and 2×2 max-pooling layers with stride 2. Table 2 shows the summary of the proposed model with layers, configuration, parameters and output shape.

$$RELU(x) = MAX(0, X) \tag{1}$$

The outputs from these layers are flattened and connected with three fully connected layers: dense layers and dropout layers. We used 25% of first two dropout layers and 50% used for last dropout layer and the first two dense layers used in 256 and another dense layer used in 128, followed by Softmax (2) output layer activation function with probability images for each of the class.

$$Softmax((x_i)) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2}$$

It must be designed with the appropriate loss function and optimizer before being used for training. Adam was used as an optimizer because it has a lower computational cost, uses less memory, and is unaffected by gradient re-scaling. This addresses concerns like large data sets, hyper-parameters, noisy data, insufficient gradients, and non-stationary problems that involve fine-tuning. We used categorical cross-entropy as a loss function, which is defined as an n-dimensional vector.

3.4 Pre-train Models

GoogleNet [28], and MobileNet [29] are one of the most popular pre-train approaches in Deep Learning and Transfer Learning that is being used by researchers broadly [31] [32]. We also used these two pre-train models that consist of 48 layers and 28 layers, respectively. These models are trained on large ImageNet dataset with 1000 classes and perform well. GoogleNet [28] has learned multi-level feature representation for a wide range of images. We use it with pre-train weight, and our images were 299×299. The last layer consists of an GlobalAverage, a Pooling2D layer and two Dense layers. SGD optimizer was applied and Binary Cross Entropy was used as the loss function [30]. Another image classification model is MobileNet [29]. We use

Table 2: The summary of the proposed model with layers, configuration and output shape.

Layers	Configuration	Output Shape
Conv2D	$224 \times 224 \times 16$; kernel size: 7×7; stride: 2; activation function: RELU	$224 \times 224 \times 16$
BatchNormalization	-	$224 \times 224 \times 16$
Max-pooling2D	Kernel size: 2×2; stride: 2;	$112 \times 112 \times 32$
Conv2D	$112 \times 112 \times 32$; kernel size: 3×3; stride: 2; activation function: RELU	$112 \times 112 \times 32$
BatchNormalization	-	$112 \times 112 \times 32$
Max-pooling2D	Kernel size: 2×2; stride: 2;	$56 \times 56 \times 64$
Conv2D	$56 \times 56 \times 64$; kernel size: 3×3; stride: 1; activation function: RELU	$56 \times 56 \times 64$
BatchNormalization	-	$56 \times 56 \times 64$
Max-pooling2D	Kernel size: 2×2; stride: 2;	$28 \times 28 \times 128$
Conv2D	$28 \times 28 \times 128$; kernel size: 3×3; stride: 2; activation function: RELU	$28 \times 28 \times 128$
BatchNormalization	-	$28 \times 28 \times 128$
Max-pooling2D	Kernel size: 2×2; stride: 2;	$14 \times 14 \times 256$
Flatten	-	4096
Dense	-	256
Dropout	0.25	256
Dense	-	128
Dropout	0.25	128
Dense	-	64
Dropout	0.50	64
Dense	activation: Softmax; unit: 4	4

this with pre-train weight and 224×224 input images. Here, the last layer consists a Dense layer with softmax activation function. SGD optimizer was applied and Binary Cross Entropy was used as the loss function. After that, we use our augmented images to train and test those pre-trained models. We used a Dense layer with softmax activation function as the last layer. Adam optimizer was applied and Categorical Cross Entropy was used as the loss function. These pre-trained models decreased the requirement of computational power and saved time for training the model. Table 3 shows the two pre-train model's training details and our proposed model.

3.5 Evaluation Metrics

For measuring our model's performance, we evaluated it to observe which model gives the highest accuracy by predicting the sample data [33] [34]. We calculated accuracy, precision, recall, Specificity, Negative Predictive Rate and False Discovery Rate by using the True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN) values. We also observed the confusion matrix for predicting each class [35]. Following equations, 3-8 were used for evaluating the results.

Table 3: Training details of three deep learning models (two pre-train models and our proposed model).

Training Details	GoogleNet	MobileNet	Proposed model
Data Augmentation	Yes	Yes	Yes
Transfer Learning	Yes	Yes	No
Last layer	GlobalAveragePooling2D Dense (1024, activation = relu) Dense (4, activation = sigmoid)	Dense (4, activation = softmax)	Dense (4, activation = softmax)
Feature Extraction Enabled	Yes	No	Yes
Classification Enabled	Yes	Yes	Yes
Optimizer	SGD	SGD	ADAM
Loss Function	Binary Cross-Entropy	Binary Cross-Entropy	Categorical Cross-Entropy
Number of Parameters	23,909,160	2,259,265	14,050,501
Number of Trainable Parameters	23,874,727	2,225,153	13,142,281

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{3}$$

$$Recall/Sensitivity = \frac{(TP)}{(FN+TP)} \tag{4}$$

$$Precision = \frac{(TP)}{(TP+FP)} \tag{5}$$

$$NegativePredictiveValue = \frac{(TN)}{(TN+FN)} \tag{6}$$

$$FalseDiscoveryRate = \frac{(FP)}{FP+TP} \tag{7}$$

$$Specificity = \frac{(TN)}{(FP+TN)} \tag{8}$$

4 RESULTS AND DISCUSSION

4.1 Experimental Setup

To train our proposed model, we used Google Colab GPU platform using python environment. For training, We compiled our proposed model with ADAM optimizer with a learning rate of 0.001, and the batch size was 32. After that, we train our model for 50 epochs with 'categorical cross-entropy' as the loss function. We measured the model train and validation graph by observing the loss function. We used pre-trained models that were fine-tuned to adjust some parameters for training.

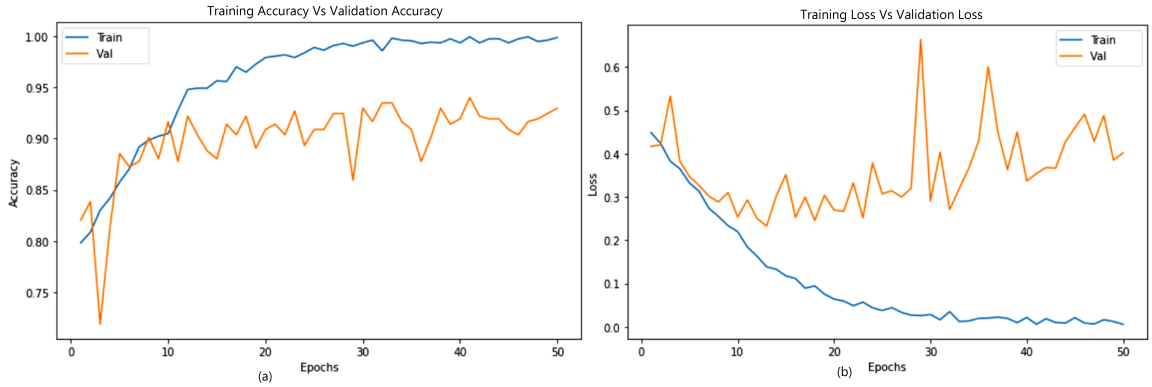


Figure 3: (a) Accuracy, and (b) Loss Graph of our proposed model.

4.2 Results

Table 4 represents the confusion matrix of our proposed model. It shows a tabular way of the performance of our model. Each entry in a confusion matrix denotes the number of predictions made by the model where it classified the classes correctly or incorrectly. Here the rows and columns indicate actual labels and predicted labels respectively. Depending on the number of classes, The size of our confusion matrix became 4×4. Table 5 shows the performance evaluation that includes Precision, Recall, Accuracy, Negative Predictive Value and False Discovery Rate for four different skin cancer classes of our proposed model. Equation 3 to 7 was used for calculating these values. The different evaluation metrics indicate the overall performance of our model. We have got the highest accuracy 97.23% in Basal Cell Carcinoma (BCC). Basal Cell Carcinoma acquired the highest values in other metrics such as Precision, Recall and Negative Predictive Value.

Table 4: The Confusion Matrix (CM) of our proposed model.

Classes	AK	BCC	MM	SCC
AK	**258**	5	9	8
BCC	3	**269**	3	5
MM	15	2	**256**	7
SCC	9	13	11	**247**

Table 5: Performance Evaluation of our proposed Model.

Types	Precision	Recall	Accuracy	NPV	F-Rate
AK	0.92	0.90	95.63%	0.97	0.08
BCC	0.99	0.93	97.23%	0.98	0.04
MM	0.91	0.91	95.80%	0.97	0.09
SCC	0.88	0.92	95.27%	0.98	0.12

Figure 3 (a) describes the training and validation accuracy of our model. Also the training and validation loss is shown graphically in Fig. 3 (b). Here, the blue line represents the training performance, and the orange line represents the validation performance. After training, we noticed there was a slight overfitting pattern. This happened due to the new dataset. The different classes had very similar characteristics which led the training to occur in an overfitting pattern.

Table 6: Different training options with performance of proposed model.

Optimizers	Loss Function	Pre	Rec	Acc
SGD	MSE	85.42%	81.65%	89.97%
SGD	cross entrophy	85.74%	85.36%	90.68%
ADAM	MSE	88.55%	87.81%	92.32%
ADAM	**cross entrophy**	**91.96%**	**91.97%**	**95.98%**

To get the best among all, we used SGD and ADAM optimizers with MSE and cross entropy loss functions. When we chose ADAM as the optimizer with cross entropy loss function, we got 95.98% accuracy with 91.96% precision and 91.97% recall. Table 6 depicts the performance of proposed model with different optimizers and loss functions.

4.3 K-Fold Cross Validation

K-fold cross validation is used to know how accurately the model will predict if a new test set is used. Here k is the number of folds where each fold will be used as a test set. In this case, to evaluate the proposed model on our dataset, we used 5-fold cross validation. The first fold was used as the test set in the first iteration. Sequentially the next folds are used as test set in next four iterations. As we have 5600 images, each fold consists of 1120 images.

The confusion matrics for each fold is shown in figure 4. The confusion matrics for each fold is shown in figure 4. Here in each confusion matrix, each row is for actual values and column is for the predicted values of the model. True positive values refers to the prediction of the existence of skin cancer when it is actually

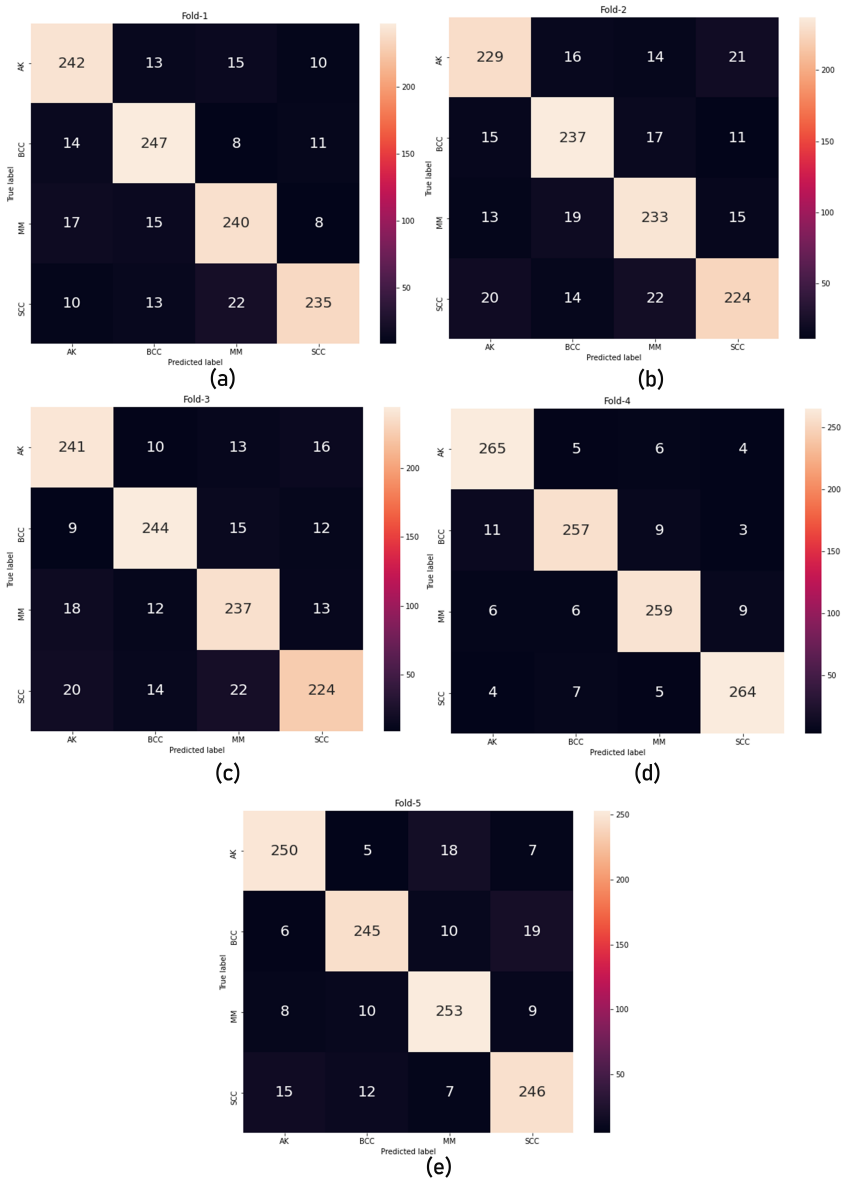


Figure 4: Confusion matrix of 5-fold cross validation process for each fold: (a) fold 1, (b) fold 2, (c) fold 3, (d) fold 4, and (e) fold 5.

present. In the first fold, the true positive values are 242, 247, 240, 235 for Actinic Keratosis, Basal Cell Carcinoma, Malignant Melanoma and Squamous Cell Carcinoma respectively. Similarly, the true positive values for next four folds are (229, 237, 233, 224), (241, 244, 237, 224), (265, 257, 259, 264), (250, 245, 253, 246), respectively. So for each iteration, we can see the model prediction is quite well.

We also measured precision and recall for each fold. Folds were shuffled 5 times and the measures are given in table 7. After observing these values for each fold, we got an average of 87.08% and 87.16% precision and recall values respectively. The average accuracy of 5 folds is 93.50%. From this measurements, we can say that overfitting is present.

Table 7: Measurement of 5-fold cross validation of this dataset.

Fold	Precision (%)	Recall (%)	Accuracy (%)
Fold-1	86.03	86.25	93.04
Fold-2	82.53	82.47	91.21
Fold-3	84.59	84.75	92.23
Fold-4	93.48	93.25	96.65
Fold-5	88.76	89.09	94.37
Average	**87.08**	**87.16**	**93.50**

4.4 Comparison

The performance comparison among the Deep Learning models was made using Sensitivity, Specificity, Pre-

Table 8: Comparison between Deep Learning Models

Dataset type	Models	Sensitivity	Specificity	Precision	Accuracy
Without Augmentation	GoogleNet [28]	79.81%	84.23%	79.57%	84.66%
	MobileNet [29]	81.09%	84.97%	81.55%	85.48%
	Proposed Model	**83.18%**	**88.93%**	**84.72%**	**87.36%**
With Augmentation	GoogleNet [28]	90.75%	94.91%	88.38%	94.21%
	MobileNet [29]	90.23%	95.66%	90.73%	94.85%
	Proposed Model	**91.97%**	**97.33%**	**91.96%**	**95.98%**

Table 9: Comparison our result with the state of the arts.

Approaches	Methods	Dataset Size	Classes	Accuracy
Moussa et al. [39]	k-NN	15	2	89%
Dubal et al. [36]	NN	463	6	76.90%
Alquran et al. [37]	SVM	11	1	92.1%
Victor et al. [38]	KNN	1000	2	93.70%
Milton et al. [40]	DNN	2000	7	76%
Ulzii et al. [22]	CNN	3753	4	94.2%
Proposed model	**Deep CNN**	**5600**	**4**	**95.98%**

cision and Accuracy. It shows that our proposed model performs better than pre-train models, GoogleNet and MobileNet. Table 8 demonstrates the results. At first, the pre-trained models and our proposed models are trained with images that are not augmented. Here we can see that our model performed better than other two models with an accuracy of 87.36%. To improve the accuracy, in the next approach, we trained the model with augmented images. This time the accuracy of our proposed model is 95.98%.

Table 9 shows the comparison of our model with previous works. After applying augmentation, our used dataset was the most enriched of all the previous works done before. Also, our proposed model outperformed all the previous researches in classifying different types of Skin Cancer. Various Neural Networks, SVM, KNN, and Convolutional Neural Networks have been used among the earlier works. The size and the feature of the dataset also had varieties. Milton et al. [40] had worked with the most established dataset with 2000 images in 2 classes. The work focused on multiple pre-train networks and was able to secure at best 76% accuracy. On the other hand, Victor et al. [38] used 1000 images with binary classes and used classical machine learning algorithm KNN and SVM. They were able to secure 93.70% on their work. Our work surpassed previous results by both the dataset and accuracy.

5 CONCLUSIONS

In this paper, we have collected a dataset of four types of skin cancer and used augmentation techniques to increase our dataset. We proposed a deep learning-based method to classify skin cancer. We used some regularization methods like batch normalization to avoid overfitting and used convolution, max pooling, dropout, and fully connected layers. Our proposed model achieved 96.98% accuracy for four types of skin cancer that is the state of arts. This accuracy is higher than two other pre-train models, GoogleNet and MobileNet. Five-fold cross-validation is used to verify the proposed model with this dataset. In summation, our proposed model provided better comparable performance to the existing state-of-the-art methods in terms of accuracy, precision, recall, sensitivity, and specificity. It is also simple and lightweight architecture than other models. In future, a light architecture can be designed without compromising the accuracy to detect skin cancer by minimizing computational complexity.

6 REFERENCES

[1] Craythorne, E. and Al-Niami, F., 2017. Skin cancer. Medicine, 45(7), pp.431-434.

[2] https://www.skincancer.org/skin-cancer-information/squamous-cell-carcinoma/ [Last visited: 12-01-2021]

[3] Siegel, J.A., Korgavkar, K. and Weinstock, M.A., 2017. Current perspective on actinic keratosis: a review. British Journal of Dermatology, 177(2), pp.350-358.

[4] Garbe, C., Peris, K., Hauschild, A., Saiag, P., Middleton, M., Bastholt, L., Grob, J.J., Malvehy, J., Newton-Bishop, J., Stratigos, A.J. and Pehamberger, H., 2016. Diagnosis and treatment of melanoma. European Journal of Cancer, 63, pp.201-217.

[5] https://cancerstatisticscenter.cancer.org [last visited: 12-01-2021]

[6] Vocaturo, E., Perna, D. and Zumpano, E., 2019, November. Machine Learning Techniques for Automated Melanoma Detection. In 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 2310-2317). IEEE.

[7] Ansari, U.B. and Sarode, T., 2017. Skin cancer detection using image processing. Int Res J Eng Technol, 4(4), pp.2875-2881.

[8] Stefanczyk, M. and Bochenski, T., 2020. Mixing deep learning with classical vision for object recognition. Journal of WSCG, Vol.28, No.1-2, 2020.

[9] Khan, M.Q., Hussain, A., Rehman, S.U., Khan, U., Maqsood, M., Mehmood, K. and Khan, M.A., 2019. Classification of Melanoma and Nevus in Digital Images for Diagnosis of Skin Cancer. IEEE Access, 7, pp.90132-90144.

[10] Gour, N. and Khanna, P., 2021. Multi-class multi-label ophthalmological disease detection using transfer learning based convolutional neural network. Biomedical Signal Processing and Control, 66, p.102329.

[11] Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M. and Thrun, S., 2017. Dermatologist-level classification of skin cancer with deep neural networks. nature, 542(7639), pp.115-118.

[12] Junayed, M.S., Sakib, A.N.M., Anjum, N., Islam, M.B. and Jeny, A.A., 2020, December. EczemaNet: A Deep CNN-based Eczema Diseases Classification. In 2020 IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS) (pp. 174-179). IEEE.

[13] Wu, Z., Zhao, S., Peng, Y., He, X., Zhao, X., Huang, K., Wu, X., Fan, W., Li, F., Chen, M. and Li, J., 2019. Studies on different CNN algorithms for face skin disease classification based on clinical images. IEEE Access, 7, pp.66505-66511.

[14] Garg, N., Sharma, V. and Kaur, P., 2018. Melanoma skin cancer detection using image processing. In Sensors and Image Processing (pp. 111-119). Springer, Singapore.

[15] Sreelatha, T., Subramanyam, M.V. and Prasad, M.G., 2019. Early detection of skin cancer using melanoma segmentation technique. Journal of medical systems, 43(7), p.190.

[16] Pham, T.C., Luong, C.M., Visani, M. and Hoang, V.D., 2018, March. Deep CNN and data augmentation for skin lesion classification. In Asian Conference on Intelligent Information and Database Systems (pp. 573-582). Springer, Cham.

[17] Yap, J., Yolland, W. and Tschandl, P., 2018. Multimodal skin lesion classification using deep learning. Experimental dermatology, 27(11), pp.1261-1267.

[18] Jianu, S.R.S., Ichim, L. and Popescu, D., 2019, March. Automatic diagnosis of skin cancer using neural networks. In 2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE) (pp. 1-4). IEEE.

[19] Hosny, K.M., Kassem, M.A. and Foaud, M.M., 2018, December. Skin cancer classification using deep learning and transfer learning. In 2018 9th Cairo International Biomedical Engineering Conference (CIBEC) (pp. 90-93). IEEE.

[20] Mohamed, E.H. and El-Behaidy, W.H., 2019, December. Enhanced Skin Lesions Classification Using Deep Convolutional Networks. In 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS) (pp. 180-188). IEEE.

[21] Nahata, H. and Singh, S.P., 2020. Deep learning solutions for skin cancer detection and diagnosis. In Machine Learning with Health Care Perspective (pp. 159-182). Springer, Cham.

[22] Dorj, U.O., Lee, K.K., Choi, J.Y. and Lee, M., 2018. The skin cancer classification using deep convolutional neural network. Multimedia Tools and Applications, 77(8), pp.9909-9924.

[23] Waheed, A., Goyal, M., Gupta, D., Khanna, A., Hassanien, A.E. and Pandey, H.M., 2020. An optimized dense convolutional neural network model for disease recognition and classification in corn leaf. Computers and Electronics in Agriculture, 175, p.105456.

[24] Junayed, M.S., Jeny, A.A., Atik, S.T., Neehal, N., Karim, A., Azam, S. and Shanmugam, B., 2019, July. AcneNet-A deep CNN based classification approach for acne classes. In 2019 12th International Conference on Information & Communication Technology and System (ICTS) (pp. 203-208). IEEE.

[25] Anwar, S.M., Majid, M., Qayyum, A., Awais, M., Alnowami, M. and Khan, M.K., 2018. Medical image analysis using convolutional neural networks: a review. Journal of medical systems, 42(11), pp.1-13.

[26] Yu, H., Yang, L.T., Zhang, Q., Armstrong, D. and Deen, M.J., 2021. Convolutional neural networks for medical image analysis: state-of-the-art, comparisons, improvement and perspectives. Neurocomputing.

[27] Zhang, Q., Bai, C., Liu, Z., Yang, L.T., Yu, H., Zhao, J. and Yuan, H., 2020. A GPU-based residual network for medical image classification

in smart medicine. Information Sciences, 536, pp.91-100.

[28] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[29] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[30] Kumar, A., Sarkar, S. and Pradhan, C., 2020. Malaria Disease Detection Using CNN Technique with SGD, RMSprop and ADAM Optimizers. In Deep Learning Techniques for Biomedical and Health Informatics (pp. 211-230). Springer, Cham.

[31] Gour, N. and Khanna, P., 2021. Multi-class multi-label ophthalmological disease detection using transfer learning based convolutional neural network. Biomedical Signal Processing and Control, 66, p.102329.

[32] Zwettler, G.A., Holmes III, D.R. and Backfrieder, W., 2020. Strategies for Training Deep Learning Models in Medical Domains with Small Reference Datasets. Journal of WSCG. 2020, vol. 28, no. 1-2, p. 37-46.

[33] Junayed, M.S., Jeny, A.A., Neehal, N., Ahmed, E. and Hossain, S.A., 2018, December. Incept-N: A Convolutional Neural Network Based Classification Approach for Predicting Nationality from Facial Features. In International Conference on Recent Trends in Image Processing and Pattern Recognition (pp. 466-475). Springer, Singapore.

[34] Macatangay, J.M.A., Ruiz Jr, C.R. and Usatine, R.P., 2017. A primary morphological classifier for skin lesion images. 25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Visionin co-operation with EUROGRAPHICS Association, p. 55-64.

[35] Jeny, A.A., Junayed, M.S., Ahmed, I., Habib, M.T. and Rahman, M.R., 2019, December. FoNet-Local Food Recognition Using Deep Residual Neural Networks. In 2019 International Conference on Information Technology (ICIT) (pp. 184-189). IEEE.

[36] Dubal, P., Bhatt, S., Joglekar, C. and Patil, S., 2017, November. Skin cancer detection and classification. In 2017 6th international conference on electrical engineering and informatics (ICEEI) (pp. 1-6). IEEE.

[37] Alquran, H., Qasmieh, I.A., Alqudah, A.M., Alhammouri, S., Alawneh, E., Abughazaleh, A. and Hasayen, F., 2017, October. The melanoma skin cancer detection and classification using support vector machine. In 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT) (pp. 1-5). IEEE.

[38] Victor, A. and Ghalib, M., 2017. Automatic detection and classification of skin cancer. International Journal of Intelligent Engineering and Systems, 10(3), pp.444-451.

[39] Moussa, R., Gerges, F., Salem, C., Akiki, R., Falou, O. and Azar, D., 2016, October. Computer-aided detection of Melanoma using geometric features. In 2016 3rd Middle East Conference on Biomedical Engineering (MECBME) (pp. 125-128). IEEE.

[40] Milton, M.A.A., 2019. Automated skin lesion classification using ensemble of deep neural networks in isic 2018: Skin lesion analysis towards melanoma detection challenge. arXiv preprint arXiv:1901.10802.

A Mobile Augmented Reality Application For Simulating Claude Monet's Impressionistic Art Style

Neil Patrick Del Gallego
De La Salle University
Taft Avenue, Malate
1004 Metro Manila,
Philippines
neil.delgallego@dlsu.edu.ph

Cedric Lance Viaje
De La Salle University
Taft Avenue, Malate
1004 Metro Manila,
Philippines
cedric_viaje@dlsu.edu.ph

Michael Ryan Gerra-Clarin
De La Salle University
Taft Avenue, Malate
1004 Metro Manila,
Philippines
michael_gerra-clarin@dlsu.edu.ph

John Marvic Roque
De La Salle University
Taft Avenue, Malate
1004 Metro Manila,
Philippines
john_marvic_roque@dlsu.edu.ph

Gary Steven Non
De La Salle University
Taft Avenue, Malate
1004 Metro Manila,
Philippines
gary_non@dlsu.edu.ph

Jesin Jarod Martinez
De La Salle University
Taft Avenue, Malate
1004 Metro Manila,
Philippines
jesin_martinez@dlsu.edu.ph

Jose Antonio Gana
De La Salle University
Taft Avenue, Malate
1004 Metro Manila,
Philippines
jose_antonio_gana@dlsu.edu.ph

ABSTRACT

In this study, we showcase a mobile augmented reality application where a user places various 3D models in a tabletop scene. The scene is captured and then rendered as Claude Monet's impressionistic art style. One possible use case for this application is to demonstrate the behavior of the impressionistic art style of Claude Monet, by applying this to tabletop scenes, which can be useful especially for art students. This allows the user to create their own "still life" composition and study how the scene is painted. Our proposed framework is composed of three steps. The system first identifies the context of the tabletop scene, through GIST descriptors, which are used as features to identify the color palette to be used for painting. Our application supports three different color palettes, representing different eras of Monet's work. The second step performs color mixing of two different colors in the chosen palette. The last step involves applying a three-stage brush stroke algorithm where the image is rendered with a customized brush stroke pattern applied in each stage. While deep learning techniques are already capable of performing style transfer from paintings to real-world images, such as the success of CycleGAN, results show that our proposed framework achieves comparable performance to deep learning style transfer methods on tabletop scenes.

Keywords

augmented reality, mobile devices, image filter, image stylization, style transfer, painterly rendering

1 INTRODUCTION

Augmented reality is an interactive experience wherein digital objects are placed on the physical environment [BCL15]. This study proposes a mobile augmented reality application that allows the user to capture a scene with virtual 3D models and then applies a rendering technique that "paints" the scene using Claude Monet's impressionistic art style (Figure 1). A specific use case for this application is to demonstrate the visuals of Claude Monet's impressionistic art style, which can be useful to art students who are learning the fundamentals of painting. Using the mobile AR application, the user can place virtual 3D models on top of other tabletop objects which allows the user to compose a still life scene containing virtual models that are not readily available (e.g. A virtual dragon model placed with other tabletop objects, such as drinking glass and basket of fruits). These 3D models can coexist within the same scene in real-time.

Recently, deep learning methods are used to generate images painted with a certain art style [GEB16]. Image-to-image translation using generative adversarial networks was proposed, such as the release of Pix2Pix [IZZE17]. Then the release of CycleGAN introduced a new class of image-to-image translation methods that

Corresponding author email: neil.delgallego@dlsu.edu.ph. The source code is available at Github: `https://github.com/NeilDG/AR-Impress-Me-Version-2`

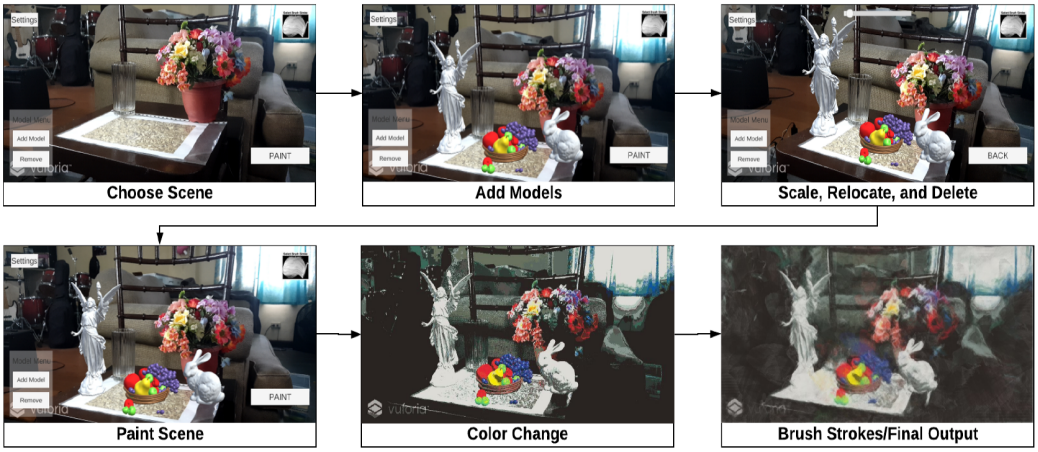


Figure 1: The proposed application. The user assembles a tabletop scene and then places one or more virtual 3D models. The scene is then captured and processed, which produces an art style similar to Claude Monet's paintings, using a pre-defined color palette and brushstroke settings consistent with an impressionistic art style.

Figure 2: Example of virtual models that can be placed in a tabletop scene. Standard 3D models commonly used for computer graphics applications are shown. From left to right: Lucy, Stanford Bunny, Stanford Armadillo, Utah Teapot. Our application can also load other custom-made 3D models.

do not need paired data [ZPIE17]. CycleGAN has been used for style transfer approaches and shows that it is comparable to other style transfer methods such as the method proposed by Gatys et.al (2016) [GEB16].

Early work showed a style of processing images and video with an impressionistic effect that looks like it was hand-painted [Lit97]. An optical flow field technique was applied that determines the brushstroke pattern frame by frame. Randomness is applied to the brush stroke attributes, such as length, color, and orientation, to enhance the hand-touched look. Similarly, a method proposed by Hertzmann (1998) paints an image with a series of spline brush strokes [Her98]. We observed that this method is effective in simulating impressionistic brush strokes and applied the same concept in the application. Other early brush stroke techniques were observed in literature [Hae90, HE04, KS04, KS06, CS07].

While deep learning methods are becoming the trend for artistic painting and non-photorealistic rendering, we observe that these methods are not tailored for images that consist of augmented 3D models, such as the case of using an AR application for painting a scene. We later show in our results that existing style transfer methods generate impressionistic images where the 3D models stand out unnaturally and do not blend properly with the overall image composition. Furthermore, our proposed framework is grounded on an analytical study of Monet's artworks, unlike other deep learning approaches that completely rely on individual paintings as training samples.

We organized this paper as follows: related work, concepts on impressionism, architectural framework, then lastly, discussion of results and conclusion.

2 RELATED WORK

2.1 Image Stylization

Image stylization is the process of taking an image as input, and producing a stylized version of it [RAGS01]. The contents of the image should remain the same. To achieve an artistic stylized image: computer vision techniques and machine learning are mostly used to transform the target images and their pixels [KCWI12]. We further discuss different image stylization methods under two categories, algorithmic and learning-based image stylization.

Algorithmic approaches in image stylization are those that rely on apriori information and typically render images in a deterministic or sometimes stochastic manner. Region-based techniques, which are typically applied to cartoon shading styles [CRH05, WXSC04], simulating stained glass [Mou03], simulating cubist painting [CH03], involve image segmentation as a pre-processing step [KCWI12]. Image stylization techniques that are applied to fine details of an image [SLKD16, KD08, KKD09b, STD+16] typically involve edge-preserving smoothing approaches, local image statistics, and approaches based on morphological filtering [KCWI12].

Learning-based approaches or machine learning approaches in image stylization rely on painting examples as training input. Notable examples in recent years are deep learning style transfer techniques [GEB16, MOT15]. Network architectures that deal with image-to-image translation, paired [IZZE17] or unpaired [ZPIE17], are shown to work on different applications, including style transfer. Other loss terms aside from L1 or L2 norms are employed to further improve the performance of previous style transfer approaches. For example, the perceptual loss function is applied during training, wherein the difference of feature maps, between the input image and the reference image are minimized [JAL16]. Feature maps are typically extracted from the convolutional layers of a pre-trained VGG-19 network [SZ14]. Another loss term was proposed in the works of Ulyanov, Vedaldi, and Lempitsky (2017)[UVL17], that improves the diversity of image stylization through the use of Julesz ensemble [Jul81], which is based on a concept that textures are treated as a family of visual patterns that share local statistical regularities.

2.2 Stroke-Based Rendering

Vanderhaeghe and Collomose (2013) defined stroke-based rendering (SBR) as a process of synthesizing artwork by combining rendering marks (such as lines, brush strokes, or even larger primitives such as tiles) to a digital canvas [VC13]. The work of Litwinowicz (1997) [Lit97] is among the earliest methods for stroke-based rendering where impressionistic paintings are simulated on video input. Using multiple frames, the brush stroke and intensity are determined based on the movement of pixels identified using optical flow [TZ99]. The video abstraction method proposed by Kyprianidis, Kang, and Döllner (2009) [KKD09a] were inspired from the Kuwahara filter [KHEK76], which is a non-linear smoothing filter typically used for image processing that applies a blurring effect while preserving the edges. The modified filter generates a painting-like effect while preserving shape boundaries which is observed to work well in cartoon illustrations and oil paintings. More recently, deep learning generative models were utilized for SBR, such as a neural model capable of simulating learned hand strokes (NeuralPainter) [Nak19], and a neural painting environment where a position and a brush encoder are trained for learning different brush strokes for simulating drawings (StrokeNet) [ZJH18].

2.3 Augmented Reality Art

AR commonly applied in art galleries, exhibits, or cultural heritage, provides features for augmenting artificial objects that can be used to enhance various art forms [Mar18, Ger18]. Sketch-based and virtual sculpting applications were also observed to use AR [BC20]. Similarly, interactive modeling applications make use of AR, such as MagicToon, which is an interactive modeling tool for mobile AR that allows a user to augment 3D cartoon scenes based on hand-drawn sketches on paper [FYX17].

3 MONET'S IMPRESSIONISTIC ART STYLE: OVERVIEW

Impressionism is an art movement that started in the late 19th century where paintings are characterized as sketch-like, while others praised it for its depiction of modern life [Sam04]. Notable impressionist painters were Claude Monet, Edgar Degas, Camille Pissarro, and Pierre-Auguste Renoir [Bre74]. An impressionistic painting captures the effects of sunlight, gray and dark tones produced as a product of mixing complementary colors. Pure impressionism is often characterized by the lack of black paint. Rather than neutral white, grays, and blacks, painters typically draw shadows and tones in color. Outdoor paintings are described as fresh, due to the use of bright and light brush colors [Sam04, FMSGA18]. In this study, we primarily focused on the works of Claude Monet, where his works accurately depict the descriptions described earlier.

To the best of our knowledge, there are limited studies that describe Claude Monet's work. A book was written by Forge (1995) [For95], and another book was written by Wildenstein (1978) [MW78] that described the life of Monet and his works in chronological order. Concepts discussed in the succeeding sections are supported by the book of Forge (1995) and serves as the main reference for designing our framework.

3.1 Color Palettes

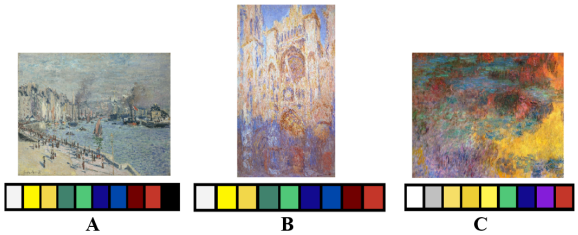


Figure 3: Three color palettes identified in the works of Monet, and corresponding paintings that contain a specified color palette. A: Pre-1886 (Port of Le Havre). B: Post-1886 (Rouen Cathedral: The Portal). C: Final years (Water Lily Pond, Evening).

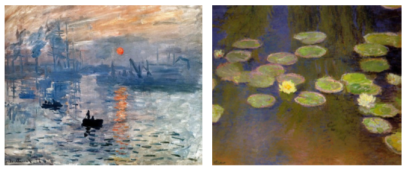
Claude Monet used three color palettes in his paintings. The first color palette was used in his pre-1886 works, which consists of lead white, chrome yellow, cadmium yellow, viridian green, emerald green, french ultramarine, cobalt blue, alizarin crimson, vermilion, and ivory black. After the year 1886, Claude Monet refrained from using ivory black. Few years before his death,

Claude Monet used another color palette, that consists of the following colors: silver, white, cadmium yellow light, cadmium yellow dark, and lemon yellow, emerald green, ultramarine extra-fine, cobalt violet light, and vermilion. We classify these as follows: *pre-1886*, *post-1886* and *final years (FY)*, which refers to the last color palette a few years before Monet's death. We illustrate these color palettes and sample paintings in Figure 3.

3.2 Brush Stroke Techniques

Claude Monet uses fast brush strokes to portray light, a technique honed throughout his career in painting [For95]. A primary example depicting this behavior can be observed in his "Sunrise" painting (Figure 4). Monet uses his signature short, choppy strokes which coaxes the viewer to "optically blend" the strokes and values when painting distant objects close to the horizon [Dun76]. Lastly, fast and broken brush strokes with little to no smoothing are also applied, where viewers see the traces of the brush strokes that sometimes give the painting an unfinished appearance. Impressionist painters like Monet typically paint in one sitting, giving their work a spontaneous feel.

Claude Monet's approach to lighting depends on the brushstroke technique. For example, Monet uses brushstrokes that are soft and diffused while using impasto,for soft, light, and outdoor scenes, which is evident in the "Water Lilies" painting (Figure 4). Impasto is a painting technique where thick layers of paint are laid on an area. The thick layers create a visual effect wherein the brush or painting knife strokes are very visible.



A B

Figure 4: Sample paintings of Monet showing his brushstroke techniques. A: In the Sunrise painting, the details of the water were created using rapid brush strokes where rough patches can be observed throughout. B: The Water Lilies painting illustrates soft and diffused brush strokes. It has an ample amount of indigo underneath each lily in addition to using the impasto technique.

4 ARCHITECTURAL FRAMEWORK

The system accepts an image captured from the mobile AR application after the user places virtual 3D models, which can be moved, rotated, and scaled accordingly to fit a given tabletop scene. The captured image undergoes three steps namely, color palette selection, color mixing, then brushstroke rendering. Figure 5 shows the architectural framework.

4.1 Color Palette Selection

The paintings of Monet were gathered and grouped accordingly to the year it was painted. There are 471 paintings in the pre-1886, 238 in the post-1886, and 149 in the final years. Given an input image captured from a given tabletop scene, the goal is to find the most suitable color palette identified by year. One method to do this is to find the nearest color composition from the cluster of paintings. Finding the nearest color composition among Monet's paintings and comparing it against the input image, by using RGB information, is not appropriate because real-world colors do not necessarily map to Monet's limited color palette. Using image edges, extracted using Sobel operator [KVB88] are sensitive to gradient properties of the image. Furthermore, different brush strokes are used across all paintings which may affect edge detection methods.

We find that using GIST descriptors [OT01] produces the most accurate classification results when selecting a color palette as it appears to be robust against changes in camera view and lighting which may vary in a tabletop scene, especially when captured indoors. GIST descriptors are extracted from the input image which is then compared to the GIST descriptors of Monet's paintings. Since GIST descriptors contain gradient information from varying scales and orientations which are used to recognize various scenes, scenes in Monet's paintings that are similar in composition with the input image will most likely be selected. To perform the color palette selection, using the norm values of GIST descriptors, a k-nearest neighbor algorithm is applied to determine the closest set of paintings for the input image using Euclidean distance ($k = 10$). The class (pre-1886, post-1886, final years) is identified by majority voting and the color palette from the class identified will be selected.

4.2 Color Mixing

With the selected palette, each image pixel's color is changed. Using hue values, the two closest colors in the palette, in terms of Euclidean distance are selected. We observe that performing blending of these two colors produces the best results.

4.3 Brush Stroke Rendering

For rendering brush strokes, a stroke rendering module is implemented where a modified implementation of Capeto's Painter software is utilized [Cap18]. The base implementation of Capeto (2018) [Cap18] was inspired from the painterly algorithm proposed by Hertzmann (1998) [Her98], combined with the brush stroke algorithm devised by Shiraishi and Yamaguchi (2000) [SY00]. To simulate Monet's brush stroke styles, three brush stroke patterns are used by default. The color-changed image from the color mixing step undergoes

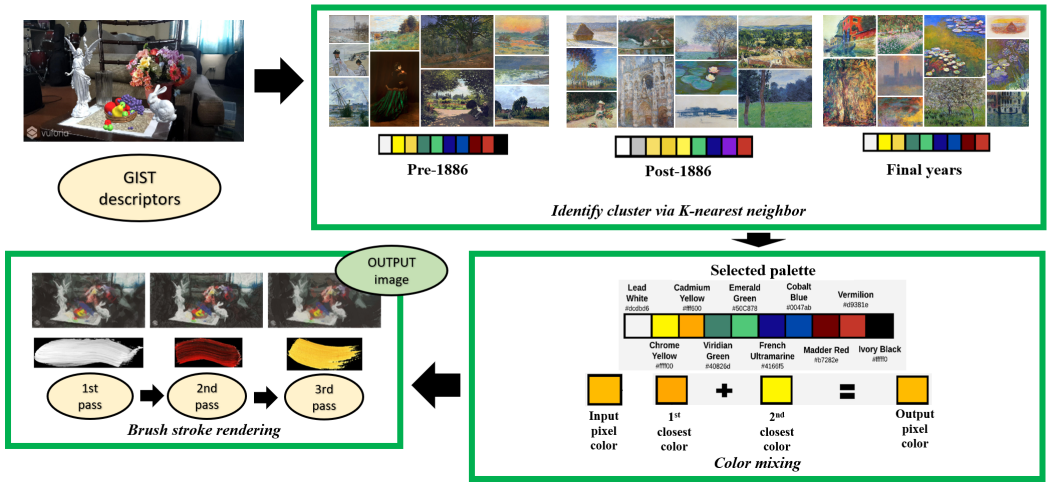


Figure 5: The architectural framework for simulating Claude Monet's impressionistic art style using images captured from a mobile device.

three passes in our stroke rendering module, selecting the corresponding brush stroke pattern. For each pass, a balanced alpha blending is performed as a method for combining the new and previously rendered image.

Curved brush strokes are sometimes not present in the paintings of Monet [For95]. As a refinement step, after the third pass, we perturb the brushstroke pattern by occasionally making the brush strokes straight, using the brush stroke algorithm proposed by Shiraishi and Yamaguchi (2000) [SY00]. Based on our initial experiments, setting a 50% probability of using straight brush strokes produce the best results.

No further major changes were created in the brush stroke implementations mentioned. We recommend the reader to refer to the papers of Hertzmann (1998), Shiraishi and Yamaguchi (2000), and the software designed by Capeto (2018) [Her98, SY00, Cap18].

4.4 AR Application Walkthrough

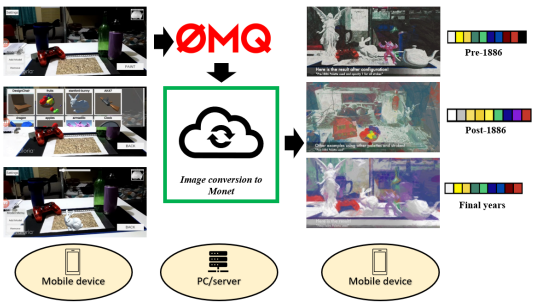


Figure 6: General walkthrough of the AR application. The user assembles their tabletop scene. Multiple 3D models can be placed, rotated, and scaled according to their preference. The image is captured and sent over to a server for processing via ZeroMQ. The rendered image using Monet's art style is sent back to the mobile device for viewing.

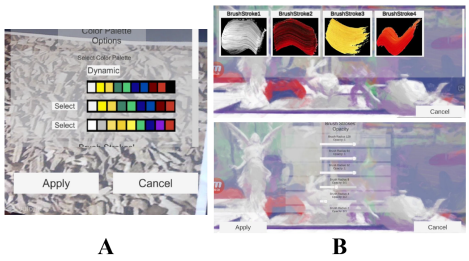


Figure 7: Configurable parameters in our AR application. A: The user can decide whether to use a certain palette or allow the system to dynamically choose a palette based on scene context. B: Additional brush stroke patterns can be imported as a texture file, as well as modify the various parameters for brush strokes, such as opacity, size, and radius.

Figure 6 illustrates a walkthrough of the AR application. The user assembles their tabletop scene where multiple 3D models are placed by tapping on-screen their desired location in the virtual space. The models can then be rotated and scaled through a drag-and-drop interface. The composited scene is captured, and the image is sent to a server for processing via ZeroMQ, using the framework shown in Figure 5. ZeroMQ is a networking library that can transport messages and files between systems through inter-process communication [Hin13]. Offloading the rendering pipeline to a server is preferable as this saves computation time on the mobile device. Based on our initial system designs, we found that it takes approximately 4 to 5 minutes processing time for a 1024×768 image if the rendering step is performed on a mobile device. Further increasing the image resolution causes out-of-memory issues on low-end devices. Offloading the rendering takes approximately 2 minutes for the output image (1024×768 size) to be displayed on a mobile device, subject to network latency.

4.5 Other Technical Details

The AR application is implemented using Unity Engine and Vuforia [LB17]. Image results reported in this paper were retrieved from the actual AR application running on a 4GB mobile device. 3D models are preloaded in the application. Public 3D models such as Stanford Bunny, Dragon, Armadillo, and others are available by default. Our proposed framework is fairly algorithmic and contains numerous hyperparameters (e.g. brush stroke size, orientation, frequency, color values, etc) that directly affect the results, in terms of the images produced. Furthermore, the visual perception and evaluation of the final rendered image are very much subjective to the viewer. Basing on this rationale, we give full control to the user to configure the hyperparameters in real-time (Figure 7). Additionally, the user can import their brushstroke pattern as a texture file to replace one of the rendering passes.

5 RESULTS AND DISCUSSION

We performed a quantitative evaluation of images produced by our framework. Due to the very subjective nature of evaluating images that follow a certain art style, and since there is no universal methodology agreed upon, we solely relied on classical image metrics such as Peak-Signal-to-Noise ratio (PSNR) and Structural Image Similarity (SSIM). The evaluation is performed as follows: Since PSNR and SSIM require paired data, each test image rendered by our system is compared to every painting by Monet, and the corresponding PSNR and SSIM are measured. Hence, mean values are provided for each test image in our discussions and compared with other existing approaches. We refer to these measurements as **MPSNR** and **MSSIM** to denote the mean PSNR and mean SSIM respectively. A total of 859 artworks, combining paintings from pre-1886, post-1886, and final years from Monet, were used for this approach. To speed up the measurement time, we resized the test images and Monet paintings to 256×256.

We compared our method to CycleGAN [ZPIE17], as it is one of the state-of-the-art methods in image-to-image translation and style transfer, and to the style-aware network model developed by Sanakoyeu et. al (2018) [SKLO18] which is trained to cover different art styles. We refer to their model as AdaptiveStyleNet. Both works provide pre-trained models for converting images to Monet paintings. To validate the effectiveness of our color palette selection scheme and brushstroke rendering, we devised an alternative framework that uses only the pre-1886 color palette that we compare against. The brushstroke rendering is replaced with an Anisotropic Kuhawara filter by Kyprianidis, Kang, and Dollner (2009) using default parameters specified in their paper [KKD09b]. We refer to this alternative framework as the baseline.

5.1 Analysis of Results: Tabletop Scenes

A total of 9 test images, consisting of indoor tabletop scenes with varying object compositions, were gathered. To maintain consistency and possibly minimize variance in the results, we limited our 3D models placed in the scene to only Stanford Bunny, Lucy, and Stanford Dragon. The best results are shown in Figure 8. Table 1 summarizes the results.

In terms of MPSNR, the difference among the methods are negligible. In terms of MSSIM, we see bigger differences with our method as compared to CycleGAN, AdaptiveStyleNet, and the baseline. Judging the images visually, our proposed method generates images where the brush strokes can be seen better as compared to CycleGAN and the baseline which positively affected the results in terms of image metrics. In AdaptiveStyleNet, the brush strokes are denser and the strokes drastically changed the edges and fine details of the image. Another interesting observation is that virtual 3D models harmoniously blend with the other foreground objects in the final rendered image, which is not evident in both the baseline and CycleGAN, where the virtual 3D models from these methods stand out in the images. To some extent, we observe that AdaptiveStyleNet exhibits this same property.

5.2 Analysis of Results: Still Life and Outdoor Sceneries

To further validate the performance of our method in terms of closeness to Monet's impressionistic art style, we gathered 11 different images available on the web, most of which are still life compositions and outdoor scenes with vibrant colors. Since most of Monet's paintings were typically still life and outdoor sceneries, we wanted to observe if our proposed method is comparable if we use these compositions as well. The best results are shown in Figure 9. Table 2 summarizes the results.

Based on the MPSNR and MSSIM, our method achieved slightly higher scores than CycleGAN. We speculate that since CycleGAN was trained on diverse real-world image samples, the model can transfer Monet's art style properly on both still life and outdoor scenes. Our method substantially performed better as compared to the baseline and AdaptiveStyleNet, which further proves that the color palette selection scheme and three-pass brushstroke rendering are necessary.

In summary, our method works well when the input images contain one or more virtual 3D models that are placed using AR. However, further study is needed on why our method produces favorable results when virtual 3D models are present.

5.3 Mean Perception Error

Aside from analyzing the performance of our method through MPSNR and MSSIM, we explored how fea-

Table 1: Results of tabletop images in terms of MPSNR and MSSIM. Best values in bold.

	MPSNR				MSSIM			
Image Number: Tabletop Scenes	**Baseline**	**CycleGAN**	**AdaptiveStyleNet**	**Ours**	**Baseline**	**CycleGAN**	**AdaptiveStyleNet**	**Ours**
1	8.9262	9.4079	8.5507	**10.1829**	0.2131	**0.2737**	0.1943	0.2268
2	9.0150	**10.6156**	9.8140	9.6599	0.3253	0.3167	0.2120	**0.3466**
3	8.0774	**9.4321**	8.8698	9.2774	0.3026	0.2749	0.2145	**0.3401**
4	**10.0533**	8.9228	8.2585	9.0555	0.1689	0.2604	0.2123	**0.2872**
5	**9.0216**	8.6646	7.7644	8.7234	0.2384	0.2786	0.2120	**0.3191**
6	9.0979	**9.9304**	9.6984	9.8368	0.3501	0.3448	0.2207	**0.3685**
7	9.0687	**10.9157**	9.9291	9.4783	0.3150	0.3234	0.2161	**0.3332**
8	10.4475	10.9830	11.3076	**12.1283**	0.2243	0.2381	0.1956	**0.2978**
9	9.1286	8.6589	8.5850	**11.0010**	0.2641	0.2767	0.2247	**0.3247**
Average	9.2040	9.7257	9.1975	**9.9271**	0.2669	0.2875	0.2114	**0.3160**

Figure 8: Comparison of images produced by different methods. A: Input. B: Baseline. C: CycleGAN [ZPIE17]. D. AdaptiveStyleNet [SKLO18]. E: Ours.

ture representations in a convolutional neural network (CNN) behave when we use our rendered images as input. Specifically, we utilized the perceptual loss function, which is a measurement between image feature representations extracted from pre-trained CNN [JAL16]. The methodology for measuring MPSNR and MSSIM is similar wherein this time, the difference between feature representations of our images and feature representations of paintings by Monet is measured. If the value is close to zero, it implies that two images have almost identical feature representations. We used VGG-16 as our pre-trained network [SZ14] for extracting the feature maps for this metric, which we refer to as the mean perception error. Table 3 shows the results.

For tabletop scenes, our method consistently achieved the lowest mean perception errors on 7 out of 9 images, which plausibly means that our rendered images have close representations with Monet's paintings. For still life and outdoor images, our method only outperformed the other methods on four images, which further proves that our method generally works best for tabletop scenes with one or more 3D virtual models.

6 CONCLUSION

In this paper, we present an augmented reality application that simulates Claude Monet's Impressionistic art style, by using a multi-color palette selection scheme and brushstroke rendering with three passes. The user can place one or more virtual 3D models on an existing tabletop scene and render the final image. Results show that our method performs generally well on tabletop scenes and comparable to CycleGAN's performance while outperforming AdaptiveStyleNet. Future studies involve studying the user interaction aspect of our AR application and how it's effective in terms of showcasing an impressionistic art style. Other effective methods for clustering Monet's paintings can be explored, such as grouping the paintings by sceneries or color composition and further improving the multi-color palette selection scheme. One possible approach for further improving the performance is to explore how 3D models can be directly altered. Instead of using an

Table 2: MPSNR and MSSIM results of images with still life compositions or outdoor scenes. Best values in bold.

	MPSNR				MSSIM			
	Baseline	**CycleGAN**	**AdaptiveStyleNet**	**Ours**	**Baseline**	**CycleGAN**	**AdaptiveStyleNet**	**Ours**
Still Life Images								
Sunflowers	8.5312	**9.6476**	7.9276	8.5401	0.1586	**0.2073**	0.1673	0.1850
Violet flowers	7.0356	**10.2705**	8.5782	8.1070	0.3118	0.2888	0.2858	**0.3205**
Skull and lemonade	9.6906	10.1778	**10.8628**	9.7463	0.3450	0.3454	0.2542	**0.3586**
Flowers in metal vase, bags and kiwis	9.9569	9.7311	10.1624	**10.3814**	0.3579	**0.3599**	0.2889	0.3522
White flowers with apples	10.3292	10.2516	10.9174	**11.4034**	0.3321	0.3413	0.2477	**0.3505**
Sunflowers with fruits	8.2634	8.4575	9.7994	**11.6888**	0.2368	0.2419	0.1718	**0.2536**
Dining table with fruits	10.1051	10.2556	**10.7580**	10.4910	0.3093	0.3054	0.2495	**0.3345**
Garden entrance	**11.3064**	9.4702	9.2243	10.1062	0.1887	**0.2521**	0.1741	0.2202
Women in the garden	10.9794	10.8531	10.4545	**11.6861**	0.1685	0.2334	0.1949	**0.2557**
Woman in kimono 1	8.6409	**11.1019**	8.3935	9.0217	0.2995	0.2838	0.2382	**0.3122**
Woman in kimono 2	8.1085	**9.8298**	8.6941	8.9167	0.2164	0.2351	0.1774	**0.2522**
Average	9.3588	10.0042	9.6156	**10.0081**	0.2659	0.2813	0.2227	**0.2905**

Figure 9: Comparison of images produced by different methods. The input images are still life compositions and outdoor scenes with vibrant colors. A: Input. B: Baseline. C: CycleGAN [ZPIE17]. D. AdaptiveStyleNet [SKLO18]. E: Ours.

image that captures the whole tabletop scene, using programmable shaders or manipulating materials to make them look impressionistic are worth pursuing.

7 ACKNOWLEDGMENTS

The authors would like to acknowledge De La Salle University, DLSU Science Foundation, and Department of Science and Technology (DOST) for funding this research. Deep learning experiments were performed through the DOST-ASTI COARE facility.

8 REFERENCES

[BC20] Sukanya Bhattacharjee and Parag Chaudhuri. A survey on sketch based content creation: from the desktop to virtual and augmented reality. In *Computer Graphics Forum*, volume 39, pages 757–780. Wiley Online Library, 2020.

[BCL15] Mark Billinghurst, Adrian Clark, and Gun Lee. A survey of augmented reality. *Foundations and Trends® in Human–Computer Interaction*, 8(2-3):73–272, 2015.

[Bre74] Richard Brettell. The'first'exhibition of impressionist painters. *The New Painting: Impressionism*, 1886:189–202, 1874.

[Cap18] Ugo Capeto. Non photorealistic rendering software - the painter: http://3dstereophoto.blogspot.com/p/painting-software.html. 2018.

Table 3: Results of images in terms of Mean Perception Error. Best values in bold.

Image Number: Tabletop Scenes	Baseline	CycleGAN	AdaptiveStyleNet	Ours
1	4.1037	3.8229	4.1341	**3.6987**
2	3.5092	3.4368	3.8146	**3.3100**
3	3.6896	3.7835	3.9466	**3.2802**
4	3.9159	3.7494	4.0065	**3.4909**
5	3.8295	3.7966	4.0794	**3.3856**
6	3.5546	3.5202	3.8391	**3.3368**
7	3.5517	**3.3281**	3.7632	3.3641
8	3.9241	**3.4735**	3.6831	3.6214
9	3.6306	3.6265	3.8762	**3.2814**
Average	3.7454	3.6153	3.9047	**3.4188**

Still Life Images	Baseline	CycleGAN	AdaptiveStyleNet	Ours
Sunflowers	3.9625	**3.6334**	3.8588	3.6404
Violet flowers	3.2255	3.2992	3.3895	**3.1899**
Skull and lemonade	**3.0381**	3.1280	3.3324	3.0637
Flowers in metal vase, bags and kiwis	**3.0093**	3.1236	3.1688	3.0714
White flowers with apples	3.2119	**3.0695**	3.2664	3.2040
Sunflowers with fruits	3.6379	**3.3804**	3.7759	3.5223
Dining table with fruits	3.2522	3.2444	3.2661	**3.1897**
Garden entrance	3.4800	**3.4130**	3.8250	3.5501
Women in the garden	3.6298	3.5177	3.5959	**3.3011**
Woman in kimono 1	3.3608	3.3371	3.7211	**3.2904**
Woman in kimono 2	4.0126	**3.5503**	3.8819	3.6180
Average	3.4382	3.3360	3.5529	**3.3310**

[CH03] John P Collomosse and Peter M Hall. Cubist style rendering from photographs. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):443–453, 2003.

[CRH05] John P Collomosse, David Rowntree, and Peter M Hall. Stroke surfaces: Temporally coherent artistic animations from video. *IEEE transactions on visualization and computer graphics*, 11(5):540–549, 2005.

[CS07] Chi Chu and Zen-Chung Shih. Painterly rendering framework from composition. *Journal of WSCG*, 15:151–158, 2007.

[Dun76] Bernard Dunstan. Painting methods of the impressionists. 1976.

[FMSGA18] Zoe Falomir, Lledó Museros, Ismael Sanz, and Luis Gonzalez-Abril. Categorizing paintings in art styles based on qualitative color descriptors, quantitative global features and machine learning (qart-learn). *Expert Systems with Applications*, 97:83–94, 2018.

[For95] Andrew Forge. *Monet*. Art Institute of Chicago Chicago, 1995.

[FYX17] L. Feng, X. Yang, and S. Xiao. Magictoon: A 2d-to-3d creative cartoon modeling system with mobile ar. In *2017 IEEE Virtual Reality (VR)*, pages 195–204, 2017.

[GEB16] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[Ger18] Vladimir Geroimenko. *Augmented Reality Painting and Sculpture: From Experimental Artworks to Art for Sale*, pages 211–225. Springer International Publishing, Cham, 2018.

[Hae90] Paul Haeberli. Paint by numbers: Abstract image representations. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 207–214, 1990.

[HE04] James Hays and Irfan Essa. Image and video based painterly animation. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 113–120, 2004.

[Her98] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460, 1998.

[Hin13] Pieter Hintjens. *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[JAL16] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[Jul81] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[KCWI12] Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the" art": A taxonomy of artistic stylization techniques for images and video. *IEEE transactions on visualization and computer graphics*, 19(5):866–885, 2012.

[KD08] Jan Eric Kyprianidis and Jürgen Döllner. Image abstraction by structure adaptive filtering. In *TPCG*, pages 51–58, 2008.

[KHEK76] M. Kuwahara, K. Hachimura, S. Eiho, and M. Kinoshita. *Processing of RI-Angiocardiographic Images*, pages 187–202. Springer US, Boston, MA, 1976.

[KKD09a] Jan Eric Kyprianidis, Henry Kang, and Jürgen Döllner. Image and video abstraction by anisotropic kuwahara filtering. *Computer Graphics Forum*, 28(7):1955–1963, 2009.

[KKD09b] Jan Eric Kyprianidis, Henry Kang, and Jür-

gen Döllner. Image and video abstraction by anisotropic kuwahara filtering. In *Computer Graphics Forum*, volume 28, pages 1955–1963. Wiley Online Library, 2009.

[KS04] Levente Kovács and Tamás Szirányi. Painterly rendering controlled by multiscale image features. In *Proceedings of the 20th Spring Conference on Computer Graphics*, pages 177–184, 2004.

[KS06] Levente Kovács and Tamás Szirányi. 2d multilayer painterly rendering with automatic focus extraction. *WSCG: The 14-th international Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006*, pages 141–146, 2006.

[KVB88] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988.

[LB17] Jonathan Linowes and Krystian Babilinski. *Augmented reality for developers: Build practical augmented reality applications with unity, ARCore, ARKit, and Vuforia*. Packt Publishing Ltd, 2017.

[Lit97] Peter Litwinowicz. Processing images and video for an impressionist effect. SIGGRAPH '97, pages 407–414, 1997.

[Mar18] Todd Margolis. *Immersive Art in Augmented Reality*, pages 183–193. Springer International Publishing, Cham, 2018.

[MOT15] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.

[Mou03] David Mould. A stained glass image filter. In *Proceedings of the 14th Eurographics workshop on Rendering*, pages 20–25, 2003.

[MW78] Claude Monet and Daniel Wildenstein. *Monet's Years at Giverny: Beyond Impressionism*. Metropolitan museum of art, 1978.

[Nak19] Reiichiro Nakano. Neural painters: A learned differentiable constraint for generating brushstroke paintings. *CoRR*, abs/1904.08410, 2019.

[OT01] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[RAGS01] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.

[Sam04] Margaret Samu. Impressionism: Art and modernity. *Heilbrunn Timeline of Art History*, 2004.

[SKLO18] Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, and Bjorn Ommer. A style-aware content loss for real-time hd style transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 698–714, 2018.

[SLKD16] Amir Semmo, Daniel Limberger, Jan Eric Kyprianidis, and Jürgen Döllner. Image stylization by interactive oil paint filtering. *Computers & Graphics*, 55:157–171, 2016.

[STD+16] Amir Semmo, Matthias Trapp, Tobias Dürschmid, Jürgen Döllner, and Sebastian Pasewaldt. Interactive multi-scale oil paint filtering on mobile devices. In *ACM SIGGRAPH 2016 Posters*, pages 1–2. 2016.

[SY00] Michio Shiraishi and Yasushi Yamaguchi. An algorithm for automatic painterly rendering based on local source image approximation. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 53–58, 2000.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[TZ99] Philip HS Torr and Andrew Zisserman. Feature based methods for structure and motion estimation. In *International workshop on vision algorithms*, pages 278–294. Springer, 1999.

[UVL17] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6924–6932, 2017.

[VC13] David Vanderhaeghe and John Collomosse. Stroke based painterly rendering. In *Image and Video-Based Artistic Stylisation*, pages 3–21. Springer, 2013.

[WXSC04] Jue Wang, Yingqing Xu, Heung-Yeung Shum, and Michael F Cohen. Video tooning. In *ACM SIGGRAPH 2004 Papers*, pages 574–583. 2004.

[ZJH18] Ningyuan Zheng, Yifan Jiang, and Dingjiang Huang. Strokenet: A neural painting environment. In *International Conference on Learning Representations*, 2018.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

A Framework Enabling Real-time Multi-user Collaborative Workflow in 3D Digital Content Creation Software

Swann Martinez
Paris 8 University
INREV research team
2 rue de la liberte
France 93526, Saint-Denis
R&D engineer at CUBE CREATIVE
swann.martinez@protonmail.com

Chu-Yin Chen
Paris 8 University
INREV research team
2 rue de la liberte
France 93526, Saint-Denis
chen.chuyin@mx.nthu.edu.tw

ABSTRACT

This paper reports on a graph-based approach to enable real-time update of 3D shared elements over the network between multiple artists working simultaneously on the same 3D scene. It presents an experimental framework for exploring real-time collaboration in Digital Content Creation such as animation. The framework combines a push-pull network architecture and data translation protocol with hybrid command/data replication mechanisms. This enables the synchronization of object components and hierarchy between multiple instances of the same Digital Content Creation application in a non-destructive way. Conflicts between users are prevented with a strong right management system. We demonstrate the interest of such an approach by means of an application example in Blender and discuss collaborative experimental sessions outcomes over a Local Area Network and through the Internet.

Keywords

Real-time collaboration, 3D creation workflow, Multi-user, Graphical human computer interfaces, 3D Scene Dependency Graph Replication

1 INTRODUCTION

An animation movie is an idea shaped throughout the production process by many people. A typical 3D animation pipeline requires a linear succession of tasks, from storyboarding to compositing through various stages; its workflow is similar to an assembly line.

Collaboration between each task of the production chain relies on the exchange of individual work files (see Fig. 1). The dataflow supports the workflow: the transition between steps requires export phases that format the file resulting from the previous task into a new file for the next task's tools.

Each production stage is handled by an almost entire studio department. It is therefore common that an artist working in department **A** does not know what another person did during a previous step in department **B**. This lack of communication (also known as the "silo effect") often has a negative impact on the final production by hindering error anticipation. A problem may go unnoticed for several successive steps before being discovered. When this occurs, the linear nature of the production pipeline requires going back to the root of the issue and redo all the subsequent steps, which is a very costly process.

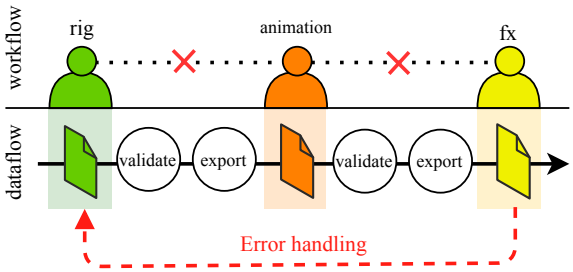


Figure 1: A traditional dataflow in animation productions

If real-time rendering engines speed up traditional linear productions by drastically reducing iteration time within the different stages, they could also become a new communication tool within animation production teams by bringing 3D designers to collaborate with each other across a real-time multi-user interface.

Real-time collaboration is widely used in other fields. In software development, it appears in peer-programming solutions such as Teletype [Git17] to

facilitate knowledge transfer. In civil engineering, Autodesk BIM[1] is an interdisciplinary collaboration hub where each actor of the construction (architects, engineers and contractors) sees in real time the impact of their work on the others.

Bringing real-time collaboration to the heart of creative applications in animation is a complex problem. From a technical point of view, the 3D scenes of a production are made of a wide variety of large software-dependant creation data (meshes, materials, rigs, textures,...). From a human point of view, the vertical communication flow and the segmented validation process do not allow for an easy multi-user collaboration.

The experimental work described in this paper results from a research-creation approach. Our framework emerged from many collaborative real-time creation sessions conducted in industrial and academic context thanks to its initial Blender integration (see section 4).

In the course of these experiments, we identified several requirements for the development of an efficient multi-user collaboration within a 3D authoring software (that will be hereafter also referenced by the name of Digital Content Creation Software or DCC):

- User interface independence: do not interfere in the interfaces of the 3D software, let the users use it as it is designed to work.
- Asynchronous editing of the 3D scene: allow different users to edit different parts and aspects of the scene simultaneously and without any friction.
- Error independence: An error caused locally by one user should not impact other users.
- Flexibility of integration: allow the DCC program to drive the replication pipeline.
- Non-destructive dataflow: allow the DCC program to specify the data types it wants to replicate.
- Adaptive workflow: adapt the behavior of the collaboration to the needs of the artists (e.g. be able to disable the replication of certain data).
- Dynamic loading: allow artists to join and leave a collaborative creation session at any time.

To address these needs, we propose a framework that brings real-time collaboration to the heart of 3D authoring software. We articulate the critical notions of dataflow and workflow at the core of the framework thanks to a data definition interface, along with a set of collaboration-aware functions allowing to model the collaborative workflow to the artists' needs (layout, lighting, modeling tasks, etc.).

Our method uses an acyclic dependency graph to store and structure the creation data for replication across the corporate Local Area Network or through the Internet. Each node contains serialized blocks of data, also called datablocks, describing any element of the 3D Scene that is stored and used by the 3D DCC software (for example: a shader, a mesh, a particle system, etc.). A translation protocol for replicated data (i.e. RDP) enables the definition of node types to match the data structures of the 3D software API.

As soon as a datablock type has been defined in the RDP, the framework will replicate instances of this given datablock type across all the connected clients. To avoid access conflicts, each node is subject to a strict modification rights management policy. This node-based approach provides an adaptive level of granularity to address the complexity of data that is specific to 3D scenes used in animation.

The user and the rights management system enable the storage of user-specific metadata in order to support the addition of new collaboration tools. These tools contribute to improve the artists' collaboration awareness by smoothing and optimizing the workflow of the multi-user creation experience.

2 RELATED WORK

This section focuses on previous work on the topic of real-time co-creation and explains the differences with the proposed approach.

The first attempts to design a framework offering a WYSIWIS experience (What You See Is What I See) [Mar87] in a single user software started in 1987. Dewan and Choudhary addressed this problem by proposing a high-level framework based on a semi-replicated network architecture [Dew92]. However, this approach based on a callback system does not take into account relational data, a crucial element for 3D creation where the entire scene is formed by a multitude of data blocks depending on each other.

Indeed, a large majority of 3D DCC solutions such as Blender[2] or Autodesk Maya[3] use dependency graphs to efficiently handle the updating of scene elements. By relying on a similar structure, our approach is thus particularly adapted to the specific needs of 3D DCC software.

In animation, real-time collaboration solutions can be categorized as cross-DCC or mono-DCC applications. A mono-DCC approach consists in replicating information within several instances of the same application. In contrast, a cross-DCC application will propagate the collaboration between different applications (Example: a texturing application, a modeling application, etc.).

By relying on the *Universal Scene Description* created by Pixar (i.e. USD) [Pix21] to manage 3D scene data, the Nvidia Omniverse solution [Nvi21] represents an

[1] https://www.autodesk.com/solutions/bim

[2] https://www.blender.org

[3] https://www.autodesk.com/products/maya

interesting cross-DCC collaborative platform. However, the standardization of 3D scene transmission formats is still limited. Some data types such as rigs are not supported. As a result, the Nvidia approach [Nvi21] is restricted to the data types supported by the USD [Pix21]. Moreover, it is complex to obtain a homogeneous interaction interface in the cross-DCC approach due to the diversity of user experiences (UX) of the solutions. To overcome these limitations, our framework focuses on a mono-DCC approach and does not impose a specific 3D data format; instead, it provides an interface for specifying the structure of the replicated creation data and their relationships to fit the application's specifics.

From an architectural point of view, the *Verse* protocol [Hni11] provides an efficient solution to the delays of 3D data transmission, but it relies on a centralized architecture. Very early in our experiments, the artists asked us for the possibility to suspend the replication of certain data on demand in order to iterate changes without impacting the other users. This requires saving updates locally, which is impossible when operations are centralized.

Another interesting approach is the Mixer add-on [Ubi20] developed for Blender by UBISOFT. Its centralized command-based architecture relies on small transfers which resulted in good responsiveness from a performance perspective. It handles the scene representation as a linear stack of commands (stored on server-side). The stack grows during the scene creation. This architecture does not provide the relational data required to prevent user editing conflicts, which is a critical need we identified in our first experiments. Since artists usually edit several aspects of an asset, the access control system must be aware of that asset's dependencies in order to lock them. Moreover, its centralized architecture does not allow the execution of local operations because it fully relies on the order of commands to rebuild the scene.

In the *MMConf* infrastructure [Cro90], Crowley and Milazzo demonstrate that a replicated architecture enables the implementation of client-specific operations when the environment is copied locally. By having a complete version of the dependency graph managed locally on each client, our framework is able to perform local operations without impacting the other connected clients.

3 PROPOSED METHOD

None of the previous approaches allowed us to explore real-time collaboration within several instances of the same 3D authoring tool in a non-destructive way and without editing conflicts.

The proposed approach is designed to be integrated in *collaboration-transparent programs* [Lau90], such as 3D authoring software that is completely unaware that it interacts with several users active on the same scene. This framework contributes to solve some of the problems listed above about multi-user experience implementation for the 3D content creation in animation thanks to the following features:

- A replicated data translation protocol: Acting as an interface for the definition of replicated datablocks.
- A session system: Managing the connection/disconnection of users at any time during the scene creation.
- An access control system: Preventing concurrent datablocks edition conflicts.
- Collaboration-aware functions: Providing functions to manage the replication pipeline behavior from the DCC application programming interface.
- Collaboration-aware interface: Providing an interface to enable the creation of collaboration coordination tools (e.g. to help users to organize themselves).

3.1 A multi-user abstraction layer for 3D content creation software

In animation, most of the 3D DCC software solutions provide a python API to extend their functionalities and integrate them in a production pipeline. Therefore, our multi-user framework has been designed as a python module to facilitate its integration.

In traditional 3D DCC, users can access many operators in order to create and modify a 3D scene. Depending on the 3D authoring software, the creation workflow will be more or less procedural. In a classic linear pipeline, a loss of information occurs during the export between tasks. This ensures to lock the artistic aspect for the next tasks. But in the context of real-time multi-user collaboration, tasks are parallelized. Thus, keeping available all the creation information in a non-destructive way is mandatory to guarantee the potential parallelism of any task within the creation. However, operators are often bound to a run-time context. Replicating an operation would require synchronizing the associated context and imposing it to other clients when the operation is being applied, resulting in potentially unexpected interface behavior and degradation of the user experience.

As a consequence, we opted for a data replication approach. This has several benefits with respect to our initial needs.

- It preserves the ability to collaborate on procedural data (e.g. a nodal network to generate geometry).
- It facilitates the implementation of a distributed architecture to apply user-local operations on data.
- It allows to extend creation data with an access control system.

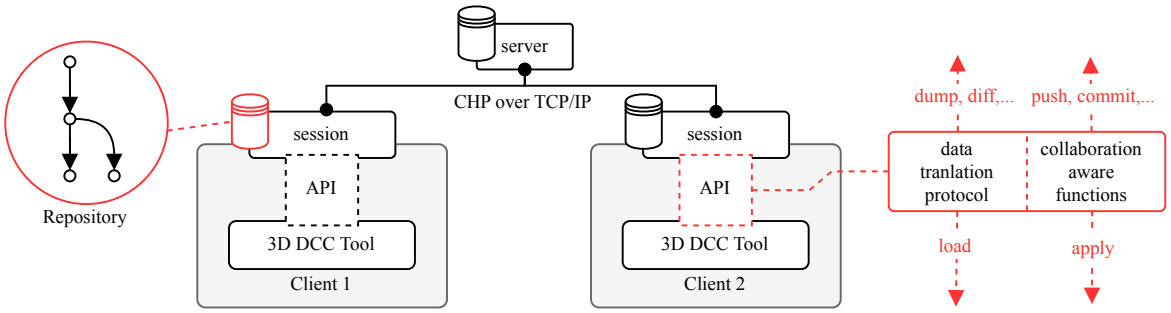


Figure 2: Overall framework architecture

In our framework, a *client* and an *application instance* are considered equivalent. Clients connect to a session stored on a replication server. The *session* is a state object over which the server has authority; it can be compared to the GameState in the video game. It stores the information of the connected clients and the status of the current creation session. The *server* is a lightweight script that can be launched on the computer of any of the clients or on a dedicated server.

3.2 Framework architecture

The framework is structured as a combination of a distributed and a centralized architecture (see Fig. 2). The distributed approach handle scene data replication while the management of commands (such as locking, connecting, etc.) is centralized to ensure server authority.

Distributed data replication

The scene data is stored in a dependency graph generated by the data translation protocol. As shown in Fig. 5, this graph is stored in a *repository* object as a key-value dictionary of entries. A key represents the unique identifier of a node and the value is the associated data.

By cloning the latter during their connections, clients get a complete local copy of this data (similar to a clone operation on Git [Jun05]). Thereafter, the changes made to it will be applied locally and then replicated. This addresses the problem of error independence. By executing certain of the collaboration-aware functions (see section 3.5) locally, errors do not affect other users.

Centralized command replication

A command consists in a set of instructions to execute on the repository (e.g. lock/unlock a node). The server which has authority for access control validates whether a command can be applied. Therefore, all commands are first sent to the server for an authorization check (e.g. does the client have the right to lock/unlock this node ?). Then they are applied on the server repository and relayed to other clients for execution. The commands are used across the different components of the framework to replicate such as:

- `Authenticate`: Login to a server.
- `Clone` : Transfer a full copy of the server repository to the local client.
- `Lock/Unlock` : Used by the right management system (see section 3.4) to acquire/release the modification right on given nodes.
- `Kick`: Remove a given user from the session.
- `UpdateUserMetadata`: Used by collaboration-aware interfaces (see in section 3.6) to update the metadata of a given user.
- `Delete`: Remove the given nodes from the repository.

Network architecture

The framework implements a modified version of the *Clustered Hashmap Protocol* [Hin11] (i.e. CHP) to exchange data across the network.

To reduce the bandwidth used by data transfers, node changes are transmitted as deltas computed during `COMMIT` (see in Section 3.5). The size of these deltas varies greatly depending on the nature of the change. E.g. moving an object will generate a small delta while updating a complex mesh will be much larger. Therefore, the granularity of the transmitted data depends directly on the modifications made by the users.

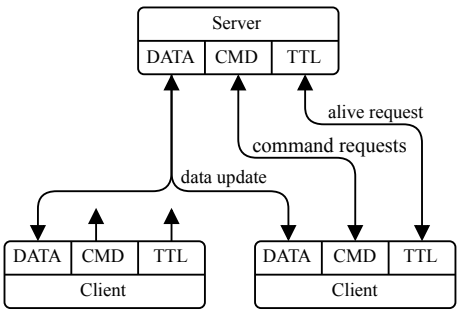


Figure 3: Network communication layer

As shown in Fig. 3, data and commands are transmitted through two different sockets. This enables to handle them asynchronously. By separating these two kinds of transactions at server level (see Fig. 4), we guarantee the responsiveness of the command-based right system even when large delta are transferred.

Each channel is based on TCP to limit packet loss. The TTL socket mechanism allows to measure the ping response and the latency to determine if the clients are online. On client side, this regular heartbeat mechanism is handled in a separate process, so that it will be less impacted by the heavy computations typical of 3D creation tools (rendering, etc.). Same strategy is applied server-side.

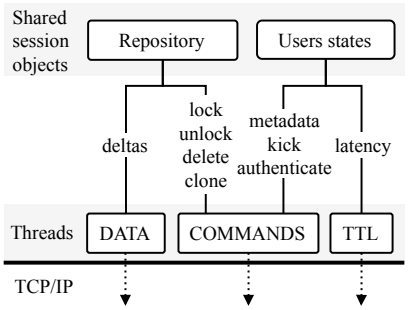


Figure 4: Server internal architecture with asynchronous network frame reception.

3.3 Replicated data definition protocol

The Replicated Data definition Protocol (abbreviated as RDP) represents the keystone between the DCC program and the framework. It acts as a translation dictionary to transport data between the two (see Fig. 5).

To enable the support of a given type of datablocks, it is necessary to define an implementation of the abstract class `ReplicatedDatablock` offering the following methods:

- `dump`: Translates the target object into a dictionary using standard types.
- `load`: Loads the dictionary data into the DCC datablock.
- `construct`: Instantiates an empty datablock.
- `resolve`: Search for an existing instance of a specific datablock.
- `resolve_dependencies`: Returns the dependencies of a given datablock (e.g. textures, meshes, etc.).

The methods above are used by the collaboration functions (see section 3.5) across the replication pipeline to exchange data between the local repository and the DCC datablocks.

The definition of these implementations takes place only once. This occurs during the integration phase of the framework into the DCC software. As they will deeply interact with the 3D authoring software, a strong knowledge of the program's python API is required. In our application example (see section 4), we developed those implementations as sub-module of the add-on responsible for the framework integration. Once defined, these implementations are stored in a key-value dictionary (see Fig. 5). The key is the datablock type and the value is the corresponding implementation. This dictionary is then transmitted to the framework during the session initial connection. In this way, the framework is able to interact with the DCC data to replicate them.

The implementation dictionary will be used all along the session by the framework to instantiate the defined implementations. These instances represent the different nodes of the replication graph stored in the repository.

3.4 Concurrency access control

Initially, as the framework had no access control system and several users were able to modify the same datablock, which naturally led to editing conflicts. To address this, we limited the modification of a datablock to a single artist by means of the right management system. The access control system manages the ownership of each node of the replication graph. It will allow or refuse/disable the modification of some nodes of the latter according to their owner. The `change_owner` function allows to change the rights on a node and optionally its dependencies. When a node belongs to a user, only this one has the ability to modify it.

By default, the right management policy assumes that a node belongs to what we call 'the common right'. That is, when a user unlocks a node, it yields its ownership to the common right. Only nodes belonging in the common right can be locked by a user. This access control system ensures frictionless collaboration by allowing users to work asynchronously on different datablocks of the 3D scene.

3.5 Collaboration-Aware Functions

The collaboration-aware functions (i.e. CA-function) are the core of the framework's replication pipeline. They are exposed to the DCC program to let it control and adapt the rate of updates to its needs. Any of them can be called automatically (by the program) or manually (by the user). The Fig. 6 shows the place of these functions in the replication chain. We took inspiration from Git [Jun05] to develop these mechanisms of collaborative work provisioning.

As shown on Fig. 5, the CA-function `ADD` first adds the targeted datablock to the local repository by instantiating a new node corresponding to its type (with the RDP presented in 3.3). This is the entry point of the replication pipeline. When a datablock is added to the repository, it is considered as being tracked. During a session, `ADD` must be called to register each new datablocks.

Then, the CA-function `COMMIT` can be executed on any node. Firstly, it will check the state of the datablock's dependencies to ensure published data integrity. For each datablock returned by

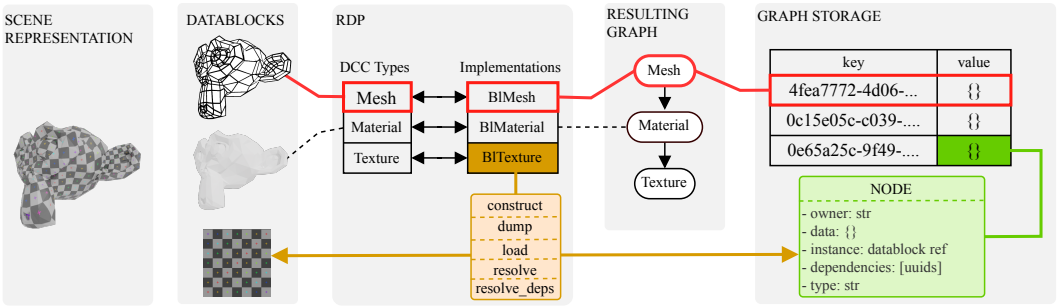


Figure 5: Data translation with the RDP

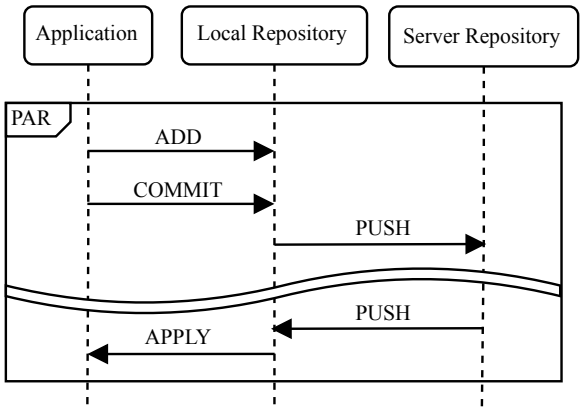


Figure 6: Data collaboration primitive pipeline

`resolve_dependencies` (see section 3.3), it will check if it is tracked and up to date. If so, the latter will be subject to an `ADD` or a `COMMIT`. Secondly it will calculate the changes and apply them to the corresponding node in the local repository. The changes consist of a differential of the data extracted (with `DUMP`, see section 3.3) from the datablock in its current state and the last committed version stored in the node in the `data` field (see Fig. 5). We use DeepDiff [Zep21] to compute the delta between the two dumped versions of the datablock.

The `PUSH` CA-function is then called to transfer local changes to the server which will apply them onto its own repository and relay them directly to connected clients.

As soon as the modifications are received by the connected clients, they can be loaded with `APPLY`. This CA-function checks the corresponding datablock existence with `resolve` (see section 3.3). If it is not resolved, it means that it is a new datablock corresponding to a new scene element. In that case, it will be instantiated using `construct` (see section 3.3). Once a datablock instance is found, the `load` method (see section 3.3) of the node implementation will load its updated data into it.

The default behavior is to `COMMIT` then `PUSH` as soon as a change occurs in the scene in order to ensure real time updates. In certain situations (e.g. working on large volumes of geometry) it is essential to give the artist the ability to temporarily stop sending data to avoid impacting the performance of other clients. By separating the replication steps into functions, we address the problem of disabling outgoing updates and the need for an adaptive workflow.

3.6 Collaboration-Aware interfaces

During 3D scene editing, the user interactively edits the assets in a spatial and temporal context. If we extrapolate this into a multi-user environment, it is important for each user to be aware of who is working on what element, where and when through collaborative user interfaces.

Key	Value
`view_matrix`	`[[0.0, 0.9, 0.0, 0.0], [-0.4, 0.0, 0.9, 0.0], [0.9, 0.1, 0.4,-5.0], [0.0, 0.0, 0.0, 1.0]]`
`color`	`[0.0, 0.3, 0.2, 1.0]`
`frame_current`	`91`
`scene_current`	`main_stage`
`selected_objects`	`['object_1', 'object_2']`

Table 1: A sample of the user metadata dictionary used in the Blender framework integration

The `user_states` object of the framework is intended to support such interfaces. More concretely, it is a dictionary of metadata (e.g. in Tab. 1) specific to each user stored and replicated across all clients. The framework provides two functions to interact with this object:

- `update_user_metadata()` allows a client to update one or more of its metadata fields across all connected clients.
- `get_users_states()` retrieve the metadata dictionnary of all the connected clients (including the local one).

To maintain collaboration awareness, it was crucial to support a high metadata refresh rate. E.g. the temporal snapping operator required a minimal 30 Hz refresh rate on the `frame_current` field. To do so, we limited the size of the updates to the strict minimum. To do so, when a client request to update a data field with `get_metadata`, only this latter

will be relayed through the network encapsulated in an `UpdateUserMetadata` command (see in Sec. 3.2) that will patch all connected client's `user_states` object.

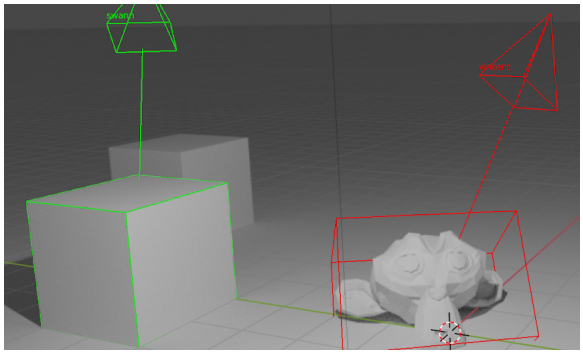


Figure 7: Collaboration aware interface in Blender viewport

For instance, the implementation of the framework in Blender relies on these interfaces to store the user's point of view (see the `view_matrix` field in Tab. 1) in order to draw their frustums in the viewport of the other connected clients (see Fig. 7). The same concept is applied to represent the active selection (see the `selected_objects` field in Tab. 1) bounding box of each user. These two user interface elements make connected artists completely aware in real time of where each other are and what they are working on.

To allow artists to find each other in space and time, two operators have been implemented in Blender. They rely on the metadata made available through a call to `get_user_states` (respectively the `view_matrix` and `frame_current` fields in Tab. 1).

4 APPLICATION EXAMPLE

As an open-source Swiss army knife for 3D content creation, Blender can handle all stages of film production.

The animation studio CUBE CREATIVE [Cub21] has made it its main DCC since 2019, from asset creation (modeling, texturing, shading and rigging) to rendering and animation.

A key advantage for real-time collaboration is its unified shader system between its real-time rendering engine (EEVEE) and its path tracing engine (Cycles), which allows for high-fidelity pre-visualization of material rendering in real-time in the viewport. Applied to the multi-user context, this provides the possibility for all users to visualize in real-time the visual changes of the scene in EEVEE while keeping a fidelity close to the final rendering in CYCLES.

We integrated our framework into Blender as an open-source add-on called *Multi-User* [Mar19]. This made possible the study of the impact of real-time collaboration in two different environments:

- An industrial one: In the studio CUBE CREATIVE (where the framework was developed) with supervisors and artists teams specialized in animated series.
- An Academic one: For teaching computer graphics at university Paris 8.

The collaboration functions described in 3.5 were placed in the callbacks provided by Blender (like: `depsgraph_update_post`). Thus, Blender directly notifies the framework of any updates of the scene and the framework publish these changes to all the clients.

5 ANALYSES AND OBSERVATIONS

From July 2019 to January 2021 we conducted a set of experimental sessions at CUBE CREATIVE using our Blender integration of the framework (the Multi-User add-on). The real-time collaboration was applied to concrete use cases reflecting the reality of a production through different exercises:

- Scene re-creation: Create scenes (any aspect) from scratch, while being guided by a 2D reference image.
- Background concept: Create scenery from already existing assets issued from a production.

As mentioned in the section 1, the production of a film linearly passes through many stages. Each step is isolated and validated one by one. Within the production, the real time collaboration can be conceived within one or several production stages. These two configurations will lead to two very different team compositions. We will refer to a multi-disciplinary team for the collaboration invoking multiple production stages, whereas for performing collectively a single kind of task, we will refer to a specialized team.

Both types of exercises were designed to test these two team configurations. For the background concept exercise, the teams are specialized. But, the scene re-creation requires several professions working together because it encompasses all aspects of creation.

These two classes of exercises allowed us to evaluate our approach in terms of efficiency and quality.

5.1 Collaboration efficiency

To study the impact of real-time collaboration on work execution efficiency, we conducted a series of scene re-creation sessions based on references images chosen for their diversity.

Each of these scenes was created twice on Blender: one version was created by a single person executing sequentially the different tasks, and the other version was created by a team collaborating in real time with multi-user add-on through the internet. In both cases, the artists had to create the following aspects:

- 3D models
- shading
- layout
- fx
- lighting
- compositing
- rendering

Despite being an abstract notion, the qualifications of the artist is a very important factor regarding the results. We can distinguish three levels of experience in animation teams: junior, mid and senior artists. In the single-user creation experiment, the artists had a mid-experience, such that they can handle all aspects of the content production. In the multi-user part of the experience, the collaborative team was mixed equally with mid and junior artists.

Scene			SU	MU	
name	*tris*	*Oc*	*t*	*t*	*Ts*
Campsite	3430	161	63	45	3
Paper Summit	22215	58	90	60	5
All seing monolith	59000	146	177	90	4
Xbox clubs image	16112	726	238	175	4
Abstract city	2267256	139	472	295	5

Table 2: Scenes re-creation sessions time (t) in minutes in single-user(SU) and multi-user(MU) configuration with corresponding team size(Ts), Total triangles($tris$) and objects count (Oc)

Tab. 2 shows how real-time collaboration can speed up the scene creation time. This gain varies according to the complexity of the scene and the size of the teams. According to our data, teams of 4 artists obtain the highest efficiency rate on complex scenes. Although encouraging, the method's efficiency could be greatly improved with an optimisation of the work distribution before the session. During the re-creation sessions, we didn't impose that, and as a result, we found the artists, by habit, tended to work individually on the assets. Thus, we observed the main factor limiting the parallelization of work was organizational. The goal would be to overlap the creation of multiple aspects of the same asset to improve the collaboration workflow parallelism. For example, one artist can go-on building an asset while another could place it in the scene.

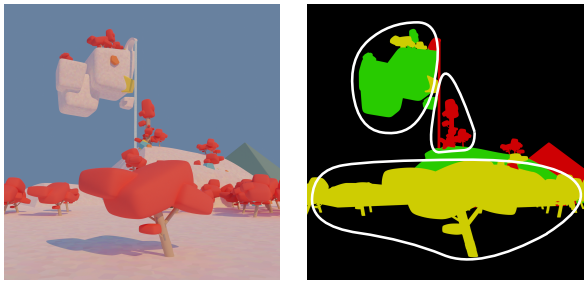
Beyond efficiency, the re-creation sessions highlighted other benefits of the real-time co-creation. Although working on individual assets, the artists were naturally led to simultaneously work on several aspects of the scene. E.g. it was common for one artist to start the lighting while other are modeling and laying out elements. As a result, they were aware in real-time of their impact on the work of others which greatly improved error handling as opposed to a linear industry pipeline.

5.2 Qualitative observations

While the analysis of the re-creation sessions has shown the efficiency of our framework for collaborative work in a concrete context (guided by references), we also questioned the ability of such a workflow in an abstract context such as in pre-production where new designs have to be created.

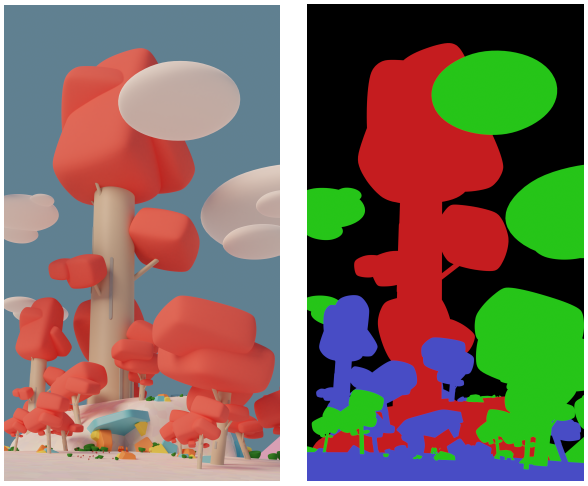
During the collaborative background concept sessions, the teams were composed of three to four members ranging from mid to senior level.

In order to be as close as possible to real life conditions, we used a set of existing assets coming from the Tangranimo series currently in production at CUBE CREATIVE. The exercise focuses on a collaborative layout to design new sets based on an existing visual style.



(a) (b)

Figure 8: Background concept session 1 result, including (a) the scene render and the corresponding (b) user work distribution by color.



(a) (b)

Figure 9: Background concept session 2 result, including (a) the scene render and the corresponding (b) user work distribution by color.

The spatial distribution of work is a particularly interesting criterion to study in this kind of improvisational exercise. It provides a visual indication of the collaboration quality: this can be measured by coloring the space regions occupied by objects according to the user

who worked on them. When the collaboration is not fluid, we observe very distinct islands that do not intersect as well as a dominance of the occupation rate of certain colors.

user	occupancy
Session 1	
yellow	3.9
green	1.9
red	0.8
Session 2	
green	6.2
blue	3.4

Table 3: Users work occupancy percentage on the background concept session results.

This was the case of the first session which was conducted without any preparatory phase. In the user's work distribution (see Fig. 8b), we observe very distinct islands that do not intersect, as well as a dominance of yellow on the spatial occupancy rate (3.9 % for the yellow user against 1.9 % for the green and 0.8 % for the red, see Tab. 3). It means that the artists were working in a spatially isolated way, revealing that the collaboration was not smooth, as if they were shy of interacting with each other. As a direct consequence, the resulting scene shown in Fig. 8a lacks of coherence.

In the second experimental session we introduced a 5-minute briefing period at the beginning of the session dedicated to settle the creation of a common foundation (in red in Fig. 9b). As shown in the user work distribution in fig. 9b, the common conception of the space resulted in a much more uniform distribution of work. The resulting scene (see Fig. 9a) is coherent.

When errors occurred (e.g., an object casting an unintended shadow), we observed that the real-time nature of the collaboration allowed the participants to notice them and instantly fix them. In a traditional production pipeline, this process would have been much more time-consuming and tedious (as shown in fig. 1). Thus, using the framework permitted to improve considerably the communication between artists, greatly increasing the anticipation of errors. Furthermore, the artists started talking to each other during the creation, which was not possible before because of the "silo effect". Adding this social dimension to the creative process has led artists to evaluate and re-calibrate their work according to the scene being created. This natural review process is intrinsic to the global vision of the project given by the real-time nature of the collaboration. It allows the artists to communicate about their practices and thus generates a natural transmission of knowledge between the different levels of experience.

Academic application

Although we mainly addressed the industrial application of this work, it turned out that it would add a new dimension to the teaching of computer graphics. For example, we used the Multi-User add-on during 3D modeling courses held at the department Art and Technology of the Image of University Paris 8 in 2020. This led the teacher to share the same virtual space as the students. By interacting with their practical work in real time, the teacher is able to react naturally and quickly to the questions and difficulties of each student. From an educational point of view, our work may have a relevant application in the academic world.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented an experimental framework to explore and question the contributions of real-time collaborative work in animation. Based on a mono-DCC approach, our Replication Data Protocol (RDP, see section 3.3) supports a non-destructive dataflow thus adapting collaborative possibilities to the specifics of 3D authoring software. With the addition of collaboration-aware functions, we adapted the collaborative workflow to the constraints of the creation software and the needs of the artists.

We evaluated our framework on the efficiency and quality of the collaboration it supports. For this purpose we conducted experimental sessions based on its integration in Blender. It turned out that in addition to speeding up the process of creating 3D scenes, the presented work also increases considerably the visibility of the artists on their ongoing creation.

By adding the real-time aspect within the collaboration we have moved away from the iterative nature of the work. As presented, the framework relies on the current version of the creation data. It would be interesting to consider the validation and the tracking of the artistic work in the context of a scene created in real-time by a team. Adding a versioning layer of the changes could help the production teams to track the team's work over time. This could be achieved by saving a full version (i.e. snapshots) of the repository in different ways. An automatic strategy triggered at a regular time interval meets the backup and security requirements. In contrast, a manual strategy driven by the artist from its DCC software would be ideal to iterate precisely on a scene aspect. The latter would be used to perform validation reviews based on a given state of the work. Currently, the framework stores the entire repository in memory, limiting the size of the supported scenes to the available memory on the client computer. A future work would be to add a disk-based cache system to achieve larger 3D scenes support. By configuring it via the data translation protocol, each type of datablocks would benefit from a caching strategy tailored to its needs.

7 ACKNOWLEDGMENTS

This work has been supported by the French National Association of Research and Technology (ANRT) through an Industrial Research and Formation Convention (CIFRE №2018/0204) established between the company CUBE CREATIVE and the INREV research team of the University Paris 8. And we would like to thank in particular Valentin Moriceau, director of the R&D department of CUBE CREATIVE for his support along the project. We also thank the Blender community for its incredible involvement in the development of the Multi-User add-on.

8 REFERENCES

[Cro90] Crowley T., Milazzo P., Baker E., Forsdick H., Tomlinson R. MMConf: An Infrastructure for Building Shared Multimedia Applications. In Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work, 329-42. CSCW "90. New York, NY, USA: Association for Computing Machinery. 1990. https://doi.org/10.1145/99332.99365.

[Cub21] CUBE CREATIVE Computer Company. http://www.cube-creative.com/.

[Dew92] Prasun D., Choudhary R. A High-Level and Flexible Framework for Implementing Multi-User User-Interfaces. ACM Transactions on Information Systems 10: 345-80. 1992.

[Git17] GitHub. Teletype. https://teletype.atom.io.

[Jun05] Hamano J. Git. https://github.com/git/git.

[Hin11] Hintjens P. Clustered Hashmap Protocol. https://rfc.zeromq.org/spec/12/.

[Hni11] Hnidek J. Network Protocols for Applications of Shared Virtual Reality. Journal of WSCG, vol. 19, pp. 31-38.

[Lau90] Lauwers J.C., Lantz A.K. Collaboration Awareness in Support of Collaboration Transparency: Requirements for the next Generation of Shared Window Systems. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 303-11. CHI "90. New York, NY, USA: Association for Computing Machinery. 1990. https://doi.org/10.1145/97243.97301.

[Mar87] Stefik M., Foster G., Bobrow D. G., Kahn k., Lanning S., Suchman L. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. CACM 30:1, pp. 32-47. January 1987.

[Mar19] Martinez S. Blender Multi-User Add-On. 2019. https://gitlab.com/slumber/multi-user.

[Nvi21] Nvidia Omniverse platform. https://developer.nvidia.com/nvidia-omniverse-platform.

[Pix21] Pixar. Official USD documenation website. http://graphics.pixar.com/usd/docs/index.html.

[Ubi20] Ubisoft. Mixer. 2020. https://gitlab.com/ubisoft-animation-studio/mixer.

[Zep21] Zepwork. DeepDiff. https://github.com/seperman/deepdiff.

Deep Rendering Graphics Pipeline

Mark Wesley Harris
University of Colorado, Colorado Springs
1420 Austin Bluffs Pkwy
80918 Colorado Springs, Colorado
wharris2@uccs.edu

Sudhanshu Semwal
University of Colorado, Colorado Springs
1420 Austin Bluffs Pkwy
80918 Colorado Springs, Colorado
ssemwal@uccs.edu

ABSTRACT

The graphics rendering pipeline is key to generating realistic images, and is a vital process of computational design, modeling, games, and animation. Perhaps the largest limiting factor of rendering is time; the processing required for each pixel inevitably slows down rendering and produces a bottleneck which limits the speed and potential of the rendering pipeline. We applied deep generative networks to the complex problem of rendering an animated 3D scene. Novel datasets of annotated image blocks were used to train an existing attentional generative adversarial network to output renders of a 3D environment. The annotated Caltech-UCSD Birds-200-2011 dataset served as a baseline for comparison of loss and image quality. While our work does not yet generate production quality renders, we show how our method of using existing machine learning architectures and novel text and image processing has the potential to produce a functioning deep rendering framework.

Keywords

Graphics pipeline, rendering technologies, machine learning, image processing, generative adversarial networks, text-to-image, semantic data processing

1 INTRODUCTION

Rendering a 3D environment often requires excessive amounts of time and computational resources. The cost of high quality rendering is multiplied by variable circumstances during production, such as changes to the scene, cast, or script. A determining factor in the speed of rendering is undoubtedly the amount of processing which occurs in the rendering pipeline. We propose the application of text-to-image deep learning networks to the rendering of a 3D environment, to help subvert the costs of traditional rendering processes. We trained the Attentional Generative Adversarial Network [Xu01a] using textual descriptions for small sections of pre-rendered frames. We then generated renders of the environment for a given frame by stitching together outputs generated from the frame's corresponding attributes.

We show how current research may be used in new ways to inject more relevance into generated images, and ultimately produce reasonable renderings of unknown scene data. Although our method does not yet produce images of high enough quality to replace current rendering technologies, we provide a proof of concept to support future advancements for deep learning approaches to the graphics pipeline.

2 BACKGROUND

Deep text-to-image synthesis was made possible through advancements in computer vision and machine learning. Although neural networks have been applied to animation and frame prediction, none so far have attempted to circumvent the rendering process as a whole [Sem01a]. We propose the application of deep generative networks to the rendering of a 3D scene using textual data alone. Two models we found crucial to solving this problem were generative adversarial learning, and attention mechanisms. We utilized an architecture combining both of these concepts, the Attentional Generative Adversarial Network developed by Microsoft Research [Xu01a], in the creation of a deep rendering framework capable of rendering images given textual descriptors of a 3D scene.

2.1 Generative Adversarial Networks

The Generative Adversarial Network (GAN) was first proposed as a method of producing realistic images from random noise [Goo01a]. The GAN is made up of two sub-architectures which are trained in tandem: a generator, *G*, and a discriminator, *D*. The generator is

trained to progressively synthesize images that closely resemble a target, while the discriminator is trained to determine if a given image is real or fake.

A non-conditional GAN model functions as a random image generator, where generated outputs share global and local qualities of images from the training dataset. The advent of the conditional GAN opened new avenues for research in adversarial learning, and has since led to applications in image-to-image and text-to-image translation.

2.2 Attention Mechanisms

Another deep learning concept applicable to text-to-image generation is called Attention. Attention is a type of sequence-to-sequence learning which uses attention functions to reference past data during each iteration of training. An attention function is a mapping of a query and a set of key-value pairs to an output. Attention may be accomplished using an encoder-decoder structure, where the encoder is trained to map feature space $(x_1,\ldots,x_n)$ to a continuous representation $\mathbf{z} = (z_1,\ldots,z_n)$, and the decoder is trained to transform $\mathbf{z}$ into an output sequence $(y_1,\ldots,y_m)$ [Vas01a]. Applied to text-to-image generation, feature space would consist of all input words and the output sequence would consist of 3-channel pixel data. The self-referential nature of attention enables the learning of complex relationships in long data sequences.

3 RELATED WORK

While they perform excellently as random generative models, GANs tend to be unstable, and their outputs often do not fully reflect the diversity in the training set. Over-training of a GAN network may result in preference of a small set of images rather than learning more detailed representations from the dataset. Researchers also found it difficult to train GANs to learn global structures; semantic meaning such as pose, composition, and color are impossible to control without modifying the GAN model. Higher quality images with more stability and semantic relevance were obtained by seeding the model with non-random data – such as a low-quality image or textual description – to generate from [Par01a]. Others found that variation inference and other types of embedding increased the quality of generated images [Luc01a]. These discoveries led to the creation of many variant GAN architectures, such as the Deep Convolutional GAN (DCGAN) [Rad01a].

The DCGAN was first proposed as a way to bridge the capabilities of unsupervised and supervised learning through adversarial training. The model architecture has a similar structure to the original GAN model, but uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively [Rad01a]. The addition of convolutional layers increases abstraction and allows the DCGAN to recognize more complex relationships. The text-conditional convolutional GAN architecture was built off of the successes from the DCGAN, and was one of the first GAN architectures to produce results in adversarial text-to-image synthesis [Ree01a].

Text-to-image generation was notably improved by Microsoft Research with the advent of their Attentional GAN (AttnGAN) model [Xu01a]. The AttnGAN architecture combines inherent inference obtained by the generator and discriminator within the GAN framework with the improvements in dependency correlation and stability from attention mechanisms. Instead of encoding the entire caption as a single input vector, each word in the sequence is allowed an opportunity to claim regional importance for the output image. This ability provides better recognition of conditional sub-regions, which improves overall semantic relevance and coherence of the output image. AttnGAN was partially based on the work of the StackGAN++ architecture, which found a multi-stage approach was necessary for stabilizing sensitivities in training GAN models [Zha01a].

Preceding adversarial training, the AttnGAN architecture loads all textual attributes in the text dataset, and creates a one-hot encoding representation of the data. A one-hot encoding assigns each symbol a bit value of a binary string of size n, where n is the size of the language of all attributes. This creates an easy way for the network to recognize words and ensures a minimal amount of resources are allocated to tracking them. The AttnGAN requires training two models: pre-training for the text and image encoders, and generative training for image synthesis [Xu01a]. Pre-training is contained in the Deep Attentional Multimodal Similarity Model (DAMSM), which initializes the text and image encoders preceding any adversarial training. This step is essential to providing the stability necessary for conditional image generation. Xu et al. used an LSTM model for their text encoder, and Google's Inception-v3 CNN as the image encoder. After pre-training is finished, the encoder paths are used for adversarially training text-to-image generation.

Another architecture was created from the advancements of AttnGAN, called LEarn, Imagine and CreAte GAN (LeicaGAN) [Qia01a]. Similar to other attention-based GANs, the LeicaGAN architecture is comprised of a text-embedding phase followed by coarse-to-fine image generation. The main improvement of LeicaGAN is a result of their use of textual-visual co-embedding (TVE) and multiple priors aggregation (MPA) to further extract semantic meaning from input images.

Finally, we review work from the related field of frame prediction. Mathieu et al. were of the first to apply adversarial learning to predict images in a seqence of

frames. They developed a recursive, multi-scale GAN framework to predict small portions of an input video sequence. Their mult-scale approach provided them the benefits of convolutional layers without lowering the quality of their output images [Mat01a]. The Spatial Transformer Predictor [Fin01a] and Geometry-Based Next Frame Prediction [Mah01a] models deployed a series of convolutional Long Short-Term Memory units to generate the next frame in a video sequence. The former predicted a series of transformations which were masked and then applied to source frames, while the latter predicted a depth map which was then converted into a final image. Liu et al. trained a convolutional encoder-decoder network to predict images between two frames. Their architecture, which they called Deep Voxel Flow [Liu01a], was trained to generate an in-between frame by using voxels borrowed from the two frames adjacent to it. Their approach notably did not require pre-processing, as was used in the prediction methods discussed previously.

4 APPROACH

We propose the creation of a machine learning framework to render frames of a 3D environment given semantic data input as text, a concept we refer to herein as a *deep rendering graphics pipeline*. We chose to use the AttnGAN as the generative architecture, to provide a well-defined baseline for current and future work. Inputs of the architecture were any number of textual attributes containing contextual meaning for the objects which were to be rendered. Once trained, the rendering system was then used to preview new views of the scene without re-rendering using the animation software or re-training the AttnGAN architecture.

The *deep rendering graphics pipeline*, as we define it, requires non-image seed data as input to produce a rendered output image. We note that frame prediction and deep rendering overlap in expectations for final outputs, however given the divergence between text-to-image and image-to-image networks, we classify the two as separate deep learning problem spaces. Although we require both text and image data during training, our final outputs were constructed from text attributes alone. This serves as a function of rendering directly from extracted scene data and does not involve the manipulation or processing of image inputs to generate the final frame, as is done in frame prediction.

4.1 Process

To measure the capabilities of our method, we created a 3D scene with many moving parts and complex lighting interactions. We applied RenderMan [Chr01a] materials with varying properties, including blue glass, soap bubble, gold, obsidian, thick glass, thin glass, beer glass, and murky water. Lighting was applied via a spherical HDRI image of a bright outdoor scene. We animated 4 axes of rotation and 1 telescopic projection over the course of 120 frames, and rendered the animation using RenderMan. These frames were then processed further to create sets of images to be used as training data.

Our datasets of training images were generated by extracting small portions of each frame, which we refer to as "frameblocks". It was clear that including insignificant or redundant information could result in over-training, so we ensured during extraction that elements of the dataset contained minimal similarity to one another. To generate frameblocks for frame F_i, we compared the visual changes between the two adjacent frames, F_{i-1} and F_{i+1}, using a bit-wise XOR and vector sum capped at 255. The result was a grey-scale image which highlighted changed components between frames F_{i-1} and F_{i+1}. While processing each frameblock in the frame, we compared the total amount of change in the block to the sum of all changes in the frame. If the value surpassed the 1.0 percentile of total change, it was included in the dataset, otherwise it was stored in a cache to be compounded in future calculations for the same frameblock. Thus, even small changes over many frames were represented in the image dataset over the course of image pre-processing.

We then extracted corresponding text attributes describing the objects present in each frameblock of the image dataset. Types of attributes we experimented with included screen distance, frame and block index, and vertex, edge, normal, and orientation information. In experimenting with what attributes produced the best outputs, we discovered the AttnGAN model found difficulty in recognizing relationships inherent to numerical data. We concluded that, due to the use of one-hot enocding, the AttnGAN network more easily recognized unique identifiers over numerical values. This is also supported by similar studies of other learning models [Zha02a].

To accommodate the loss in recognition, we attempted to more precisely describe an object's characteristics in relation to its rendered pixels. Each frameblock was divided into $m \times n$ regions, and each region was assigned a bit position. We used these bit positions as flags for marking the area a given 3D object was present inside of. Figure 1 demonstrates this effect for finding the precision attribute of an object in the lower right corner of the sample frameblock. Using the 3×3 grid shown, we see the object resides inside of regions 0x020, 0x080, and 0x100. The logical "OR" operator ($\oplus$) was used to combine these values and create the final precision attribute. Thus, for our example, the precision value was calculated from $0x020 \oplus 0x080 \oplus 0x100 = 0x416$, and resulted in the output attribute of "v416". This process was repeated for all objects present inside of each

frameblock. As we expected, the increase in precision improved recognition and output quality. We experimented with the resolution of precision, from 2×2 subdivisions to 8×8. In general, results did not improve visually past 4×4, so we decided to generate our final results with $m = 4$ and $n = 4$, for a total of 65,536 possible resolution cases.

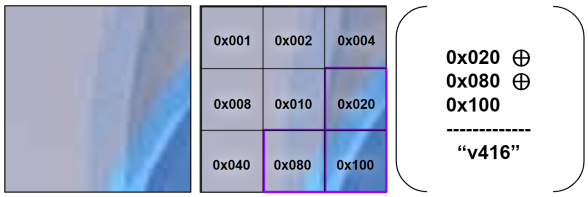


Figure 1: Example of the precision attribute for an object in the bottom right corner of the sample frameblock.

While the precision attribute was able to mask the presence of an object for a given frameblock, it was not able to provide information on what part of the object was in view. We decided to again employ unique identifiers, this time for indexing groups of faces visible in the frameblock. For each object, we assigned an index to the group of face indices exactly containing visible faces. A running list of all face groups was maintained, so that if a given group of faces was seen again, the same identifier was used. This strategy provided spatial relevance while maintaining uniqueness, and was found to be paramount to the visual quality of our output images.

We obtained the best results by prepending the mesh name to each attribute. Recognition also improved with the addition of identifying words; we used "and" to separate attributes connected to the same object, and "also" to identify the end of one attribute group and the start of the next. A sample from the 8×8 dimension dataset is shown in Figure 2, outlining the location of extraction from the source frame in white. The attributes we used included distance to the screen (d), face group identifier (i), precision value (v), and concatenated Euler rotation (r). The text required for this 8×8 pixel sample describing 4 objects was approximately 400 characters in length, while simpler samples with only one object were represented in as few as 30 characters in total.

4.2 Training

To begin training, we generated text attributes for all elements of the corresponding image dataset, as well as for any test frames used in evaluating our framework. We performed two experiments, one using 2 frames of known input, and another using 30 frames. For these experiments, we selected every other frame to be used as a test case. Once the AttnGAN architecture was fully trained, the model was used to generate the first four frames of animation, 2 of which were novel views.

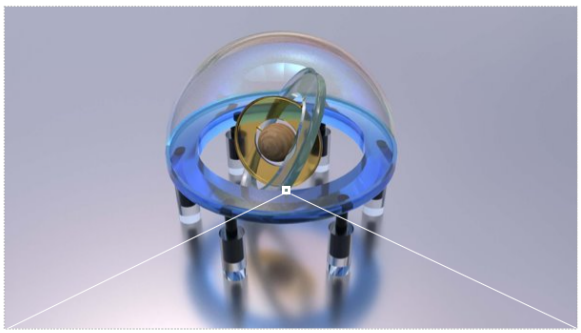


Figure 2: Example of attributes for an extracted 8×8 pixel frameblock of a known frame.

The final stages of our process involved frame generation and analysis. To create a final synthetic frame, we fed in the semantic attributes for all frameblocks of the frame, and combined the generated images in their corresponding block locations.

We experimented with varying frameblock sizes, to gain insight into the efficacy of our text attributes. We generated and tested with frameblocks of dimensions $2^k \times 2^k$ pixels, for $k = 2, 3, \ldots, 6$. We found that using large values of k resulted in smaller datasets which took less time to train, but also generated more visual anomalies and less coherence. Smaller block sizes resulted in very slowly training architectures with higher image quality. Table 1 shows a synopsis of statistics for various frameblock sizes, and Table 2 shows our generated dataset size compared to the total size if all frameblocks were included.

	64	**32**	**16**	**8**
PT (hrs)	0.364	1.216	8.477	17.726
GT (hrs)	1.109	2.727	17.617	34.874
RT (hrs)	0.050	0.167	0.605	2.289

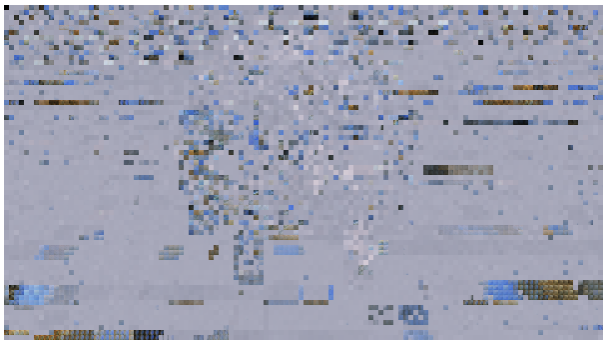
Table 1: Training and evaluation times in hours per dimension of frameblock. **PT** represents pre-training time, **GT** represents generation training time, and **RT** represents final render time for a single frame.

	64	**32**	**16**	**8**
Dataset Size	4,389	17,862	73,741	296,526
Dataset Max	14,400	118,800	241,200	9,720,000

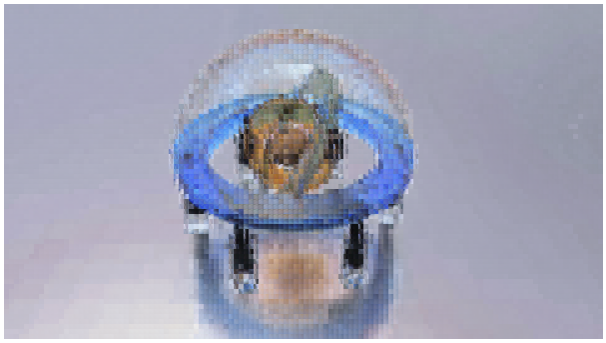
Table 2: Total number of image-text pairs for each dataset compared to the same dataset generated without our selective pre-processing.

In using our method of frameblock selection, we reduced dataset size to 30% of the maximum on

average, and achieved an improvement of 112.8% for SSIM and reduction of 143% in MSE. The model trained on all frameblocks in the frame – in this case a total of 241,200 images – over-trained on the background attributes, which were more abundant per frame than those of any other object in the scene. Our pre-processing phase yielded a dataset of 73,741 unique images and attributes, which resulted in better training and more visual cohesiveness, as seen by the improvements between Figures 3a and 3b. This shows the benefits of our novel use of frameblocks in adversarial text-to-image synthesis.



(a) The frame generated with all attributes present in the frame, including redundant information.



(b) The frame generated using our method of pre-processing.

Figure 3: Visual improvement accomplished using our method of frameblock pre-processing for datasets generated with 16×16 pixel resolution frameblocks.

To serve as a baseline for comparison, we also trained on images of the Caltech-UCSD Birds-200-2011 (CUBS) dataset. CUBS contained 200 categories of bird types with a total of 11,788 images [Wah01a]. The attribute data was represented in multiple variable-length captions for each image. An example caption for an image in CUBS is, "this bird has a bright yellow crown, a long straight bill, and white wingbars" – these captions are much like the attributes used for our datasets of frameblocks, however they are less specific concerning the make up of the image, such as in object placement, pose, and setting. Results from all of our datasets outperformed those of CUBS, which implies that our selection of attributes more accurately represented images compared to the CUBS captions.

5 RESULTS

Since GANs do not optimize any kind of objective function and operate instead on a learned latent space, they can be difficult to analyze analytically [Ric01a]. To evaluate our results, we measured the closeness of our generated images to their target frames via Mean Squared Error (MSE), Structural Similarity Index (SSIM), and Peak Signal-To-Noise Ratio (PSNR). Training losses of all block sizes are produced in Figure 4.

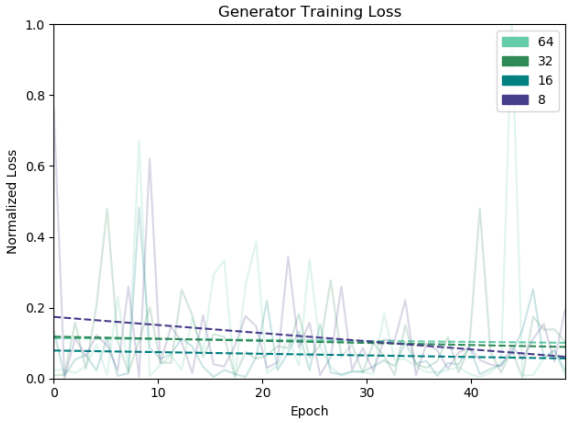


Figure 4: Final training losses for the AttnGAN model trained on frame data.

5.1 Image Analysis

The final generated frames for our 2-frame and 30-frame experiments are shown in Figures 5 and 6, respectively. In both cases, we trained and tested with 4 datasets of block sizes 64, 32, 16, and 8 pixels. We found that the smaller the resolution, the better quality of image was generated. The decreased size in image space improved the ability for the model to learn relationships between pixel value and textual attributes. The larger frameblock sizes of 64 and 32 did not have enough context to produce cohesive outputs. For the 2-frame experiment in particular, these sizes did not contribute enough data to produce arguably good results.

While the model trained on 8 pixel dimension frameblocks produced the best renders of those we generated, it did not generate enough detail or coherence between blocks to constitute a visually accurate rendering. The very center portion, which should display a rotating wooden texture, rendered very little texture in all of our outputs. We note this quality is true for most portions of the image with extremely fine textural details, such as the subsurface scattering on the clear dome and dark glass objects, and the reflection on the floor. This is likely due to the lack of texture and lighting information in the text attributes. Although we theorized that decreasing the block size would improve textural generation, frameblock dimension did not affect the level of

detail generated aside from improving object structure. Further experiments would be necessary before we may determine how to improve the accuracy for rendering textures and lighting using our method of data generation.

Our analysis of output images is produced in Table 3. The values of any given category were mostly uniform across dimensions, which does not trend with the dramatic change in visual quality between outputs. Our further analysis in Table 4, which compares results on a block-by-block basis, reflects greater differences between dimensions. This shows that generated frameblocks were more accurate individually, but scored lower when combined for the final frame. This premise is also supported by the significant improvement in loss over the CUBS baseline. Notably, the most accurate images generated from our datasets were close to or completely correct, while the most accurate images of CUBS exhibited larger amounts of error.

Frames 2 and 4, which were not included in the training or test datasets, still held high correspondence to their targets; the AttnGAN model was able to extract enough semantic meaning from our attributes to create plausible renderings of unknown views, and thus function as a true rendering system. Further, the high similarity between our 2-frame and 30-frame experiments demonstrates the abstraction power of our method. The 64 dimension case of the 2-frame experiment, however, did not contain enough data for learning the semantic descriptions, and thus we conclude our approach is limited by dataset size and training time. Optimizing the number of epochs to data and frames processed would create crisper images, however we found that no amount of training could entirely overcome cross-boundary incoherence or greatly improve the generation of textures. Taking the average of all frameblock sizes, our dataset yielded 156% better MSE, and 130% better SSIM over the CUBS dataset.

Blockdim 64			
Frame	**MSE**	**SSIM**	**PSNR**
1	45.443	0.857	31.556
2	46.304	0.863	31.475
3	45.607	0.859	31.540
4	46.533	0.863	31.453
Blockdim 32			
Frame	**MSE**	**SSIM**	**PSNR**
1	46.341	0.862	31.471
2	47.772	0.859	31.339
3	47.272	0.857	31.385
4	48.513	0.857	31.272
Blockdim 16			
Frame	**MSE**	**SSIM**	**PSNR**
1	42.997	0.874	31.796
2	45.049	0.867	31.594
3	45.273	0.863	31.572
4	45.778	0.864	31.524
Blockdim 8			
Frame	**MSE**	**SSIM**	**PSNR**
1	39.723	0.867	32.140
2	41.894	0.855	31.909
3	40.865	0.857	32.017
4	42.083	0.854	31.890

Table 3: Results of MSE, SSIM, and PSNR for all dimensions and frames analyzed.

6 CONCLUSION

Generative Adversarial Networks show promise for image generation, but can be highly unstable and at times generate undesirable results. Attention-based GAN networks are proven to synthesize higher-quality images with more semantic relevance and improved composition. Using the Attentional GAN model created by Microsoft Research, we successfully implemented a proof of concept for generating frames without a commercial renderer. Our method of frameblock selection reduced dataset size for long animated sequences, and yielded 156% improvement in MSE over the CUBS dataset. While the model trained on the CUBS data generated birds in the likeness of their input captions, fine control of the pose and background was not possible. Our semantic text attributes improved this capability for all block sizes, as we were able to dictate form, material, and structure. The smaller dimensions of 8 and 16 obtained the most accurate renderings, since attributes were more focused for each image. The larger sizes of 32 and 64 failed to provide enough textual detail to generate accurate outputs.

Our results present plausible basis for future work in this area. Replacing the rendering pipeline is a computationally very difficult problem, and would not be possible without the use of advanced deep neural networks. We assert that future advancements in a *deep rendering graphics pipeline* could decrease rendering costs for static and dynamic rendering systems, and ultimately provide an alternative to traditional rendering processes.

7 REFERENCES

[Aug01a] Augusteijn, M., F., Motion generation with a recurrent neural network, in Conf.proc. ICNN'94,

Averages Per Datatype			
Datatype	**MSE**	**SSIM**	**PSNR**
CUBS	105.495	0.392	27.925
Ours (64)	45.971	0.882	34.209
Ours (32)	47.475	0.887	33.310
Ours (16)	44.775	0.899	33.848
Ours (8)	**41.141**	**0.900**	**34.989**
Medians Per Datatype			
Datatype	**MSE**	**SSIM**	**PSNR**
CUBS	106.526	0.349	27.856
Ours (64)	27.068	0.993	33.807
Ours (32)	32.881	0.994	32.961
Ours (16)	26.673	0.996	33.871
Ours (8)	**23.334**	**0.999**	**34.472**
Best Output Per Datatype			
Datatype	**MSE**	**SSIM**	**PSNR**
CUBS	5.184	0.989	40.984
Ours (64)	0.948	0.998	48.366
Ours (32)	0.993	0.998	52.156
Ours (16)	0.245	0.999	54.243
Ours (8)	**0**	**1**	**100**

Table 4: Results per frameblock, with calculations for average and median. The best value for each column of every category is shown in bold.

Vol.5, pp. 2726-2731, 1994.

[Chi01a] Child, R., et al. Generating long sequences with sparse transformers, in CoRR abs/1904.10509, 2019.

[Chr01a] Christensen, P., et al. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering, in Conf.proc. ATG'37, Vol 3, No. 30, 2018.

[Fin01a] Finn, C., Goodfellow, I., and Levine, S. Unsupervised Learning for Physical Interaction through Video Prediction, in Conf.proc. NIPS'16, Barcelona, Spain, Curran Associates Inc., pp. 64-72, 2016.

[Goo01a] Goodfellow, I., et al. Generative Adversarial Nets, in Conf.proc. NIPS'14, Montreal, Canada, MIT Press, pp. 2672-2680, 2014.

[Hua01a] Huang, H., Yu, P.S., and Wang, C. An Introduction to Image Synthesis with Generative Adversarial Nets, in CoRR abs/1803.04469, 2018.

[Liu01a] Z. Liu, et al. Video Frame Synthesis Using Deep Voxel Flow, in Conf.proc. ICCV'2017, pp. 4473-4481, 2017.

[Luc01a] Lucic, M., et al. High-Fidelity Image Generation With Fewer Labels, in Conf.proc. PMLR'19, Long Beach, California, USA, No. 97, pp. 4138-4192, 2019, pp. 4183-4192.

[Mah01a] Mahjourian, R., Wicke, M., and Angelova, A. Geometry-based next frame prediction from monocular video, in Conf.proc. IV'17, IEEE, pp. 1700-1707, 2017.

[Mat01a] Mathieu, M., Couprie, C., and LeCun, Y. Deep multi-scale video prediction beyond mean square error, in CoRR abs/1511.05440, 2015.

[Par01a] Parmar, N., et al. Image Transformer, in Conf.proc. PMLR'18, No. 80, pp. 4055-4064, 2018.

[Qia01a] Qiao, T., et al. Learn, Imagine and Create: Text-to-Image Generation from Prior Knowledge, in Conf.proc. NIPS'19, Curran Associates, Inc., No. 32, pp. 887-897, 2019.

[Rad01a] Radford, A., Metz, L., and Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, in Conf.proc. ICLR'16, San Juan, Puerto Rico, 2016.

[Ree01a] Reed, S., et al. Generative Adversarial Text to Image Synthesis, in Conf.proc. ICML'16, New York, NY, USA, No. 48, pp. 1060-1069, 2016. Adversarial Networks, in CoRR abs/1710.10916, 2017.

[Ric01a] Richardson, E., and Weiss, Y. On GANs and GMMs, in Conf.proc. NIPS'18, Montreal, Canada, Curran Associates Inc., No. 32, pp. 5852-5863, 2018.

[Sem01a] Sudhanshu K., S., A Proposal for using ANNs for CG Animation, in CC-AI: The Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology, Vol. 10, No. 1-2, pp. 93-106, 1993.

[Vas01a] Ashish Vaswani et al. Attention is All you Need, in Conf.proc. NIPS'17, No. 30, Curran Associates, Inc., pp. 5998-6008, 2017.

[Wah01a] Wah, C., et al. The Caltech-UCSD Birds-200-2011 Dataset. Tech. rep. CNS-TR2011-001, California Institute of Technology, 2011.

[Xu01a] Xu, T., et al. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks, in Conf.proc. IEEE/CVF'18, pp. 1316-1324, 2018.

[Zha01a] Zhang, H., et al. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks, in CoRR abs/1710.10916, 2017.

[Zha02a] Zhao, Y., et al. 3D Point Capsule Networks, in CoRR abs/1812.10775, 2018.

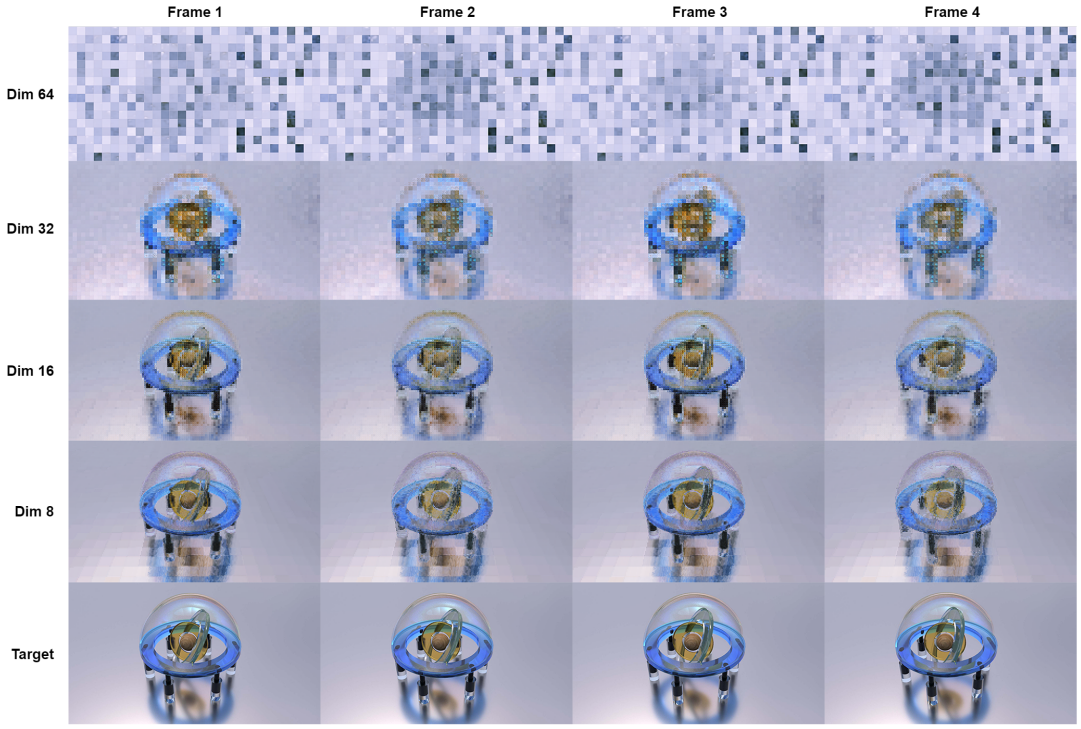


Figure 5: Results of our framework for 64, 32, 16, and 8 dimension frameblocks, using 50 epochs and the first and third frame of the 60 frame animation. The network did not have enough training data for the 64 and 32 dimension cases, and thus more epochs would be necessary to produce rendering given only 2 frames as inputs.

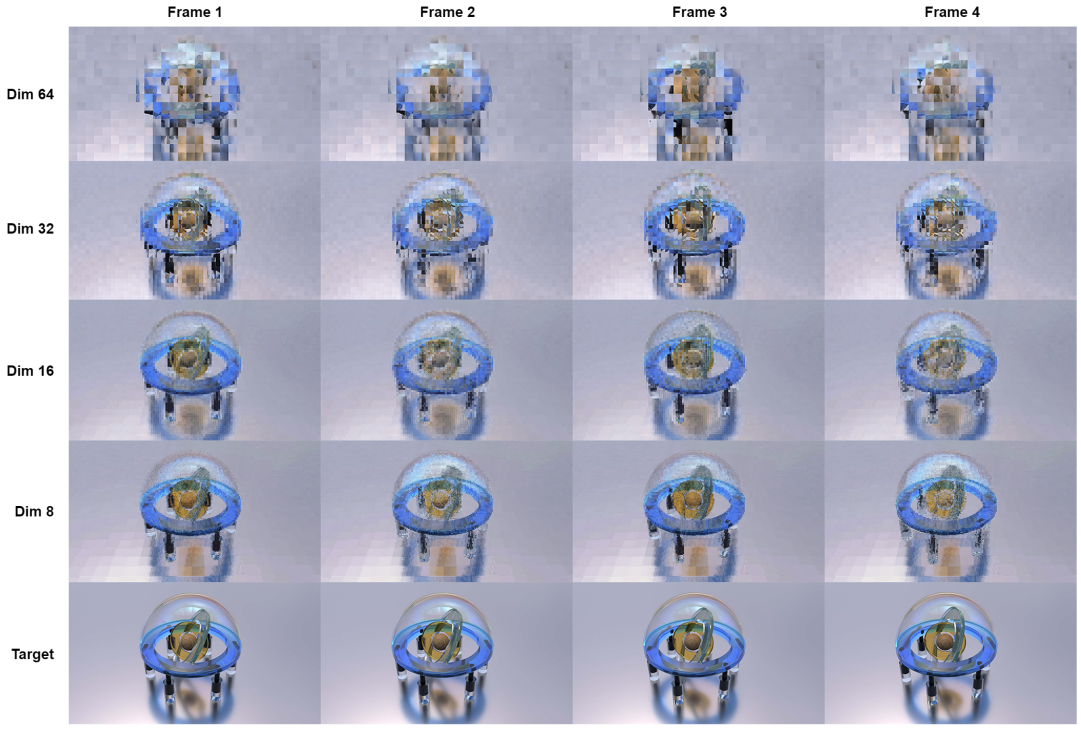


Figure 6: Results of our framework for 64, 32, 16 and 8 dimension frameblocks, using 50 epochs and every other frame of the source 60 frame animation. Frames 1 and 3 were included in the training set, while 2 and 4 were not.

Enhanced Visualization of Customized Manufacturing Data

Olga Kurasova
Mykolas Romeris University
Ateities str. 20
LT-08303 Vilnius, Lithuania
olga.kurasova@mif.vu.lt

Virginijus Marcinkevičius
Mykolas Romeris University
Ateities str. 20
LT-08303 Vilnius, Lithuania
virginijus.marcinkevicius@mif.vu.lt

Birutė Mikulskienė
Mykolas Romeris University
Ateities str. 20
LT-08303 Vilnius, Lithuania
birute.mikulskiene@mruni.eu

ABSTRACT

Recently, customized manufacturing is gaining much momentum. Consumers do not want mass-produced products but are looking for unique and exclusive ones. It is especially evident in the furniture industry. As it is necessary to set an individual price for each individually manufactured product, companies face the need to quickly estimate a preliminary cost and price as soon as an order is received. The task of estimating costs as precise and timely as possible has become critical in customized manufacturing. The cost estimation problem can be solved as a prediction problem using various machine learning (ML) techniques. In order to obtain more accurate price prediction, it is necessary to delve deeper into the data. Data visualization methods are excellent for this purpose. Moreover, it is necessary to consider that the managers who set the price of the product are not ML experts. Thus, data visualization methods should be integrated into the decision support system. On the one hand, these methods should be simple, easily understandable and interpretable. On the other hand, the methods should include more sophisticated approaches that allowed reveal hidden data structure. Here, dimensionality-reduction methods can be employed. In this paper, we propose a data visualization process that can be useful for data analysis in customized furniture manufacturing to get to know the data better, allowing us to develop enhanced price prediction models.

Keywords

Data visualization, dimensionality reduction, machine learning, cost/price estimation and prediction, customized furniture manufacturing.

1. INTRODUCTION

A visualization is a powerful tool in data exploration. A human being is able to understand visually presented information much better than that shown in other forms. Visualization allows data analysts to delve deeper into the data. Good data knowledge enables developing or selecting methods for further data analysis to solve specific tasks, such as classification, prediction, etc. When solving real-world problems, usually, data are of specific nature and complex structure, thus, sophisticated methods for data visualization are needed. Commonly, the real-world data are multidimensional. So, data dimensionality reduction-based visualization methods are widely employed in order to see the general structure of the data. On the other hand, if the developed methods are integrated into a decision support system, and its users are not familiar with data mining and machine learning, visualization tools should be simple, easily understandable and interpretable. The challenge is to reconcile these two aspects of visualization methods. Moreover, the methods must best reveal the characteristics of the data being analyzed, considering the specificities of the domain. In this paper, we propose a data visualization process in order to adapt it to a specialized decision support system for an early price estimation in customized furniture manufacturing when data visualization is used by untrained experts for rapid prediction experiments.

2. CUSTOMIZED FURNITURE MANUFACTURING

Recently, customized furniture manufacturing is gaining much momentum. Consumers do not want mass-produced products but are looking for unique and exclusive furniture. Furniture manufacturing company faced with custom furniture pricing, which must be done quickly, but with sufficient accuracy. Here, machine learning techniques can be employed. The price estimation problem can be solved as a prediction problem using machine learning techniques [Kur21].

Usually, the furniture price consists of two components: the total cost of production C_p and the profit P_p. The price P_w is a sum of these components, $P_w = C_p + P_p$. The cost C_p consists of direct costs and overhead costs. Direct cost includes direct material costs, direct labor costs, and other costs. Production costs, administrative costs, the cost of disposing belong to overhead costs. Usually, the early cost estimation is the most relevant problem [Nia06]. It is a complicated and time-consuming process due to the need to evaluate many components in the early design stage when information is most limited. When the cost is estimated, the profit is added as a certain percentage of the cost.

If machine learning methods are to be used, it is necessary to have data from the domain in question. When analyzing customized furniture manufacturing data, one data item is a product (furniture), characterized by a set of various features: item measurement (length, height, width, weight, the volume of the bounding box); material data (the materials used for production and their costs); operational data (operation list and the time required to complete the operation process); labor data (much manual work, expensive machine tools are used); production time (the customer's requirement for production time); batch size (more the same products reduce the cost of one product); manufacturing complexity (a qualitative parameter indicative of the uniqueness of the item and complexity of the work). Let's denote these features by $x_1, x_2, \ldots, x_m$. They are used as independent variables if regression-based methods are used for the cost prediction. In this case, the cost is a dependent variable. Let's denote it by y.

The cost estimation by machine learning is challenging due to the very wide variety of custom furniture. The prediction accuracy is not as high as one would like. Thus, it is purposeful to apply visualization methods to delve deeper into the data using visualization output as an additional source of information for external experts. The visual analysis will help select appropriate data subsets and data features that should be used in machine learning to improve cost prediction results.

3. DATA VISUALIZATION METHODS

Data visualization is crucial in the big data era. The data visualization helps to notice the totality of the analyzed data, allows for better knowledge of the data, resulting in easier decision making [Med17]. The data visualization can be used for various purposes: initial data cognition, exploratory data analysis [Bat17], interpretation of machine learning [Cha20a], etc. This is especially important if the person working with the data is not an expert in data mining and machine learning. Data should be presented in such a form that they would easily understandable and interpreted. Many methods for data visualization are developed [Sor14], [Wan15], [Dze13]. The methods can be divided into two large groups: direct visualization and dimensionality reduction-based visualization [Dze13]. When using direct visualization methods, the data are presented in a visual form acceptable to a human being. Scatterplots, bar plots, histograms, and others are assigned to this group. Dimensionality reduction is one of the major data abstraction techniques in visual analytics. It aims to represent multidimensional data in low-dimensional spaces while preserving most of its relevant structure, such as outliers, clusters, or underlying manifolds [Sac16].

Principal component analysis (PCA) is the best known and most popular approach for dimensionality reduction. It finds a linear subspace that aims to maintain most of the variability of the data [Wan16]. PCA is a linear projection method, while multidimensional scaling (MDS) is a nonlinear one. MDS aims to find low-dimensional points such that the distances between the points in the low-dimensional space were as close to the proximities of multidimensional points as possible [Dze13]. The t-distributed stochastic neighbor embedding (t-SNE) based on non-convex optimization has become the *de facto* standard for visualization in a wide range of applications [Aro18]. The state-of-the-art t-SNE algorithm manages to create low-dimensional representations that accurately capture complex patterns from the high-dimensional space, showing them as well-separated clusters of points [Cha20b]. Recently, one more group of dimensionality reduction approaches becomes popular—autoencoder neural networks [Wan16]. They can cope with large amounts of data because deep learning strategies are applied to their training.

The problem arises how to select more appropriate visualization methods and how to organize the whole visualization process. In Fig. 1, the proposed data visualization process is depicted. In the beginning, a data set should be formed. After that, feature selection can be helpful. It can be performed manually or applying some feature selection techniques. The next step is data visualization. At first, simple direct visualization methods can be used (scatterplots, histograms, etc.). After that, more sophisticated methods should be employed. As the data to be analyzed usually are multidimensional, dimensionality reduction-based visualization methods can help dive into the hidden structure of the data. As a result, a set of visual presentations is obtained. A data analyst/decision maker should review and interpret the results obtained, i.e., to look for data properties and interrelationships that would help solve the main task of the data analysis—a classification, prediction, etc. If the obtained results are satisfactory, the final decision making can be performed, and the

process is completed. Otherwise, the process is continued selecting a subset of the data set if it turns out that the data items form groups and it is appropriate to analyze individual groups. Then the visual analysis is performed for data subsets. The steps are repeated until the final decision is made.

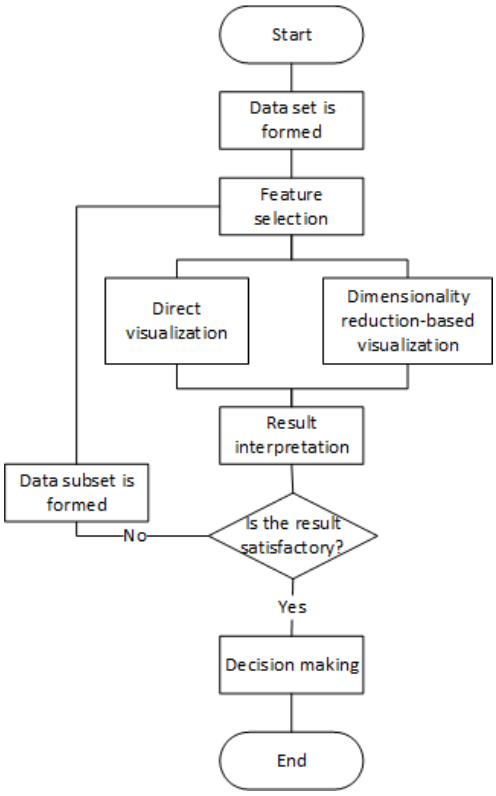


Figure 1. Data visualization process

The proposed visualization processes should be adapted to a domain to be investigated. As mentioned before, our domain is related to customized furniture manufacturing. Thus, historical data on already manufactured products should be collected. The data can be described by metal and wood processing times, various material features, number of different components, etc. A subset of these features is selected. After that, some direct visualization techniques can be used for initial explorative analysis. Scatterplots and histograms are most suitable here. In order to see the general structure of the data, the well-known principal component analysis and multidimensional scaling are irreplaceable. However, more modern methods, t-SNE and autoencoder neural networks, must be used, too. Moreover, integrating cluster analysis into visualization allows us to dive much deeper into the data and notice important insights. Here, we suggest using a specific clustering technique—Louvain algorithm [Tra19]. Suppose the results of the performed analysis on the whole data set do not satisfy the decision maker. In that case, a data subset should be selected, and the visual analysis is repeated for this data subset.

4. VISUAL ANALYSIS

The real manufacturing data for 1007 products provided by a Lithuanian furniture manufacturing company are used in the visual analysis to demonstrate the proposed visualization process. The data gathered over the last five years include the real prices of these products. A set of products includes items of various sizes and complexity (from small pieces of furniture to large furniture kits). Each product is characterized by some features $x_1, x_2, \dots, x_m$. The selected features are described in Table 1, $m = 17$. Here, the price is denoted by y. All the feature values are numerical. Thus, a data matrix can be formed. Let's denote the i^{th} product by X_i. Then we have a $d \times m$ matrix $\mathbf{X} = (x_{ij})$, $i = 1, \dots, d$, $j = 1, \dots, m$. The i^{th} row of this matrix corresponds to the vector $X_i = (x_{ij})$, $j = 1, \dots, m$, $i \in \{1, \dots, d\}$, which elements are the feature values of the i^{th} product, $d = 1007, m = 17$. The data mining software *Orange* is used for data visualization [Dem13] (https://orangedatamining.com). For training autoencoder neural network and visualizing the results obtained, the following *python* libraries are used: *tensorflow*, *keras*, *scikit-learn*, *numpy*, *pandas*, *seaborn*, *matplotlib*.

x_i	**Notation**	**Short descriptions**
$x_1 \dots x_5$	m10-m50	metal processing times
$x_6 \dots x_9$	w60-w90	wood processing times
x_{10}	m	total meters of materials
x_{11}	m^2	total square meters of materials
x_{12}	kg	total weight of materials
x_{13}	qty	total amount of materials
x_{14}	qty_ parts	number of parts
x_{15}	qty_ diff_parts	number of different parts
x_{16}	qty_ materials	number of different materials
x_{17}	qty_ order	quantities (order size)
y	price	price

Table 1. Manufacturing data features

At first, initial data exploration should be performed. Here, we confine ourselves to visual analysis. It is purposefully to see a general structure of data. Histograms or other simple direct visualization techniques are usable for this purpose. In Fig. 2, a histogram of the price is presented. We can see that the data set includes a lot of cheap products. About 700 items are cheaper than 200 Eur. Only a few expensive products are costing more than 1000 Eur. Knowing this information, the data analyst can divide

the data into several groups according to the size of the prices.

The histograms of other data features can be explored to see data similarities and dissimilarities. Due to the limited size of the paper, here, we present only a feature histogram. In Fig. 3, we see a frequency of the number of different parts of furniture. In many cases, the number of different parts is not greater than 20. Only a few pieces of furniture consist of many components.

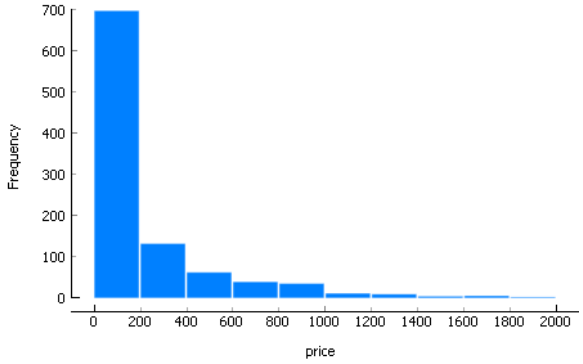


Figure 2. Price distribution

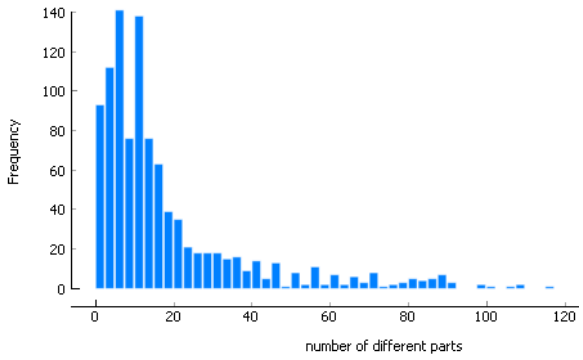


Figure 3. Distribution of different parts of furniture

As the data are multidimensional, dimensionality reduction methods can be employed for data visualization. Various approaches can be used. At first, the data dimensionality is reduced by PCA. If we want to represent the data of low-dimensionality in a 2D Cartesian coordinate system, only two first principal components (PC1 and PC2) can be used (Fig. 4). Here, a point corresponds to a product (a piece of furniture). The points are colored according to the price size. We can see that the points corresponding to cheaper products are huddled in one place. Meanwhile, the points corresponding to expensive furniture are distributed farther apart. It should be noted that two principal components explain only 53% of the variance. In order to get 80%, even seven principal components need. However, in that case, the data cannot be represented visually.

The well-known fact that PCA is a linear dimensionality reduction method. If the data are of nonlinear nature, nonlinear dimensionality reduction methods can be more suitable. Thus, the data are visualized by MDS (Fig. 5). Here the Euclidean distance is used as a data proximity measure. We can see not only accumulations of points but also data outliers. Using PCA, two yellow points were distant from other points. In the case of MDS, two yellow points remain outliers; additionally, two blue points are far away from the majority of the other points.

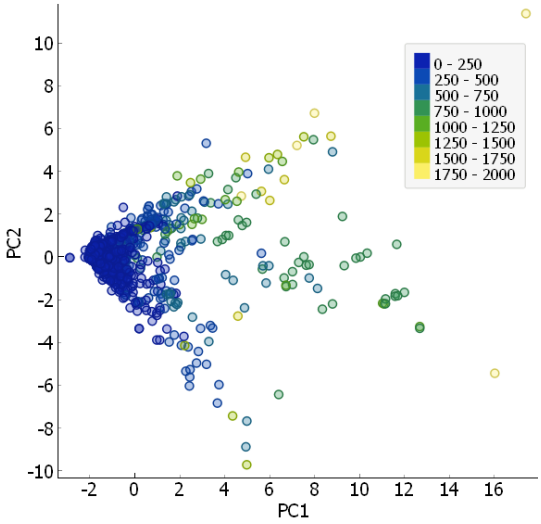


Figure 4. The data visualized by two principal components

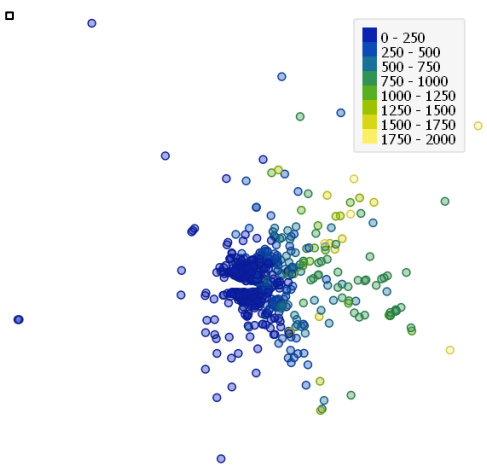


Figure 5. The data visualized by MDS

It is interesting to see how the data are distributed if the state-of-the-art t-SNE is used. The result is presented in Fig. 6. We can see that the data representation differs from Fig. 4 and 5. Here, some data clusters can be observed. One large group of the points is monitored at the bottom of the image. A smaller cluster is obtained at the top of the image. It should be noted that the points corresponding to more expensive products (green and yellow points) form

some clusters. Meanwhile, in the cases of PCA and MDS, no such clusters were obtained. It is worth noting that no clustering method was used here. We interpret cluster formation only by observing the image.

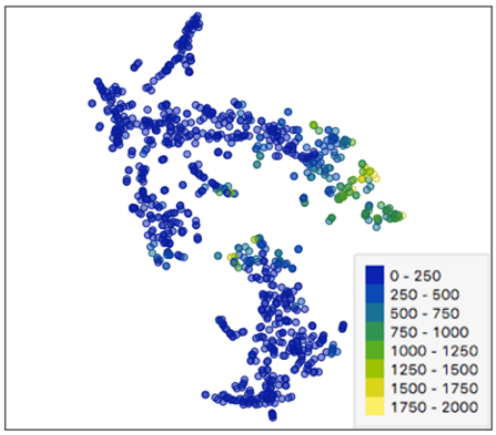


Figure 6. The data visualized by t-SNE

However, clustering methods can also be employed. Data clustering is commonly used in visual analysis. In this investigation, we use a specific clustering technique—Louvain algorithm [Tra19]. It is a hierarchical clustering algorithm that recursively merges communities into a single node and executes the modularity clustering on the condensed graphs. Originally it is proposed for community detection, but it can be used to solve other clustering problems. The furniture data also be clustered by the Louvain algorithm, and after that, they are visualized by t-SNE (Fig. 7). Here points are colored according to the clusters to which they are assigned. The Louvain algorithm automatically identifies six clusters. It is purposeful to see how clusters are related to the price size. Let's take the points of cluster C6 (magenta) and represent them in a scatter plot (Fig. 8). The points are colored according to the price size. We can see that cluster C6 consists of the points corresponding to the furniture cheaper than 250 Eur. In Fig. 8, a few points fell outside of their cluster. It is necessary to review these products and look for the reasons for this. The reasons can be different: some products can be priced in an unusual way, there are inaccuracies in fixing the feature values, etc.

In Fig. 9, the data are visualized by an autoencoder neural network. We can see that the points (blue, orange, and green) corresponding to cheaper products are huddled close together; meanwhile, the points corresponding to more expensive products (red and magenta) are spread far apart.

It should be noted that the set of the analyzed products includes items of various sizes and complexity. There are a lot of small pieces of furniture, but large furniture kits are also included. It is purposeful to perform a visual analysis of the data subsets. Let's take a subset of the products, the price of which is between 100 and 500 Eur. The data of these products are visualized by t-SNE (Fig. 10). Here the clusters are obtained by the Louvain algorithm. We can see three large clusters.

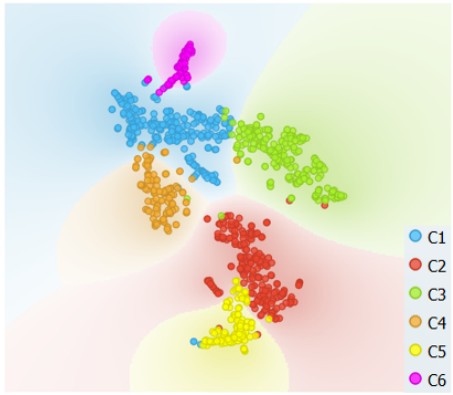


Figure 7. The data clustered by Louvain algorithm and visualized by t-SNE

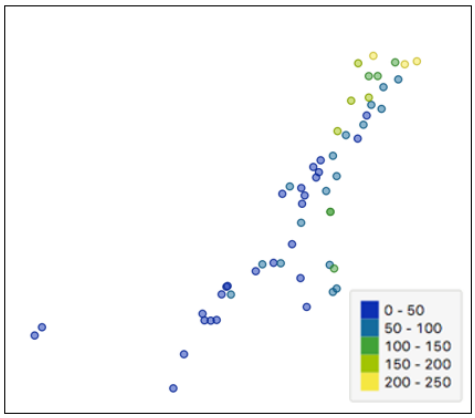


Figure 8. Representation of the cluster C6

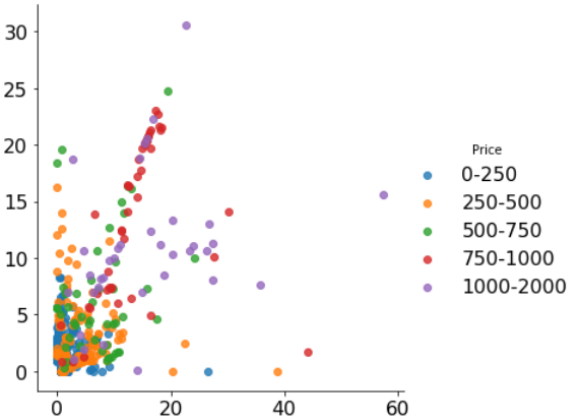


Figure 9. The data visualized by autoencoder neural network

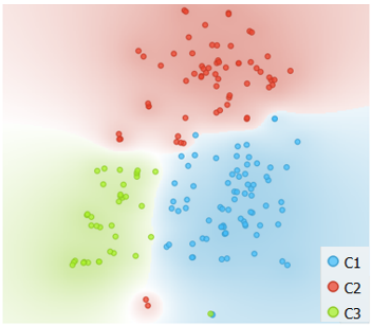


Figure 10. The data subset clustered by Louvain algorithm and visualized by t-SNE

5. CONCLUSIONS

The paper aims to highlight the benefits of visualization in solving data mining tasks. Visual analysis is performed in a very specific domain—customized furniture manufacturing. Here, the main task is to predict the prices of the products before a designed phase. We have proposed a data visualization process that included various visualization approaches that would be useful in explorative data analysis. The approaches will allow decision makers to know the data better and, as a result, will be able to estimate/predict more accurate product prices. Moreover, the proposed process will be implemented into a decision support system. This system is designed for managers and constructors of furniture companies who want to quickly and conveniently evaluate the received individual furniture order and provide the preliminary price of the products. In the system, a machine learning-based module is integrated for price prediction, too. The visual analysis module will enhance this decision support system and ensure its usability and efficiency.

6. ACKNOWLEDGMENTS

This project has received funding from European Regional Development Fund (project No 01.2.2-LMT-K-718-01-0076) under grant agreement with the Research Council of Lithuania (LMTLT).

7. REFERENCES

[Aro18] Arora, S., Hu, W., Kothari, P. K. An analysis of the t-SNE algorithm for data visualization. In Proceedings of the 31st Conference on Learning Theory, PMLR, vol. 75, pp. 1455-1462, 2018.

[Bat17] Batch, A., Elmqvist, N. The interactive visualization gap in initial exploratory data analysis. IEEE Transactions on Visualization and Computer graphics, vo. 24, no. 1, pp. 278-287, 2017.

[Cha20a] Chatzimparmpas, A., Martins, R. M., Jusufi, I., Kerren, A. (2020). A survey of surveys on the use of visualization for interpreting machine learning models. Information Visualization, vol. 19, no. 3, pp. 207-233, 2020.

[Cha20b] Chatzimparmpas, A., Martins, R. M., Kerren, A. t-viSNE: interactive assessment and interpretation of t-SNE projections. IEEE Transactions on Visualization and Computer Graphics, val. 26, no. 8, pp. 2696-2714, 2020.

[Dem13] Demsar, J., Curk, T., Erjavec, A., Gorup, C., Hocevar, T., Milutinovic, M., Mozina, M., Polajnar, M., Toplak, M., Staric, A., Stajdohar, M., Umek, L., Zagar, L., Zbontar, J., Zitnik, M., Zupan, B. Orange: Data Mining Toolbox in Python, Journal of Machine Learning Research, vol. 14, pp. 2349−2353, 2013.

[Dze13] Dzemyda, G., Kurasova, O., Žilinskas, J. Multidimensional data visualization: methods and applications. New York: Springer, 2013.

[Kur21] Kurasova, O., Marcinkevičius, V., Medvedev, V., Mikulskienė, B. Early cost estimation in customized furniture manufacturing using machine learning. International Journal of Machine Learning and Computing, vol. 11, no. 1, pp. 28-33, 2021.

[Nia06] Niazi, A., Dai, J. S., Balabani, S., Seneviratne, L. Product cost estimation: Technique classification and methodology review. Journal of Manufacturing Science and Engineering, vol. 128, no. 2, pp. 563-575, 2006.

[Med17] Medvedev, V., Kurasova, O., Bernatavičienė, J., Treigys, P., Marcinkevičius, V., Dzemyda, G. A new web-based solution for modelling data mining processes. Simulation Modelling Practice and Theory, vol. 76, pp. 34-46, 2017.

[Sac16] Sacha, D., Zhang, L., Sedlmair, M., Lee, J. A., Peltonen, J., Weiskopf, D., Keim, D. A. Visual interaction with dimensionality reduction: A structured literature analysis. IEEE Transactions on Visualization and Computer Graphics, vol. 23, no. 1, pp. 241-250, 2016.

[Sor14] Sorzano, C. O. S., Vargas, J., Montano, A. P. A survey of dimensionality reduction techniques, 2014. arXiv preprint arXiv:1403.2877.

[Tra19] Traag, V. A., Waltman, L., van Eck, N. J. From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports, vol. 9, no. 5233, 2019.

[Wan15] Wang, F., Sun, J. Survey on distance metric learning and dimensionality reduction in data mining. Data mining and knowledge discovery, 29(2), 534-564, 2015.

[Wan16] Wang, Y., Yao, H., Zhao, S. Auto-encoder based dimensionality reduction. Neurocomputing, vol. 184, pp. 232-242, 2016

Autonomous Parking Spot Detection System for Mobile Phones using Drones and Deep Learning

Guilleum Budia Tirado
Department of Computer Sciene
University of Colorado Colorado Springs
Colorado Springs, CO, 80918, USA
guillemgbt@gmail.com

Sudhanshu Kumar Semwal
Department of Computer Sciene
University of Colorado Colorado Springs
Colorado Springs, CO, 80918, USA
ssemwal@uccs.edu

ABSTRACT

Many parking lot facilities suffer from capacity overloads and many times there are no monitoring tools to provide feedback. As a consequence, the people, wanting to park, become frustrated as there is considerably loss of time. In this paper, we present a novel prototype of an automatic parking-lot analysis platform using image-based machine learning to (a) guide a drone autonomously; and (b) to process useful information to be handled into a smartphone application to communicate with the parking lot users.

We have collected a reasonable amount of test images to build a classification model using Convolutional neuronal networks (CNNs) to classify parking lot images, and build different object detection models to identify free and occupied parking spots. Those models have been exported to the back-end module of our platform so it can control the drone and record the computed information to its database. In addition, we have implemented an iOS application that requests and displays the parking lot status and its empty spots.

We have been able to prove that this prototype is feasible, functional, and opens a path towards future improvements and refinements. The flight control and the data classification algorithms have been shown to work using the machine learning models. In summary, we found a clear and and concise way to display useful information in real time to our users.

Keywords

Drones, AR, Parking Lot, Navigation, Deep Learning

1 INTRODUCTION

It is really a universal experience that we have all faced parking lots frustrations, such as spending time looking for an available free spot, or even realizing that there are no spaces after visiting all of the parking spot. There are several related papers: [1, 2, 3, 4]. In most of these cases, static cameras pointing to the parking lot are used, resulting into non-variant images. Then build manually a mask to segment each parking space into the parking lot and then, after segmenting the original image, individual parking lot space images are used as input to a classifier, using mainly Convolutional Neuronal Networks [5] to determine if those are free or occupied. Traditionally, in robotics, sensor information such as LIDAR [9], sonar and other technologies that give proximity information could be used as well. Yet, our drones are the cheapest available in the market and these drones do not provide this kind of sensors. Other approaches are depth maps computed from drone images. One of the most well known models is the Faster Regional Convolution Neuronal Network (*Faster-RCNN*) [6]. The difference between Faster R-CNN and the other R-CNN techniques is that instead of using Selective Search it uses Region Proposal Networks (RPN) [7]. Other object detection approach very different from Faster-RCNN are the *You Only Look Once* (YOLO) models [8]. The research found in [9] concluded that they could autonomously guide a drone through indoor corridors only using the front camera and without any GPS or other sensor information. To detect and classify parking spaces given dynamic images we will use object detection models to localize and identify two classes, the free parking lots and the occupied parking lots. To achieve good results, we will train different object detection models defined in the Tensorflow Object Detection API such as Faster R-CNN or SSD. We would evaluate the goodness of fit for each model and its speed to choose the final models we will export to use in our back-end system. Figure 1 shows the working of our overall algorithm. The central server will be developed with Python and Django framework. This way, our algorithm integrates the Tensorflow models. The server will be responsible for controlling the drone and receive images from it. Furthermore, serve will also use the trained object detection and flight prediction models to process all the data and store useful information and compute accurate flight instructions for the drone. We have also designed and built a simple iOS application developed with Swift that interacts and retrieves information from the server's database and displays it to the user so we can have a complete prototype of the desired system.

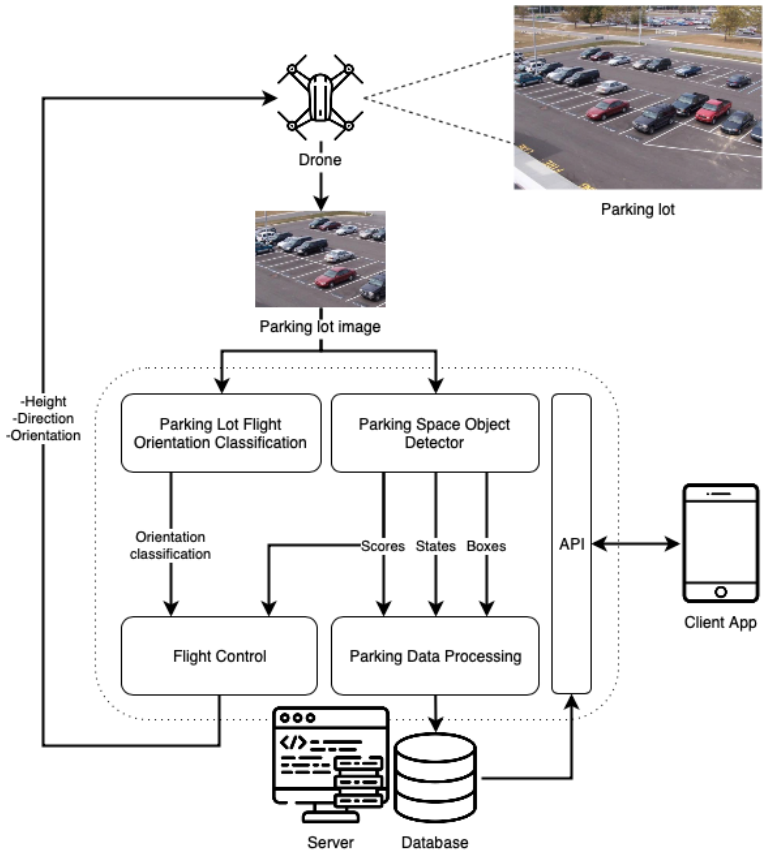


Figure 1: Overall system structure diagram.

2 DATA SETS

All of the machine learning models we wanted to develop receive an input of a parking lot image. That is why we started by collecting them. We created a Python program that could control the drone manually from the computer and display its video stream in the screen. Using different keyboard keys we could fly the drone around parking lot 228, 7 and the different sections of the lot 224. The available instructions to the drone are listed in the Table 2. After 50 takes, we achieved 1129 RGB 960 by 720 images.

Graphics	Top	In-between	Bottom
Tables	End	Last	First
Figures	Good	Similar	Very well

Table 1: Table captions should be placed below the table

Figure 2: Examples of captured images.

This is a binary image classification problem where the positive class is the image that contains a parking lot and the negative are the images that don't. Since it is a supervised machine learning scenario, we manually label the drone images we collected so we can transfer the human experience into observations and transfer knowledge into our classification algorithm. To make things easy, we developed a simple Python script that goes through each captured images placed in the "Takes" directory mentioned earlier, displays each image one by one and asks if that current image is a parking lot or not through the terminal console. After going through the original 1129 drone images, we classified 1096 as valid observations. 737 observations are set as positive parking lot (67,3%) and 358 were found as negative parking lot (32,7%).

Key	**Action**
Up arrow	Move forward
Down arrow	Move backwards
Left arrow	Move left
Right arrow	Move right
W	Move up
S	Move down
A	Rotate counter clockwise
D	Rotate clockwise
T	Take-off
L	Land
C	Start/Stop capturing images

Table 2: Manual drone flight instructions.

In order to retrieve more visual information to guide the drone automatically, and obtain parking spot information from images we implement different parking spot object detection models. And to train them, we need a labelled parking spot object detection data-set. The "PKLot" data set [11] contains nearly 13.000 images of different angle and conditions of the parking lots of the Federal Univeristy of Parana and the Pontical Catholic University of Parana. From the previous mentioned images, nearly 700.000 segmented images of the parking spaces are obtained too. For each parking lot image there is an XML file that contains the position of each bounding box of the parking spaces and the occupancy status.

The bounding boxes have a rhomboid shape as shown in Figure 1. The majority of object detectors work with rectangular boxes aligned with the image axis. That is why we converted each bounding box found on the XML files into a aligned-rectangular shape before being recorded into the CSV file. This method basically chooses min and max points to create a rectangle.

3 CREATING MODELS

In order to guide a drone over a parking lot automatically we will require image information. We decided that one of those key pieces would be a binary image classification machine learning algorithm developed with a Convolutional Neuronal Network. Due to the specifications of our own parking lot classification data-set, the network is required to have as input tensor of 128 by 128 grayscale image. Furthermore it has to be able to be used by our back-end service implemented with Python and Django. The resulting network needs to present acceptable prediction results since it will have a primary relevance in the navigation system of the drone. That means that we are aiming for an accuracy of 90% and above.

There are many different models to choose and perform fine-tuning in the *Tensorflow Object Detection API*. We will test two different families of algorithms. The Faster-RCNN family, known for their speed over the previous RCNN models and its accuracy, and the Single Shot Detector family, known for their speed and real time applications. The difference between Faster R-CNN and the other R-CNN techniques is that instead of using Selective Search it uses Region Proposal Networks (RPN) [7]. This change improves the speed of the network. The first model we have chosen is the faster-rcnn-resnet50-coco pre-trained model since it has a good balance between speed and accuracy between all other Faster-RCNN models found in the library. This model in particular is been pre-trained with the Common Objects in Context (COCO) data-set and uses *Residual Networks* (ResNets) in its implementation [13]. The second model is the faster-rcnn-inception-coco pre-trained model. This model uses the *Inception* architecture [14]. The main idea behind the Inception architecture is that in a same layer of convolution, different filter sizes are being used and then combined to get the output tensor. The *GoogLeNet* CNN developed by Google(TM), found also in [14] uses 1 by 1 filters, 3 by 3 filters and 5 by 5 filters working together in the same layer. This model is also pre-trained over the COCO data-set. Single Shot Detector (SSD) model takes a different approach to the object detection problem [15]. Previous state-of-the-art object detection models were composed by two steps. For each location, K default boxes, called anchors, with multiple shapes and aspect rations are considered. For each anchor, the network will predict the probability of each class for the object that may fall into the box and also four offsets for the anchor to get an accurate bounding box for the object. The first SSD model we have chosen is the **ssd-mobilenet-v2-coco** pre-trained model. It is a balanced model among all the pre-trained SSD models. This model is also pre-trained over the COCO data-set and as a feature extraction architecture uses the one proposed by MobileNetV2 CNN architecture [17], a fast classification CNN developed by Google, found in many portable and real time systems. The second SSD model we will test is the **ssd-inception-coco** pre-trained model. It combines the SSD detection architecture with the Inception feature extraction architecture mentioned above.

Before testing the resulting object detection models, we can observe that the Faster-RCNN ones tend to converge at a lower loss levels than the SSD ones.

Comparing them with other reported loss metrics of other fine tuned models, we can see that are quite low values. It is a sign that our model is being able to fit correctly to the test data. Yet, we can see that the Faster-RCNN models presents less loss than the SSD models. In particular, the Resnet50 version is achieving remarkable results. Besides, among the SSD models, the MobilenetV2 seem to present better results in terms of loss than the Inception one.

4 BACK-END

The central functional piece of our system is the back-end, it handles multiple responsibilities. In the first place in here we define the information models of our system as SQL tables. Furthermore it establishes endpoints so that HTTP clients, such as our iOS application, can retrieve data and send actions to the server and trigger actions. Those two responsibilities are part of the RESTful API block of our back-end. In the other hand, our server is responsible of flying the drone automatically and intelligently by integrating and using the previously exported image-based machine learning models. Besides, given the drone images and the model predictions, it has to be able to process and retrieve

the required information to create and update the required database models so we can serialize them and send them over HTTP requests and provide the clients with the needed information. We have used Python as the programming language since it has many resources and frameworks to support an incredible number of possibilities. In particular, we are using the popular Django framework to easily design the REST API and the structure of the program. In addition, we are using the Tensorflow graph framework to load the exported machine learning models and use them for the drone flight control and the data retrieving from images.

5 INFORMATION MODELS

When creating the database models of information, we need to do a design effort and figure out the basic blocks of information that better represent the problem we want to represent and resolve and how they interact. Django provides a smooth way to define those models. It allows us to define those models as Python classes by sub-classing the *Model* class found in the framework. That subclass will generate the particular SQL table for us. Then, each attribute of that class will map to its table's columns. When creating instances of that class, filling the attributes and calling its *save* method, we are recording a new row in the table. Each model inherits the **id** property that represents the primary key of the table's row.

5.0.1 Lot model

The lot model represents any parking lot we want to monitorize in our system. It consists of metadata information and useful data regarding the state of it. The first attribute, or database field is the **name** given to it as a char field. For example, one of our analyzed lots is names *lot 224* . The **created** and **updated** attributes are *Date* fields representing when the lot has been created and when it has been modified. The *image* attribute is an image field representing the path where a descriptive image of the parking lot is found on the static files directory of the server. The **occupancy** is a Float value representing the percentage of occupancy of the lot recomputed every time new spots are recorded. In a similar way, the **tendency** is a discrete attribute of five possible string values that represent how the occupancy is changing over time, if the lot's occupancy is decreasing, increasing or maintaining given different predictions over time.

5.0.2 Spot model

The spot model represents a single parking spot found in a specific parking lot that we are analysing. This entity also has a **created** date field that represents when the model is created, that is, when it has been detected. Since we want to provide a descriptive image for each spot, we include a *image* field too. A Boolean field named **is_free** is used to determine if the spot is occupied or free. In addition, the Integer field **lot_id** is a *foreign key* that matches each spot with its relative parking lot where it belongs.

5.0.3 FlightState model

The FlightState model represents a more abstract entity in our system. In this structure, we represent the state of the flight algorithm. It is used as a central piece of communication between the HTTP client and the flight algorithm through the REST API. The **state** field represent each possible state the flight algorithm can be. It consists into *LANDED*, *STARTING*, *SCANNING*, *STOPPING* and *ERROR*. The **lot_id** also is the foreign key to the parking lot we are analysing. The **enabled** Boolean flag is manually set if the flight is enabled. And finally the **created** date field is specified too.

6 ENDPOINTS

The endpoints are defined URLs that allow the HTTP clients to interact and request data with the back-end. Each URL is linked with a particular action called *View*. The View is responsible to perform its defined action and return a HTTP response with the data body and status of the request.

7 MACHINE LEARNING MODEL INTEGRATION

As mentioned in previous chapters, we used Tensorflow framework to build, test and export the image-based machine learning models proposed by this project. Now is time to deploy them and make them useful. We exported them in a standard way where in a single file the network structure, operations and weights were defined. This file is called *frozen graph*. To make use of the frozen graphs of our lot classification model and the different parking spot object detection models, the Tensorflow framework must be present in our back-end too. We will use it to recreate the models in memory reading from the disk files. To make our code modular and intuitive to use, we encapsulated the model business logic into Python class definitions and build useful methods so we define a simple interface to work with the image predictions in our algorithms. All the files regarding the model class definition and the actual frozen graphs are placed in a sub-module inside our back-end.

7.1 Lot Classification Model

The first class containing all the classification model logic is called *IsLotCNN*. In the initialization phase of that class the Tensorflow graph is loaded into memory. As a default parameter we pass the path to where the

frozen graph is placed. Once the graph is loaded into memory, we define the input tensors to the network so we can pass new image to predict. The first tensor is the actual **input**, it consists of a tensor of size *[None, 128, 128, 1]*. That means that the network can handle an undefined number of 128 by 128 grayscale images. The second input tensor is the *keep probability* of the dropout technique explained in [9]. This class defines a method to predict the probability of an input image to be a parking lot.

7.2 Spot Object Detection Models

When predicting the parking spots given a drone image, we use the defined method called *detect_drone_img*. This method runs the output tensors *boxes, classes and scores* in the defined session with the drone image as the input value from the input tensor of the network. The output tensors return arrays of boxes (x,y minimum and maximum) defining the bounding boxes of each spot detection, the class of each detection and the confidence score of each detection respectively. Yet, to work easily with the predictions, we defined a class called *PKLotBox* that keeps all the information of a single detection so we can return a single array that each element contains all the detection information instead of a collection of arrays.

8 PARKING LOT DATA PROCESSING

Before handling the flight control algorithm, we decided to prepare first the components that retrieve information from the images and predictions since it is a smaller component that can be easily tested and prepared before facing the main algorithm of the system.

The parking lot data processing module expects that we already have an incoming image and we have already computed the spot detection on that image. Those two elements combined with the previous information in the database regarding the state of the parking lot model and the corresponding spots found in early iterations are the inputs for this process. As for the output, we expect to have registered the new classifications found in the incoming image transformed into Spot database models. Furthermore, once the new spots have been recorded, we need to recompute the new occupancy and tendency of the Lot model we are analyzing.

To compose the Spot models from spot predictions, we are segmenting the original image and storing it into the static files directory so we can access it over HTTP requests, assign this image to the Spot model, retrieving the class of the detection and save the model into the database. Once we recorded all the new spots in the database, retrieve all stored spots given the lot we are analyzing. The new occupancy will be estimated given the ratio of all occupied spots over the total spots. Then, quantizing the difference between the old and new occupancy we obtain the new tendency category.

9 FLIGHT CONTROL – NOVEL ALGORITHM

The flight control algorithm is the core process of our system. It is responsible of the drone guide and communication, including the retribution of images over the UDP video streaming that the drone provides. Furthermore it uses the imported image classification model and multiple object detection models to perform decisions and retrieve information using the defined classes *IsLotCNN* and *PKLotDetector*. Besides it uses the *PKLotDataRetriever* class to obtain information from the spot predictions and update the database. In addition, the algorithm is responsible to update the *FlightState* database model so we can have its information centralised. Also it will check the *FlightState* to know if the HTTP client requested for the flight to stop.

As we mentioned before, this process will be triggered from the action related to the *flight/lot/<int:pk>/start/* endpoint exposed by our REST API module. This means this will be triggered by the client iOS application once we detect that the user wants to know the actual situation of a parking lot. Since the HTTP responses of our back-end have to respond quickly to the client petition, we will need to execute this algorithm in a background thread from our back-end Python process.

9.0.1 Algorithm Overview

Find in Figure 3 the main skeleton of the flight control algorithm. In the first step we initialise all the resources that the algorithm needs, starting with retrieving the needed database models, initialising the machine learning models and setting the drone connection, video streaming and taking off.

9.0.2 Implementation

In Algorithm 1 can be seen the pseudo-code implementation of the **start** method, that is, the main body of the algorithm. It maps the overview algorithm diagram we described above. Yet it delegates important tasks to other functions of the class as explained in the *FlightControl* class diagram to better structure the code. Below we explain the most important methods used by the main algorithm.

The algorithm 2 shows the pseudo-code for the **is_pointing_to_lot(image)** method. It can be seen that we use the lot classification model **cnn** to compute the probability if the input image of being a parking lot or not. Following with the description of internal methods, the Algorithm 3 illustrates how we adjust the initial pose of a scanning iteration when at that point we are not properly pointing to a parking lot. We will do that by performing small and random movements. Afterwards, we will check again if we are pointing to a lot or not. Eventually the drone may find a good position and proceed with the scanning process.

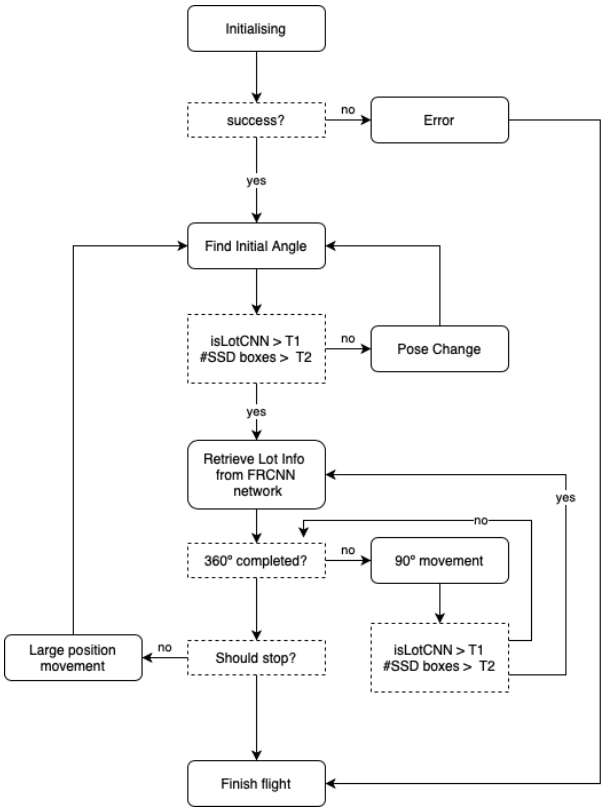


Figure 3: Flight control algorithm diagram

Algorithm 1 Flight control main algorithm (start method)

```
flight_state := get_flight_state();
lot := self.get_lot();
set_up_networks();
set_up_drone();
set_initial_position();
set_state_to(SCANNING);
While True
While not is_pointing_to_lot(image=stream.frame)
adjust_initial_pose();
retrieve_info_from(image=stream.frame);
for angle in [90, 180, 270]  move_to_next_angle();
If is_pointing_to_lot(image=stream.frame) re-
trieve_info_from(image=stream.frame);
should_finish_flight() break
change_initial_position();
finish_flight()
```

First, we will determine the rotation direction, then between 20 and 180 degrees we will find a random rotation angle magnitude. The forward movement magnitude will be found between 200 and 300 cm. The Algorithm 4 shows how the algorithm uses the FRCNN model to predict the spots found in a given image and then pass the image and the spot detections to the *PKLotDataRetriever* module explained above. Finally, in the Algorithm 5 we can see the implementation of how we change to the next initial scanning position after a complete iteration. First, the forward distance is set to 500cm. Then we rotate the drone with a random angle until we are certain that the drone is pointing correctly to the parking lot. This way we make sure that the drone is moving forward in a direction where more probably the parking lot extends.

Algorithm 2 is_pointing_to_lot(image) method

```
image is_lot
lot_prob := cnn.predict_drone_img(image=image);
lots = ssd.detect_drone_img(
image,confidence=ssd_detect_conf);
lot_count := length(lots);
is_lot := (lot_prob >= is_lot_thr) AND (lot_count >=
ssd_count_thr);
```

Algorithm 3 adjust_initial_pose() method

```
rotation_direction := random(0, 1);
rotation_angle := random(20, 180);
forward_distance := random(200, 300);
rotation_direction drone.rotate_clockwise(rotation_angle);
drone.rotate_counter_clockwise(rotation_angle);
```

Algorithm 4 retrieve_info_from(image) method

```
detections := frcnn.detect_drone_img(image, confidence=frcnn_detect_conf)
info_retriever.retrieve_data_from(lot_image=image, predictions=detections)
```

Algorithm 5 change_initial_position() method

```
forward_distance := 500;
WhileTrue
rotation_direction := random(0, 1);
rotation_angle := random(20, 180);
Ifrotation_direction
drone.rotate_clockwise(rotation_angle);
drone.rotate_counter_clockwise(rotation_angle);
Ifis_pointing_to_lot(image=stream.frame) break;
drone.move_forward(forward_distance);
```

9.0.3 Testing the flight control

The Figures 4 through 7 show four real tests over different points of the University of Colorado at Colorado Springs parking lot 224. In each test we tested a different combination of object detection models for the tasks of guide the drone and the retribution of information to see which one is performing better. In each one of the tests a complete scanning loop iteration has been recorded and analysed.In the 90 degree angle analysis the classification CNN detected that it may not be a parking lot even though the SSD model was detecting some spots yet some of them where wrong classification. That image was a hard one to analyse due to the occlusions so the CNN did a good job by skipping it. In the rest of the images the drone was pointing to the parking lot correctly and the CNN gave high probability values for those, meaning that it is working. The SSD MobilenetV2 model is doing a good job by analysing fast the drone images for possible spots. Regarding to the third test found int Figure 6, we are using the FR-CNN Inception model for both information and guide tasks. It is performing a bit slower than the first test since the SSD models compute faster. Yet the timings are acceptable for our task. In this test can be seen that our detection models struggle most with the detection of empty parking spots. Again, the CNN is working as expected by giving a high probability to the lot images and a low one for the images not pointing to it. Finally, the last test found in Figure 7 we are using the FR-CNN Resnet50 model for the data computing task and the FRCNN Inception for the fast spot analysis. Again, using the Resnet50 model is resulting into a non acceptable timing in the scanning iterations. Yet in this test, we are able to detect a good amount of spots, including the empty ones.

10 CLIENT APLICATION

At this point, we have already put all the pieces together to obtain parking lot state information and detecting their spots by controlling the drone automatically, processing its images with our own built machine learning models and recording it to the database. Yet we are missing the piece that would allow to make this system useful by displaying the information we are processing. In our case, this piece is the iOS application. It serves as a client of our back-end and requests data through the defined endpoints to display parking lot information in an intuitive way. To implement it we have used Apple's native tools. We used *Swift 5* as the programming language and XCode as integrated development environment.

11 SUMMARY OF RESULTS AND FUTURE RESEARCH

In total, we tried four different model architectures so we could evaluate their performance and have multiple options at the time of the flight control algorithm implementation. Those models were: two Faster-RCNN family models, one using Residual Networks and the second one using the Inception technique as a feature extractor. The other two models were from the Single-Shot-Detector family, one using MobileNetV2 as feature extractor and the other using also the Inception technique. As a result we achieved really precise models with a higher time complexity, which can only improve with better infrastructure such as faster networks and computers in future. We implemented methods which were usr selected and could be more precise yet are capable of working on real time. So we feel that we have provided good balance in our research. As mentioned earlier, it is not only necessary to build algorithms to retrieve information from parking lots. The resulting model could classify with a 90% of accuracy over the test images and it predicts with an speed of half a second. After deploying that model in the flight control algorithm, the parking lot classification model was able to provide confident predictions, resulting into nice drone navigation results with 90% accuracy on limited training.

12 ACKNOWLEDGEMENTS

We want to thank WSCG anonymous reviewers for providing us feedback so that our paper has improved. In addition, first author can be contacted for code for our applications as our research has benefitted from open source data sets and deep learning algorithms.

13 REFERENCES

[1] Qi Wu, Chingchun Huang,Shih-yu Wang,Weichen Chiu, Tsuhan Chen. ROBUST PARKING

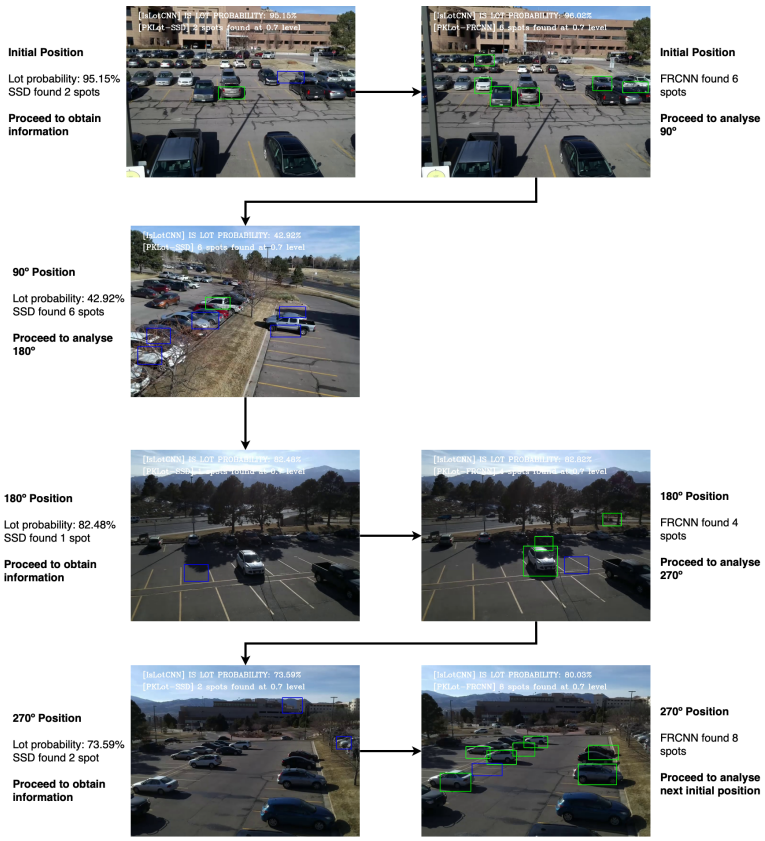


Figure 4: Flight test with FRCNN Inception and SSD MobilenetV2

Figure 5: Flight test with FRCNN Resnet50 and SSD Inception

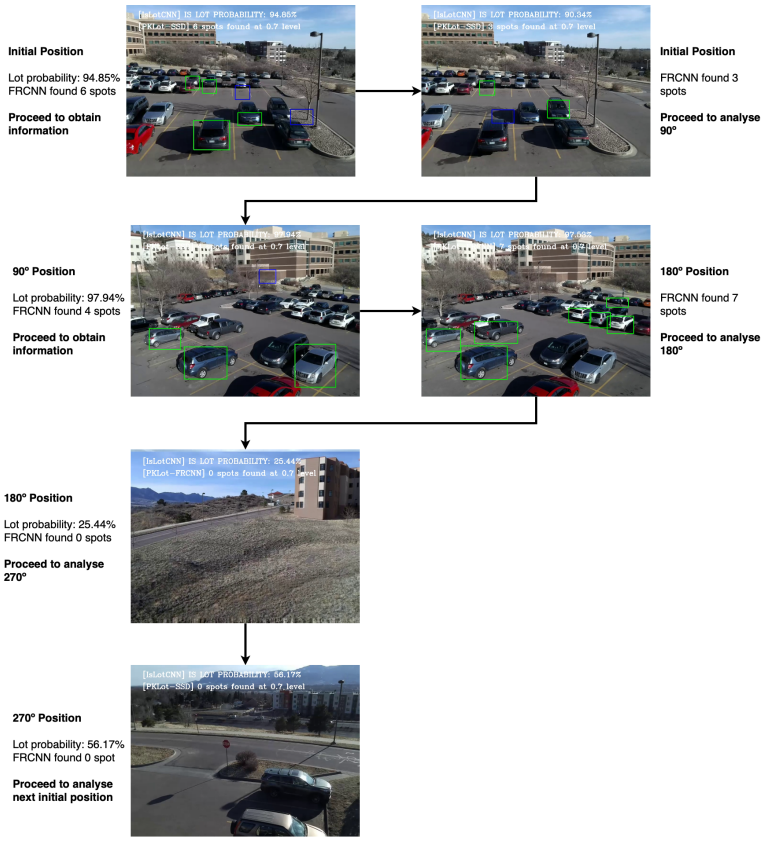


Figure 6: Flight test with double FRCNN Inception

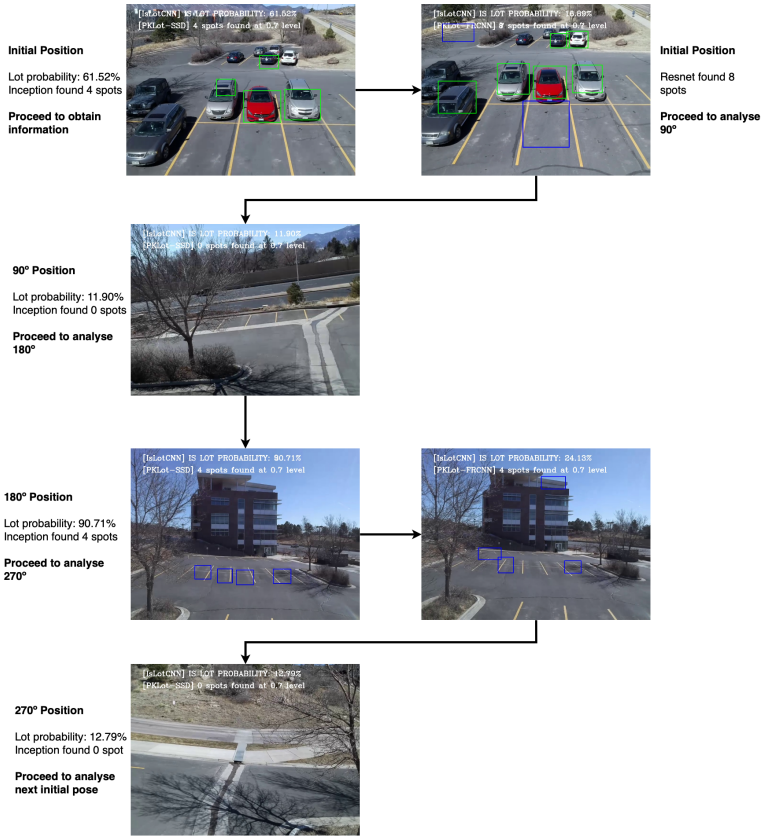


Figure 7: Flight test with FRCNN Resnet50 and FRCNN Inception

Figure 8: Sample of Lot list

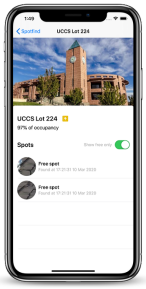


Figure 9: Pull to refresh data

SPACE DETECTION CONSIDERING INTER-SPACE CORRELATION, IEEE International Conference on Multimedia and Expo (2007)

[2] MartinAhrnbom ,Kalle Amstron and Mikael Nilsson. Car Parking Occupancy Detection Using Smart Camera Networks and Deep Learning, IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016.

[3] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro and Claudio VairoCar Parking Occupancy Detection Using Smart Camera Networks and Deep Learning.)2016 IEEE Symposium on Computers and Communication (ISCC), 2016

[4] Julien Nyambal, Richard Klein, Pattern Recognition Association of South Africa and Robotics and Mechatronics, Automated Parking Space Detection Using Convolutional Neural Networks, 2017.

[5] Yunchao Wei, Wei Xia, Min Lin, Junshi Huang, Bingbing Ni, Jian Dong, Yao Zhao and Shuicheng Yan, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, HCP: A Flexible CNN Framework for Multi-Label Image Classification, 2016

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, Microsoft Research, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2016.

[7] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, Xiaolin Hu, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), High Performance Visual Tracking With Siamese Region Proposal Network, 2018.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), You Only Look Once: Unified, Real-Time Object Detection, 2016

[9] Guilleum Budia Tirado, Automatic Parking Lot Detection with Image-Based machne learning, drones, and mobile platforms, MS thesis, Department of Computer Science, University of Colorado Colorado Springs.CO, USA. Advisor: Sudhanshu Kumar Semwal, pp. 1-104, 2020.

[10] Punarjay Chakravarty, Klaas Kelchtermans, Tom Roussel, Stijn Wellens, Tinne Tuytelaars and Luc Van Eycken, IEEE International Conference on Robotics and Automation (ICRA), CNN-based Single Image Obstacle Avoidance on a Quadrotor, 2017.

[11] Widodo Budiharto , Alexander A S Gunawan , Jarot S. Suroso , Andry Chowanda , Aurello Patrik and Gaudi Utama, International Conference on Computer and Communication Systems, Fast Object Detection for Quadcopter Drone using Deep Learning, 2018.

[12] Paulo Almeida, Luiz S. Oliveira, Alceu S. Britto Jr, Eunelson J. Silva Jr Alessandro Koerich, PKLot - A Robust Dataset for Parking Lot Classification, 2015.

[13] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Proceedings of the IEEE, Gradient-based learning applied to document recognition, 1998.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Deep Residual Learning for Image Recognition, 2016.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Going deeper with convolutions, 2015.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Google AI, SSD: Single Shot MultiBox Detector, 2016.

[17] Jonathan Huang, Vivek Rathod, Chen Sun, Speed/accuracy trade-offs for modern convolutional object detectors, 2016.

[18] Mark Sandler Andrew Howard Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen, IEEE/CVF Conference on Computer Vision and Pattern Recognition, MobileNetV2: Inverted Residuals and Linear Bottlenecks, 2018.

Using Autonomous Drone Interactions towards Mobile Personal Spaces for Indoor Environments

Eric Marin Velazquez
Department of Computer Science
University of Colorado Colorado Springs
Colorado Springs, CO, 80918, USA
evelazqu@uccs.edu

Sudhanshu Kumar Semwal
Department of Computer Science
University of Colorado Colorado Springs
Colorado Springs, CO, 80918, USA
ssemwal@uccs.edu

ABSTRACT

We propose an extension of a recent work using convolutional neural networks and drones, such as Learning to fly by using DroNet [8] that can possibly safely drive a drone autonomously. The combination of (i) the DroNet architecture and weights to apply to CNNs to avoid the crashes; (ii) combining it with DLIB tracker, a correlation implemented tracker based on Danelljan et al.'s paper [3] work; (iii) the extraction of descriptors using Speeded Up Robust Features [1]; and (iv) Fast Library for Approximate Nearest Neighbors [10] for the feature matching – leads a drone to track any object and avoid crashes autonomously without any prior information about the object. The main goal is to create a partnership between the drone(s) and the participant as the drone follows the participant and avoids collisions. Our work extends existing methods to also included a way for a drone to follow a person even if the person is hidden for a few frames. Our algorithms also work in low/poor ambient light satisfactorily. In future, our technique can be used to provide novel indoor applications for drones.

Keywords

Drones, Navigation, Deep Learning, Tracking a person

1 INTRODUCTION

A drone, in a technological context, is an unmanned aircraft formally known as unmanned aerial vehicles (UAVs) or unmanned aircraft systems (UASs). Essentially, a drone is a flying robot. These flying robots can be controlled or can fly autonomously using neural networks, for example, working in conjunction with on board sensors, cameras or GPS [**?**]. Our focus is to use machine learning algorithm to control the functionality of drones and extend the personal spaces. State-of-the-art approaches on this topic have already provided models and algorithm to autonomously fly a drone indoor or outdoor avoiding crashes. While such approaches are able to safely fly the drones, they are not using it to provide any other application than just flying the drone. By combining flying with object tracking with drones' navigation we have implemented a new application for drones in our work, and extended drones usage to follow the participant in the indoor spaces. Our efforts can allow the participant to extend there environment and work with drones in a collaborative manner .

2 RELATED WORK

There have been several attempts to use drones effectively in both outdoor and indoor environments [8, 3, 4, 1, 10, 7, 6, 11, 5, 2, 9]. DroNet [8] is a convolutional neural network (CNN), whose purpose is to reliably drive an autonomous drone through the streets of a city.

2.1 Learning to fly by driving

DroNet [8] is trained with data collected by cars and bicycles, this system learns from the collected data to follow basic traffic rules, e.g, do not go off the road, and to safely avoid other pedestrians or obstacles. Surprisingly, the policy learned by DroNet can be generalized. For example, we were able to extend the training to fly a drone in indoor corridors and parking lots. Learning approaches predicts a steering angle and a probability of collision from the drone on-board forward-looking camera. These are later converted into control/flying commands which enable a UAV to safely navigate while avoiding obstacles. Since the goal is to reduce the image processing time, this paper advocate a single convolutional neural network (CNN) with a relatively small size. The architecture is partially shared by the two tasks to reduce the networks complexity and processing time, but is then separated into two branches at the very end: Steering prediction is a regression problem, which means that it requires the prediction of a quantity, while collision prediction is addressed as a binary classification problem. During the training procedure, only imagery recorded by manned vehicles is used. Steering angles are learned from images captured from a car,

while probability of collision is learned from images captured from a bicycle. DroNet system has been tested [8] for autonomously navigation on a number of different urban trails including straight paths and sharp curves as described in citedronet. Moreover, to test the generalization capabilities of the learned policy, they also performed experiments in indoor environments. They compare the approach against two baselines: Straight line policy and Minimize probability of collision policy. One of the metric is to use the average distance travelled before stopping or colliding. Results indicate that DroNet is able to drive a UAV for a long time on almost all the selected testing scenarios. The main strengths of the policy learned by DroNet are twofold:

- the platform smoothly follows the road lane while avoiding static obstacles.
- the drone is never driven into a collision, even in presence of dynamic obstacles, like pedestrians or bicycles, occasionally occluding its path.

2.2 Learning to fly by crashing

In this paper [4], a drone whose goal is to crash into objects is built: it samples random trajectories to crash into random objects. A larger number of crashes are recorded (11,500 times) to create one of the biggest UAV crash related data-set. This negative dataset provides scenarios of different ways that a UAV can crash. It also represents the policy of how UAV should NOT fly. They use all this negative data in conjunction with positive data samples from the same trajectories to learn a simple yet surprisingly powerful policy for UAV navigation. A simple self-supervised paradigm is quite effective in navigating the UAV even in extremely cluttered environments with dynamic obstacles such as humans. Parrot Ar-Drone 2.0 is used. The research is very interesting as no additional sensors/cameras are attached in the flying space during the data collection process. A two-step procedure for data collection is implemented. First, sample naive trajectories lead to collisions with different kind of objects. Then an annotation procedure for collected trajectories is described. The trajectories are first segmented automatically using the accelerometer data. This step restricts each trajectory up to the time of collision. Finally, trajectories are segmented into positive and negative data classification. For the Neural network portion, AlexNet architecture uses the ImageNet-pertained weights as initialization for the network. The network learns how to do a simple classification which uses an input image to predict if the drone should move forward in straight line or not. Based on the right cropped image, complete image and left cropped image, the network predicts the probability to move in right, straight and left direction respectively. If the straight prediction (P(S)) is greater than alpha, drone moves forward with the yaw proportional to the difference between the right prediction (P(R)) and left prediction (P(L)). Intuitively, based on the confidence predictions of left and right, robot is turned to left or right while moving forward. Model's performance is tested using (i) a Straight-line policy, (ii) a depth prediction based policy and (iii) a human controlled policy. The method is tested on 6 complex indoor environments doors, floors, entrance, etc. To evaluate the performance of different baselines, average distance and average time of flight without collisions is used as the metric of evaluation. This metric also terminates flight runs when drones take small loops (spinning on spot). On every environment/setting, this method performed better than the depth baseline. The best straight baseline provides an estimate of how difficult the environments are. The human controlled baselines have better results than their method for most environments. However, for some environments such as hallway and chairs, presence of cluttered objects makes it difficult for the participants to navigate through narrow spaces, and in this case drone method surpasses human level control in this environment.

2.3 DroNet Architecture

DroNet[8] is a convolutional neural network which is able to predict the probability of collision and the steering angle in real time having as the input, each frame of the drones camera. It was trained using data collected by cars and bicycles driving through different cities. It can guide a drone through a road following the traffic signs, avoiding obstacles and without going off the road. To reduce the image processing time, DroNet architecture is shared by the two tasks (probability of collision and steering angle) but then is separated into two branches at the end. Steering angle is a regression problem, which means that requires the prediction of a quantity, and probability of collision is a binary classification problem. DroNet uses the ResNet-8 architecture plus a dropout layer of 0.5 and a reLu non-linearity. The residual blocks of ResNet were proposed by He et al. [6]. After the ReLu layer, the two tasks share more parameters and the architecture splits into two different fully-connected layers. The first output is the steering angle and the second one is the probability of collision. The input of this convolutional neural network is a 200x200 frame in gray-scale.

2.4 Drone control

To control the drone, we use the outputs of DroNet. They used probability of collision to modulate the forward velocity and the predicted steering angle to calculate the yaw angle of the drone. For the forward velocity, the drone fly at a maximal speed when the probability of collision is zero and the drone stops when the probability of collision is close to one. A low-pass filtered version

of the modulated velocity to drive the drone smoothly is used.

$$v_k = (1-\alpha)V_{k-1} + \alpha(1-p_t)V_{max} \quad (1)$$

For the steering angle, a [-1, 1] range is mapped to a desired yaw angle in the range $[-\pi/2, \pi/2]$ and low-pass filtered using:

$$\omega_k = (1-\beta)\omega_{k-1} + (\beta\pi s_k/2) \quad (2)$$

Depending on the environment the value can be changed as well. One of the benefits of using DroNet is that it works perfectly in many indoor and outdoor scenes that contain line features but it fails in environments such as forests because of the lack of this features in the data set.

2.5 DLIB correlation tracker

DLIB tracker is a correlation implemented tracker based on [3]. Their work is based on MOSSE tracker [2]. The MOSSE tracker works well for objects that are translated but it does not work properly for objects that change in scale. DLIB tracker uses a scale pyramid to accurately estimate the scale of an object after the optimal translation is found. This helps us to track objects that change in translation but also in scaling throughout a video stream and furthermore in real-time. The approach works by learning discriminating correlation filters based on scale pyramid representation. They learn separate filters for translation and scale estimation. This scale estimation approach is generic and it can be incorporated into any tracking method with no inherent scale. Incorporating scale estimation makes the computational cost become much higher and their goal was to have an accurate and robust scale estimation approach and at the same time computationally efficient. Their method propose a fast scale estimation approach by learning separate filters for translation and scale. This restricts the search area to smaller parts of the scale space. In order to estimate the scale of the target in an image, [3] uses separate I-dimensional filter. and compares several trackers. The list includes ASLA tracker[7], SCM tracker [**?**], Struck[11], the LSHT tracker [5]. DLIB tracker works well for scaling and translation, and 2.5 times faster than Struck, 25 times faster than ASLA and 250 times faster than SCM in median FPS [**?**].

Obviously, the main advantage of using this correlation tracker in our research is that it will track our selected target and it will compute the translation and the scale of it. This lets our algorithm know either the object is moving left or right and furthermore we will be able to know if the tracking object is moving closer or further from us by just computing the area of the tracking object. While this algorithm helps us in a very prominent and efficient way it has a very big disadvantage which could make our algorithm fail. This disadvantage occurs when the object we are following, completely disappears from the camera field and then it appears again. As this tracker is based in the correlation between frames, it will not be able to detect and track the same object again. To address this problem, we implemented a novel algorithm using descriptors and key points of the tracked object so that when the object we are tracking disappears for a few frames, the algorithm will be able to find it again. This feature will be explained and discussed later.

2.6 Feature extraction: SURF

Speeded Up Robust Features (SURF) is a feature extraction algorithm presented in [1]. It is a faster version of SIFT algorithm [9]. While Scale Invariant Feature Transform (SIFT) uses Lowe approximated Laplacian of Gaussian with Difference of Gaussian for finding scale-space, SURF approximates LoG with Box Filter. The main advantage is that convolution with box filter can be easily calculated with help of integral images and it can be done in parallel for different scales. SURF rely on determinant of Hessian matrix for both scale and location. For orientation, SURF uses wavelet responses in horizontal and vertical direction. Gaussian weights are also applied to it. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window of angle 60 degrees. For feature description, this algorithm uses Wavelet responses in horizontal and vertical direction. A neighborhood of size 20sX20s is taken around the key-point where s is the size. It is divided into 4x4 sub-regions. For each sub-region, horizontal and vertical wavelet responses are taken and a vector is formed. This when represented as a vector gives SURF feature descriptor with total 64 dimensions. Lower the dimension, higher is the speed of computation and matching, which provide better distinctiveness of features. Another important improvement is the use of sign of Laplacian (trace of Hessian Matrix) for underlying interest point. It adds no computation cost since it is already computed during detection. The sign of the Laplacian distinguishes bright blobs on dark backgrounds from the reverse situation. In the matching stage, we only compare features if they have the same type of contrast. This minimal information allows for faster matching, without reducing the descriptor's performance. SURF adds a lot of features to improve the speed in every step. Analysis shows it is 3 times faster than SIFT while performance is comparable to SIFT. SURF is good at handling images with blurring and rotation, but not good at handling viewpoint change and illumination change.

3 OUR ALGORITHM FOR A DRONE TO FOLLOW A PERSON

The most useful feature that we have taken profit from DroNet is the probability of collision because the steering angle must cause some bad prediction. This is caused

Algorithm 1 DroNet usage pseudo code

```
WHILE TRUE{
    frame = drone_camera()
    probability of collision = DroNet(frame)
    WHILE probability of collision > THRESHOLD_PROB {
        STOP Drone movement
        frame = drone_camera()
        probability of collision = DroNet(frame)
    }
    DRIVE Drone
}
```

Figure 1: DroNet Usage pseudo code

Algorithm 2 DLIB tracker usage pseudo code

```
WHILE TRUE {
    frame = drone_camera()
    confidence = DLIB(frame)
    IF confidence < THRESHOLD_TRACK {
        Tracking_object = Lost
        DRONE try to RE-FIND object
    }
    ELSE{
        Tracking_object = Found
        DRONE MOVES FOLLOWING Tracking_object
    }
}
```

Figure 2: DLIB tracker usage pseudo code

because DroNet is trained to follow roads and not objects, by that we mean that the steering angle will follow the road direction and will not follow the object if the object tries to get off-road. Knowing that we have used the probability of collision as basis of our implementation, we are proposing that if the probability of collision is higher than a threshold, the drone should stop immediately. If this probability of collision is lower, then the velocity of the drone will not be calculated based on that probability but based on the velocity of the object we are following. This is a very different feature than the one used in DroNet but since our objectives are very different, our proposal is the best way to add their work into our project.

While the probability of collision is higher than the $THRESHOLD_PROB$, the drone will be stopped at the same position as shown in Figure 1. This is because our goal is to follow anything, for in our example a person. If we are behind this person, following him, we will not have any collision if he has no collision either but we could have a collision if something crosses between the drone and the person. In this case the drone will stop and once this object has crossed, the drone will restart following the person again.

3.1 Using DLIB tracker in our research

The DLIB tracker is a correlation algorithm. So this algorithm does not know what is exactly following does not have memory. This means that if for example, the light goes off and on, the tracker will not know what to follow and it will get lost. This tracker returns a number which tells the confidence the tracker has for following the object correctly. If the object is partially occluded then this probability will decrease. Knowing this number, we can tell when we are tracking our object or when it is lost. Therefore, our algorithm for the tracker is very similar to the Algorithm in Figure 1. If the confidence of the tracker is lower than a threshold, then we will mark our tracking object as lost and we will start trying to find it again. If the confidence of the tracker is high enough, then we will move the drone as the object is moving as shown in Figure 2.

As we are going to follow every object from the same approximate distance, we know exactly which is the distance that separates the drone and the tracking object. We can also, easily know, before losing the object what was the direction it was moving (right or left). By knowing that, we could re-find our object again if our tracker would know what was the object we were following. As our tracker does not know it, we use descriptors and key points to be able to re find the lost object.

3.2 Novel Algorithm: use of descriptors

We extract features, called descriptors, of the object we are tracking. We use this descriptor to re-find the object we were tracking. The basic idea is that we create a buffer of size N where we store these descriptors while we are tracking the object and the confidence of the tracker is high. Once the object has been lost, we use this buffer to check descriptor by descriptor if someone matches with what the drone is capturing. The main point of using a buffer, instead of a single variable is that we want to store several descriptors of the object we are following in case this object changes its form during the following process (due to moving faster or for example sitting down etc). Imagine we are following a person from the back, but then this person turns 180 degrees and he is facing the drone, then we want the descriptors of the person facing the drone and not his back. To be able to do that, our buffer has a fixed size N and every position has a different descriptor value. This value is overwritten once the buffer is full. The only position we do not overwrite is the first position of the buffer where we store the descriptors of the Region of Interest (ROI) of the first frame we used to track because the first ROI will have for sure the descriptors of the object we want to follow without any kind of occlusion. As the drone is sending up to 30 frames per second (fps) and the object we are pursuing will rarely move or change its form that fast we just store 2 descriptors per second. Using this technique, our algorithm becomes more efficient and we

Algorithm 3 Saving descriptors usage pseudo code

```
Buffer size = N
frame = drone_camera()
Once ROI selected from Frame -> Save descriptors at position
Start tracking {
    Frame = Drone_camera
    confidence = DLIB(Frame)
    POS = 1
    IF confidence > THRESHOLD_TRACK {
        Save descriptors of ROI at position POS of the buffer
        IF POS == N { POS = 1}
        ELSE { POS = POS + 1}
    }
}
```

Figure 3: Saving Descriptors Pseudo Code

Algorithm 4 Using descriptors to re-find objects usage pseudo code

```
frame = drone_camera()
Actual_descriptor = descriptor(Frame)
Found = False
FOR descriptor in BUFFER {
    If actual_descriptor matches descriptor {
        Found = True
        BREAK
    }
}
IF Found == True {
    Homography = H(actual_descriptor, descriptor)
    Object_in_frame = Homography(frame)
    Start tracking DLIB(Object_in_frame)
}
```

Figure 4: Using Descriptors to re-find the object Pseudo-code

need less space to store the buffer. The algorithm of storing descriptors is shown in Figure 3.

Saving these descriptors in a good and efficient way is really important to be able to re-find our object once it has been lost. Once the object is lost, our algorithm tries to re-find the object by using the descriptors stored in the buffer position by position and starting at position 0 which is where the initial descriptors were stored. If the object is found, then we start tracking again the object using the DLIB tracker. This is a major improvement proposed towards resiliency of our work, we added descriptors to to re-find the object as shown in Figure 4

4 HANDLING TOUGH SCNEARIOS

The most common tough cases our system could face are:

- No light
- Occlusions
- Very sharp turns

Algorithm 5 General edge cases usage pseudo code

```
Distance_from_object = D (meters)
Last_direction = Right/Left
WHILE DRONE MOVES D (meters) straight {
    IF object found {
        Start tracking again
        BREAK
    }
}
DRONE TURNS Last_direction
IF object found {
    Start tracking again
    BREAK
}
ELSE {
    Keep rotating until object found or 360 degrees
    IF object found {
        Start tracking again
        BREAK
    }
    ELSE {
        LAND Drone
        STOP
    }
}
```

Figure 5: General Edge case usage Pseudo-code

These are handled by implementing (a) Re-finding algorithm and (b) and an algorithm to handle poor lighting conditions.

4.1 Re-finding the object of interest

As we cannot distinguish each of these cases while we are flying, we have implemented a general algorithm to be able to re-find the object. As we previously know the distance between the object and the drone and also the last direction that our object moved, the RE-FIND object algorithm is shown in Figure 5:

Using this algorithm, the drone will be expected to first move straight D meters and then turn either left or right based on that last direction the object moved to find it again. If the object has not been found, then the drone will make a 360 degrees rotation movement to try to find the object again, see Figure 5 for pseudo code. If the object has not been found, then the drone will land and the program will be stopped.

4.2 No light: Novel Algorithm

When there is no light at all, our tracking algorithm confidence value is NaN (Not a Number). Therefore, our implementation is to set the probability of collision to 1 when the tracking algorithm value is NaN. Using this approach, our drone will stop when the light is off and then re-start the tracking when the light is on as shown in Figure 6.

Algorithm 6 No light pseudo code

```
WHILE TRUE{
    Frame = Drone_camera
    confidence = DLIB(Frame)
    IF confidence == NaN {
        STOP drone
        SET probability of collision = 1
    }

    ELSE {
       Start tracking again
    }
}
```

Figure 6: No light Pseudo code

5 OVERALL WORKING OF OUR ALGORITHM

To control the drone, we have to combine all the algorithms we have explained before into one single algorithm. The most relevant parameter we have to check is the probability of collision. If it is higher than the threshold, then we will stop the drone movement until this probability has decreased. No matter what is happening, we do not want our drone to crash. The second parameter we have to take into consideration is the DLIB tracker confidence. If this confidence is higher than the threshold, then we will follow the object while we save the correspondent descriptors. If it is lower than the threshold, then we will stop saving descriptors and we will start to find the object again. If it is found, then we will start the tracking movements again, if it is not after completing the edge-cases movements, we will land the drone and stop the program.

Some parameters can change when we follow different objects. It is not the same to follow a human than a bike or even a car. These parameters are initialized by the user. These parameters are the following: Distance between the drone and the object; Height of the drone; Velocity of the drone. Complete drone algorithm implemented by us is shown in Figure 7 as a Pseudo code.

6 IMPLEMENTATION AND RESULTS

We have programmed everything using Python and PyCharm as the IDE. The libraries we have used for this project are the following: AR Parrot 2.0, DJI Tello python API, OpenCV, DLIB: main tracker API, and Tensorflow.The most useful feature that we have taken profit from DroNet is the probability of collision because the steering angle must cause some bad prediction. This is caused because DroNet is trained to follow roads and not objects, by that we mean that the steering angle will follow the road direction and will not follow the object if the object tries to get off-road. Knowing that we have used the probability of collision as main feature of our project. we are proposing that if the probability of collision is higher than a threshold, the drone should stop immediately. If this probability of collision is lower, then the velocity of the drone will not be calculated based on that probability but based on the velocity of the object we are following. This is a very different feature than the one used in DroNet but since our objectives are very different, our proposal is the best way to add their work into our project. While the probability of collision is higher than the $THRESHOLD_PROB$, the drone will be stopped at the same position. This is because our goal is to follow anything, for example a person. If we are behind this person, following him, we will not have any collision if he has no collision either but we could have a collision if something crosses between the drone and the person. In this case the drone will stop and once this object has crossed, the drone will restart following the person again. This DLIB tracker returns a number which tells the confidence the tracker has for following the object correctly. If the object is partially occluded then this probability will decrease. Knowing this number, we can tell when we are tracking our object or when it is lost. Therefore, our algorithm for the tracker is very similar to the Algorithm 1. If the confidence of the tracker is lower than a threshold, then we will mark our tracking object as lost and we will start trying to find it again. If the confidence of the tracker is high enough, then we will move the drone as the object is moving. As we are going to follow every object from the same approximate distance, we know exactly which is the distance that separates the drone and the tracking object. We can also, easily know, before losing the object what was the direction it was moving (right or left). By knowing that, we could re-find our object again if our tracker would know what was the object we were following. As our tracker does not know it, we use descriptors and key points to be able to re find the lost object. We extract features, called descriptors, of the object we are tracking using the algorithms we have developed. We use this descriptor to re-find the object we were tracking. The basic idea is that we create a buffer of size N where we store these descriptors while we are tracking the object and the confidence of the tracker is high. Once the object has been lost, we use this buffer to check descriptor by descriptor if someone matches with what the drone is capturing.

The main point of using a buffer, instead of a single variable is that we want to store several descriptors of the object we are following in case this object changes its form during the following process (due to moving faster or for example sitting down etc). Imagine we are following a person from the back, but then this person turns 180 degrees and he is facing the drone, then we want the descriptors of the person facing the drone and not his back. To be able to do that, our buffer has a fixed size N and every position has a different descriptor value.

Algorithm 7 Drone control pseudo code

```
Distance_from_object = D (meters)
Last_direction = Right/Left

WHILE True{
    Frame = Drone_camera
    Probability of collision = DroNet(Frame)
    IF Probability of collision > THRESHOLD_PROB{
        STOP DRONE MOVEMENT
        }
        ELSE{
                Confidence = DLIB(Frame)
                IF confidence == NaN {
            STOP drone
            SET probability of collision = 1
        }
                ELIF Confidence < THRESHOLD_TRACK {
                        Tracking_object = Lost
                        WHILE Tracking_object == Lost {

                                WHILE DRONE MOVES D (meters) straight {
                                        find_descriptors()
                                        IF object found {
                                                Tracking_object = Found
                                                BREAK
                                        }
                                }
                                DRONE TURNS Last_direction
                                find_descriptors()
                                IF object found {
                                        Tracking_object = Found
                                        BREAK
                                }

                                ELSE {
                                        Keep rotating until object found or 360 degrees
                                        find_descriptors()
                                        IF object found {
                                                Tracking_object = Found
                                                BREAK
                                        }
                                        ELSE {
                                                LAND DRONE
                                                STOP
                                        }
                                }

                        }

                }
                ELSE{
                        Tracking_object = Found
                        DRONE MOVES FOLLOWING Tracking_object
                        Last_direction = Right/Left
                        save_descriptors()

                }

        }
}
```

Figure 7: Drone Control Pseudo-code

This value is overwritten once the buffer is full. The only position we do not overwrite is the first position of the buffer where we store the descriptors of the Region of Interest (ROI) of the first frame we used to track because the first ROI will have for sure the descriptors of the object we want to follow without any kind of occlusion. As the drone is sending up to 30 frames per second (fps) and the object we are pursuing will rarely move or change its form that fast we just store 2 descriptors per second. Using this technique, our algorithm becomes more efficient and we need less space to store the buffer. Saving these descriptors in a good and efficient way is really important to be able to re-find our object once it has been lost. Once the object is lost, our algorithm tries to re-find the object by using the descriptors stored in the buffer position by position and starting at position 0 which is where the initial descriptors were stored. If the object is found, then we start tracking again the object using the DLIB tracker.

7 DRONE CONTROL

The DJI Tello is a newer and smaller drone but it is also cheap and robust. As it is newer, the implementation of the python API is more efficient and more reliable. The most important specifications for the DJI Tello drones are: Weight: 0.2 lbs; Dimensions: 3.86 x 1.61 x 3.64 inches; Max flight time: 13 minutes; Battery: 1100 mAh LiPo; Camera: 5MP camera 720p, 30 fps video output. The drone has three axes to control: Pitch (Y axis), Yaw (Z axis) and Roll (X axis). If the aircraft rotates around the Pitch axis it will move in the X axis direction. If the Pitch angle is positive, the direction will be backwards, or in the negative X axis. The same will be applied when rotating the Roll axis, which will move the aircraft in the Y axis direction. If the pitch value is positive, the drone will move forward and if it is negative the drone will move backwards as we can see in the following image. If the yaw value is positive, the drone will rotate in clockwise direction on itself and if the yaw is negative, the drone will rotate counter clockwise on itself. If the roll value is positive, the drone will move right and if it is negative the drone will move left as we can see in the following image. There is another drone variable which has to be controlled, the throttle. This controls the aircraft's average thrust from its propulsion system. When the aircraft is level, adjusting the throttle will move the aircraft up or down as all the thrust is in the vertical direction. However, when the aircraft is not level (has non-zero pitch or roll), the thrust will have a horizontal component, and therefore the aircraft will move some horizontally. A larger pitch or roll angle will result in more horizontal thrust and therefore faster horizontal movement. To control this axes of the drone autonomously, what we want is to have always our tracking object in the middle of the frame. If the object we are tracking is in the green zone of the

Figure 8: Simple walk result 3 (left) and 4 (right)

frame, we will not move the drone. If the object is in any blue zone, we will set the throttle of the drone to either go up or down. We always set the roll angle to 0 as we do not want the drone to move right or left but in case the object is in any white space, we will set the yaw in order to rotate the drone to the correct direction. To be able to set the drone's pitch to move the drone forward or backward, what we do is to calculate the area of the tracking object at the beginning of the tracking and if this area increases, we move the drone backwards because it means that the object is approaching the drone, if this area decreases, we move the drone forward because the tracking object is moving further. The yaw, pitch and throttle velocities are changed depending on the environment and the distance between the drone and the object has to be defined by the user in order to be able to run the re-finding algorithm when the object is lost.

8 RESULTS: TEST CASES FOR ALGORITHMS DEVELOPED

The test cases we are going to test are going to follow three different things/objects: a human, a bike and a car. Each of the test cases are going to have different situations which its difficulty will have a range between easy, medium, hard and extreme as explained in the following table. Also, we show several frames which were recorded by the drones. Test cases can be divided based on situation and difficulty: (i) Human walking with no interference is easy; (ii) Human walking with a probability of collision is harder; (iii) Human walking with interference is of medium complexity; (iv) Human walking when person disappears and then appears is of extreme difficulty; (v) Human walking with sharp turns is harder; (vi) Low light is of medium complexity, and (vii) Light on and off is harder. We have implemented all these cases with success as the following images and results show below. Two video sequences submitted with this paper also show that these algorithms have been successfully implemented.

8.1 Human walking: no interference

This is the most basic test case. We want our drone to follow a human while he is walking and there are no interference in the tracking. By that, we mean that there are no objects crossing between the drone and the human.

8.2 Human walking: with interference

This test case is the same as the last test case but with objects/persons crossing between the drone and the hu-

Figure 9: Walking with interference 5 (left) and 6 (right)

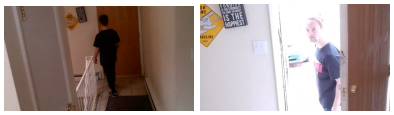


Figure 10: Probability of collision 3 (left) and right (7).

Figure 11: Disappear and appear 4 (left) and right (7)

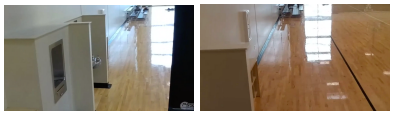


Figure 12: Sharp turns 5 (left) and 6 (right)

man. Having this interference will force our algorithm to re find the human if the obstacles occlude him.

8.3 Human walking: probability of collision

We want to test if our drone stops moving when the probability of collision is really high and if it starts tracking again when this probability of collision becomes lower.

8.4 Human walking: disappear/appear

When the tracking object disappear and appears again we ant to make sure we recognize it as the object we were following before. We wan to test how we apply the descriptors algorithm against this situation.

8.5 Human walking: sharp turns

Sharp turns are really difficult to overcome. The sharper the turn is the more challenging is to follow the tracking object.

8.6 Low light

We also want to test our algorithm again adverse conditions and one of them is the low ambient light. The tracker will need to track the human with low light and this could cause some problems.

Turning the light off could cause a lot of problems calculating the probability of collision, tracking the object and saving descriptors. We want to test our system against these difficulties.

9 CONCLUSION

In this paper we implemented a new way to track and follow objects in real time with no previous information

Figure 13: Low light recognized the edges of a mobile phone.

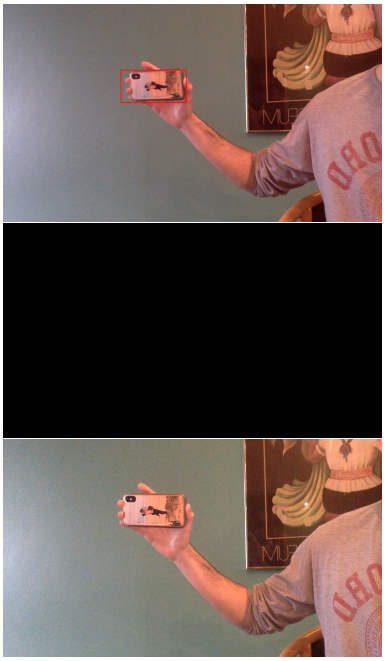


Figure 14: Light on/off 3

about that object before. This mean that the user selects the object to track in real time. As the real time feature and the lack of delay between what the user is seeing and what the drone is doing were some of our priority goals, we used a multi-threading approach in order to implement our algorithm. We found a new application to apply DroNet and the probability of collision. We have also introduced a new technique on how to use features descriptors. Storing descriptors in a buffer and then using this information in case that the object is lost is a new and efficient way to re-find lost objects which worked for us successfully as can be seen by two video sequences we have submitted with this paper. There has been a huge trade off between the tacker and the probability of collision. As our primary requirement was not to crash the drone, no matter what, the probability of collision predominates the control of the drone and this decision could lead to sometimes loosing the tracking object in some cases. In future, we would like to extend our ideas to providing interactive spaces where drone-interactions can lead to novel aapplications.

10 REFERENCES

[1] H Bay, T Tuytelaars, and L Van Gool. 2006. SURF:

Speeded Up Robust Features. In *In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision ? ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11744023*$_3$*2.ECCV*.

[2] DS Bolme, JR Beveridge, BA Draper, and YM Lui. 2010. Visual Object Tracking using Adaptive Correlation Filters. In *CVPR*. ACM, 2544–2550.

[3] Martin Danelljan, H Gustav, S. Khan, and Michael Felsberg. 2014. Accurate Scale Estimation for Robust Visual Tracking. In *Proceedings British Machine Visual Conference*. BMVA Press, 1–11.

[4] Dhiraj Gandhi, L Pinto, and A Gupta. 2017. Learning to Fly by Crashing. In *IEEE RSJ International Conference on Intelligent Robots and Systems (IROS)*. IROS, 3948–3955.

[5] Sam Hare, Amir Saffari, and Phillipe Torr. 2013. Visual tracking via locality sensitive histograms. In *CVPR*. IEEE, 1–8.

[6] H He, Q Yang, R Lau, J Wang, and M Yang. 2012. Deep Residual Learning for Image Recognition. In *CVPR*. ACM, 1–12.

[7] Xu Jia, H Lu, and Yang Ming-Hsuan. 2012. Visual Tracking via Adaptive Structural Local Sparse Appearance Model. In *IEEE Conference on Computer Vision and Pattern Recognition*. ACM, 1–2.

[8] Antonio Loquercio, Ana Maqueda, Carlos del Blanco, and David Scaramuzza. 2018. DroNet: Learning to Fly by Driving. In *IEEE ROBOTICS AND AUTOMATION LETTERS, vol. 3, no 2 April*. IEEE, 1088–1095.

[9] DG Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. In *International Journal of Computer Vision vol 60, issue 2*. Springer, 91–110.

[10] M Muja and David Lowe. 2014. FLANN - Fast Library for Approximate Nearest Neighbors. In *Pattern Analysis and Machine Intelligence (PAMI) vol. 36,*. IEEE, 1–14.

[11] W Zhong, H Lu, and Ming-Hsuan Yang. 2011. Struck Structured output tracking with kernels. In *ICCV*. ACM, 263–270.

Point-to-Block Matching in Depth Estimation

Dawid Mieloch Dominika Klóska Magdalena Woźniak
dawid.mieloch@put.poznan.pl

Institute of Multimedia Telecommunications, Poznań University of Technology
Polanka 3, 61-131 Poznań, Poland

ABSTRACT

In this paper, the novel correspondence search method called point-to-block matching was proposed. Recently, in many proposed multimedia systems, depth estimation is performed on compressed input views. To address this problem and increase the quality of such depth maps, in the proposal, a point in a view is not compared simply with a point in another view, but to the most similar point in a small block surrounding it. The introduction of this method in the depth estimation process is beneficial to the quality of depth maps, as it decreases the influence of small shifts in images, caused e.g., by encoding-related errors introduced to input views. The method was implemented in one of the state-of-the-art depth estimation methods and tested in series of experiments. Based on the comparison of synthesized virtual views with the input views, the proposal increases the quality of estimated depth maps in most of the tested configurations.

Keywords

Depth maps, depth estimation, inter-view correspondence, similarity metric, virtual view synthesis.

1. INTRODUCTION

The methods of creating natural three-dimensional content are recently under constant development, as the emergence of new applications of immersive media systems such as free-viewpoint television [Tan12] and virtual reality systems can be easily seen [Dom17] [Laf16].

One of the most common representations of such multimedia content is the use of depth maps that correspond to each view captured by a camera system and show the distance from the acquired 3D point to the camera [Mül11]. Depth maps can be used to synthesize virtual views, so the user can freely move and see an acquired three-dimensional scene from any viewport.

In its essence, the depth estimation methods are based on the inter-view correspondence search. The result and efficiency of this search are dependent on a similarity of a point in one view of the scene to a point that most probably shows the same part of the scene in another view [Tip13]. Unfortunately, this search can be very difficult due to factors that we will summarize below.

Recently, in many proposed multimedia systems, the depth estimation is proposed to be performed using compressed input views, e.g., in the representation server that sends the requested virtual view to the user [Dzi16], or directly, in the client-side decoder of the multi-view image encoder [Gar19]. Therefore, the depth estimation methods should take into account that some encoding-related errors can be present in used videos.

The comparison of uncompressed and compressed input view can be seen in Figure 1, while Figure 2 shows the difference between uncompressed and compressed images, unveiling artifacts introduced by the high compression. The main differences are clearly visible in highly-textured areas and all edges visible in the scene.

Figure 1. Input view of Fencing sequence [Dom16] without compression (A) and with high (B) / low (C) compression.

The presence of high-frequency harmonics is necessary for the correct representation of the edges in the image, unfortunately, due to coding, when the

transform coefficients are quantized, high-frequency components are notched out, which in turn leads to blurring or, in the worst case, an edge shift in the image. Such shift can also occur in highly compressed videos because of the high share of temporally predicted parts of videos [Leo07].

Figure 2. The logarithm of difference between compressed and uncompressed input view of Fencing sequence for high compression. The contrast of the image was increased to increase the visibility of differences.

Depth maps are of course most often estimated before any compression of the representation of a scene. In this case, the input views are usually uncompressed, so the compression-related distortions do not influence the quality of depth maps. On the other hand, this quality can be decreased due to low accuracy of camera parameters (understood as the accuracy of camera distortion models [Tan17] and intrinsic and extrinsic camera parameters [San17] [Che21]), and the use of floating-point representation [God20] and rounding errors.

Other causes of depth estimation errors are related to inter-view correspondence search can be seen also in noise present in input views of natural sequences [Sta14], as such noise influences the original color of points.

In order to address all listed problems, we propose the novel correspondence search method called point-to-block matching. In the proposal, a point in a view is not compared simply with a collocated point in another view, but to the most similar point in a small block surrounding it.

The introduction of such a degree of freedom in the estimation can be very beneficial on the quality of depth maps, as it decreases the influence of small shifts in images. Moreover, the proposal also can decrease the noise-induced errors, as the proposed method lets us find a less noised similar point in the nearest neighborhood.

To fully test the proposal, the point to block matching was implemented in one of the publicly available state-of-the-art methods of depth estimation [Mie20a] and tested in 10 different configurations to find the most proper applications in which the proposed matching should be used.

The organization of the paper is as follows: Section 2 shows the description of the most relevant similarity measurement methods, with emphasis on their usefulness in the case of use of compressed views in the depth estimation process; Section 3 describes in details the proposed inter-view matching method; Section 4 includes the results of the comparison of the proposed point-to-block matching with the traditional point-to-point matching.

In the end, Section 5 summarizes the paper and includes conclusions drawn from the performed experiments.

2. RELATED WORKS

Some studies were already performed in order to evaluate the efficiency of the existing matching or similarity measurement methods in various cases, e.g., for noised views. Study [Weg09] has shown that a mixture of census or rank transforms with the simple sum of absolute differences (SAD) or sum of squared differences (SSD) can be very efficient for such multi-view sequences. [Zeg20] proves that the use of structural similarity performs very well when radiometric distortions are present in processed input views.

Authors of [Say15] indicate that similarity measurement should be adapted to the local features of input views. It can be used to change the weight of used mixture similarity metrics and lead to better estimation in problematic areas, such as low-texture regions or occluded parts of views. Another method that tries to adapt to processed views is [Cha18], but in this proposal, the size and shape of the matching window become dependent on the characteristics of the part of the image.

Other methods were proposed to improve the efficiency of inter-view matching by introducing temporal information into the similarity metric [Shi17]. The temporal stability of depth maps is crucial for the quality of the final view synthesis. On the other hand, such stability can be ensured on other stages of the depth estimation process, e.g., by post-processing filtering [Köp13].

An interesting method is proposed by [Suo12]. Its authors propose to perform the matching of views in the transform domain. Such an approach results in exposure-independent estimation, very valuable when multi-view systems are used. However, if the encoding of input views would be performed as simulcast compression (independent in each view, e.g., performed in each camera as post-processing step), the distribution of frequencies could significantly vary in compressed views, negatively influencing the efficiency of transform domain matching.

Matching methods base on minimum spanning tree, such as [Zha19], show high robustness for complex scenes, however, no qualitative results of efficiency for noised or compressed images are provided. This method uses also SURF [Bay08] and KAZE [Alc12] features detectors, which are translation invariant, so they could be also invariant to compression-induced errors, however, these detectors are used only for calculation of disparity range, not depth maps themselves.

To the best knowledge of the authors, contrary to the proposal presented in Section 3, none state-of-the-art method of inter-view correspondence search takes specifically into account the compression-induced artifacts present in the encoded input views.

3. PROPOSED INTER-VIEW MATCHING METHOD

The problem of the depth map estimation can be in most cases presented as a cost (goal) function minimization [Kol02]. In its most commonly used form, the cost function is defined as:

$$E(\bar{d}_p) = \sum_{p \in \mathrm{P}} D_p(\bar{d}_p) + \sum_{p \in \mathrm{P}} \sum_{q \in \mathrm{Q}} V_{p,q}(\bar{d}_p, \bar{d}_q), \quad (1)$$

where P is set of points of an input view, p is point of this view, $\bar{d}_p$ is currently considered depth of a point p, D_p is data term that represents a cost of assigning the depth d_p to the point p, Q is set of points in the neighborhood of p, q is point in this neighborhood, $\bar{d}_q$ is currently considered depth of a point q, $V_{p,q}$ is smoothness term that represents the intra-view discontinuity cost of using the depth values d_p and d_q.

The proposed method changes the data term D_p, which is responsible for the correspondence between points in different views. D_p usually uses some similarity metric (e.g., one of metrics described in Section 2) between a point p and a point p', which is a point in neighboring view that corresponds to the point p for the considered depth of a point ($\bar{d}_p$).

In the proposal, the data term compares the point p with points in a small block of points R that surround point p' and chooses the point that gives the lowest matching error (highest similarity) to point p. So, if the SAD metric would be used, then:

$$D_p(\bar{d}_p) = \min\big(SAD(p,r)\big), \; r \in R. \quad (2)$$

Therefore, if 3×3 block would be used as R, the $D_p(\bar{d}_p)$ would be equal to the smallest of 9 SAD values, calculated for the central point p' and its 1- and 2- pixel neighborhood, respectively.

Note that any similarity measure can be used with the proposal. Therefore, such a method can enhance almost any depth estimation method with robustness to small shifts in images caused e.g., by image compression and decrease the noise-induced errors.

The drawback of the proposal can be seen in the small increase of the computational complexity, as the similarity is now performed not once for each point, but R^2 times, nevertheless, the calculation of similarity can be easily parallelized.

As the proposal was implemented in the depth estimation software provided in [Mie20a], the Graph Cut method [Kol02] was used to minimize the function (1).

4. EXPERIMENTAL RESULTS

The performed experiments were performed for different configurations of the modified depth estimation software [Mie20a]:

1. Standard parameters:
   - 256 levels of depth,
   - 50,000 superpixels per view,
   - uncompressed input views,
   - all views used in the estimation.
2. Decreased number of depth levels:
   2.1. 16 levels of depth,
   2.2. 128 levels of depth.
3. Changed number of superpixels [Ach12]:
   3.1. 5000 superpixels per view,
   3.2. 100000 superpixels per view.
4. Compression of input views:
   4.1. Low compression,
   4.2. Medium compression,
   4.3. High compression.
5. Decreased number of used cameras:
   5.1. 70% of cameras,
   5.2. 20% of cameras.

All configurations were tested 3 times: for the unmodified depth estimation and the point-to-point matching performed in 3×3 and 5×5 blocks. The use of larger blocks is possible, but as preliminary tests have shown, does not have a large influence on the estimated depth maps.

The quality of estimated depth maps directly affects the final product, which is the most faithful representation of the three-dimensional scene. Nevertheless, in the case of using depth maps for the view synthesis purposes, such faithfulness is understood more as the ability to provide virtual views of high quality, rather than the low absolute error in comparison with ground-truth depth [Fan16]. Moreover, such ground-truth depth maps are not available for natural multi-view sequences, making such comparison not possible.

Therefore, in order to test the proposal, a set of estimated depth maps was used for the synthesis of

virtual views which were placed in the same position as the real cameras that captured a scene. The synthesized views were in the end compared with the original input views. The objective comparison was performed using IV-PSNR [MPEG20], which is the PSNR-based metric used by the ISO/IEC MPEG Video Coding to evaluate immersive video quality. The calculated IV-PSNR was averaged for all views in each test sequence (listed in Table 1).

The experiments were performed for 5 multiview test sequences that vary in the number of input views and their arrangements.

Sequence name	Number of views	Resolution	Source
Carpark	9	1920×1088	[Mie20b]
Fencing	10	1920×1080	[Dom16]
Frog	13	1920×1080	[Sal18]
Painter	16	2048×1088	[Doy18]
Street	9	1920×1088	[Mie20b]

Table 1. Test sequences used in experiments.

The virtual view synthesis software used in the experiments is VVS 2.0 with its default configuration [Boi19], i.e., the synthesis of each virtual view was performed using 4 nearest views (and corresponding depth maps).

The software used for compression of input views is x265 [X265] which is the implementation of the MPEG HEVC encoder. The values of the quality parameter *crf* were equal to 0 (low compression, near-lossless), 26 (medium compression), and 51 (high compression).

The following five subsections include the description of acquired results for each test sequence. Color green in tables denotes that the quality of the proposal (point-to-block matching) was higher than for unmodified depth estimation. The last subsection contains a summary of performed experiments.

Carpark

Table 2. shows the average IV-PSNR of synthesized views for the Carpark sequence.

For 8 out of 10 the point-to-block matching 3×3 provides a gain in the quality, making this option the best for this sequence. On the other hand, the highest gain (more than 3 dB) can be seen for the number of superpixels decreased to 5000 for 5×5 block.

Fencing

For the Fencing sequence, the results, presented in Table 3, show that the proposed matching provides a gain in the quality mainly for the compressed input views. This sequence is the only one with a non-planar camera arrangement, which can suggest that in such a case, the use of point-to-block matching is not beneficial when uncompressed input views are used.

	Average IV-PSNR [dB] of synthesized view		
Depth estimation configuration	Point-to-point matching	Point-to-block matching (3×3)	Point-to-block matching (5×5)
Standard parameters	37.09	37.19	37.11
16 levels of depth	36.09	36.13	36.13
128 levels of depth	37.07	37.20	37.16
5000 superpixels	32.28	32.39	35.55
100000 superpixels	38.28	38.35	38.21
Low compression	37.10	37.23	37.14
Medium compression	37.36	37.28	37.18
High compression	34.08	34.17	34.20
70% of cameras	36.19	36.21	35.95
20% of cameras	35.21	35.17	35.21

Table 2. Results of experiments for the Carpark sequence.

	Average IV-PSNR [dB] of synthesized view		
Depth estimation configuration	Point-to-point matching	Point-to-block matching (3×3)	Point-to-block matching (5×5)
Standard parameters	40.24	40.13	39.90
16 levels of depth	35.40	35.55	35.72
128 levels of depth	40.56	40.23	40.05
5000 superpixels	37.87	37.77	37.71
100000 superpixels	40.89	40.87	40.55
Low compression	40.49	40.10	39.98
Medium compression	38.19	39.03	39.14
High compression	34.09	36.19	36.18
70% of cameras	33.80	39.65	39.06
20% of cameras	38.41	38.40	38.43

Table 3. Results of experiments for the Fencing sequence.

Frog

In the Frog sequence, the best results were obtained for Point-to-block matching in 3×3 block (Table 4).

One of the possible explanations to results observed for Frog can be a very high amount of noise present in this sequence. Table 5 shows the standard deviation of noise estimated for each sequence (from [Dzi20]). As it can be seen, Frog has more than twice the amount of noise than other sequences. Therefore, in configurations without the compression of input views, the point-to-block matching provides a gain in most cases, as it decreases the influence of noise on the inter-view matching.

On the other hand, compression of video highly reduces the noise of sequences, even for low compression. In this sequence, objects have very detailed, rich textures (see Figure 3), so the changes in the position of edges in compressed views do not

occur so often as e.g., in the Fencing sequence, where the foreground and background objects have similar color characteristics. We can see that a high gain in the quality in Frog cannot be seen until high compression is applied.

	Average IV-PSNR [dB] of synthesized view		
Depth estimation configuration	Point-to-point matching	Point-to-block matching (3×3)	Point-to-block matching (5×5)
Standard parameters	38.95	39.00	38.54
16 levels of depth	32.63	32.97	33.05
128 levels of depth	38.99	39.01	38.66
5000 superpixels	37.27	36.97	36.53
100000 superpixels	39.07	39.16	38.62
Low compression	39.04	38.98	38.47
Medium compression	38.99	38.92	38.33
High compression	33.90	34.26	33.88
70% of cameras	38.05	37.89	37.58
20% of cameras	33.28	33.31	33.08

Table 4. Results of experiments for the Frog sequence.

Sequence	σ_Y	σ_U	σ_V
Carpark	10.29	4.23	4.57
Fencing	12.83	3.38	3.83
Frog	26.77	5.59	5.62
Painter	10.17	3.89	3.89
Street	9.85	3.84	4.21

Table 5. Estimated noise standard deviation for luma (σ_Y) and chromas (σ_U and σ_V) [Dzi20]. Evaluated sequences have 10 bits per sample.

Painter

Painter is the only sequence that shows decreased quality in almost all cases of estimation with the point-to-block matching (Table 6). This sequence is the only sequence with lightfield-like arrangement of cameras (matrix of cameras).

	Average IV-PSNR [dB] of synthesized view		
Depth estimation configuration	Point-to-point matching	Point-to-block matching (3×3)	Point-to-block matching (5×5)
Standard parameters	40.55	40.18	39.57
16 levels of depth	39.04	39.12	39.13
128 levels of depth	40.63	40.36	39.73
5000 superpixels	39.51	39.14	38.70
100000 superpixels	40.64	40.24	39.47
Low compression	40.68	40.17	39.86
Medium compression	39.87	39.66	39.04
High compression	36.32	35.85	35.58
70% of cameras	40.22	39.84	39.66
20% of cameras	40.31	40.12	39.70

Table 6. Results of experiments for the Painter sequence.

Street

Results for the Street sequence (Table 7) clearly benefit from using the proposed point-to-block matching for all tested configurations of depth estimation.

The gain of quality is very high in most cases (up to 3 dB). As it can be seen in Figure 4, most of the differences between synthesis that used depth maps with point-to-point matching and point-to-block matching are still visible mainly on the edges of objects.

	Average IV-PSNR [dB] of synthesized view		
Depth estimation configuration	Point-to-point matching	Point-to-block matching (3×3)	Point-to-block matching (5×5)
Standard parameters	37.34	40.23	40.09
16 levels of depth	36.14	39.09	39.03
128 levels of depth	37.23	40.23	39.98
5000 superpixels	31.99	32.75	32.83
100000 superpixels	38.34	40.43	39.95
Low compression	37.37	40.35	40.06
Medium compression	40.24	40.08	40.53
High compression	32.52	34.59	34.49
70% of cameras	35.34	37.99	38.04
20% of cameras	38.97	38.32	40.10

Table 7. Results of experiments for the Street sequence.

Summary of experiments

In order to summarize the performed experiments, the results were averaged for all sequences and presented in Table 8.

	Average IV-PSNR [dB] of synthesized view		
Depth estimation configuration	Point-to-point matching	Point-to-block matching (3×3)	Point-to-block matching (5×5)
Standard parameters	38.83	39.35	39.04
16 levels of depth	35.86	36.57	36.61
128 levels of depth	38.90	39.41	39.12
5000 superpixels	35.78	35.80	36.26
100000 superpixels	39.44	39.81	39.36
Low compression	38.94	39.37	39.10
Medium compression	38.93	38.99	38.84
High compression	34.18	35.01	34.87
70% of cameras	36.72	38.32	38.06
20% of cameras	37.24	37.06	37.30

Table 8. Results of experiments averaged for all sequences.

Although the results vary for different sequences, as was presented in previous subsections, these results show that for the used set of test sequences the point-to-block matching in 3×3 block was on average better than standard point-to-point matching in all but one

Figure 3. Comparison of virtual views synthesized using depth maps estimated using point-to-point matching (left column), point-to-block matching with 3×3 block (center), and with 5×5 block (right). Test sequences (top to bottom): Painter, Frog, Fencing, Carpark, Street.

tested configurations. Most significant gains can be observed for the reduced number of cameras, high compression, and levels of depth reduced from 256 to 16.

5. CONCLUSIONS

This paper presents the novel method of the inter-view correspondence search called point-to-block matching. In the proposal, a point in a view is not compared simply with a point in another view, but with the most similar point in a small block surrounding it.

Series of performed experiments based on the comparison of synthesized virtual views with the input views showed that on average the proposal increases the quality of estimated depth maps in almost all configurations. The results for individual sequences seem to confirm that the method provides the best results for highly compressed input views or when the

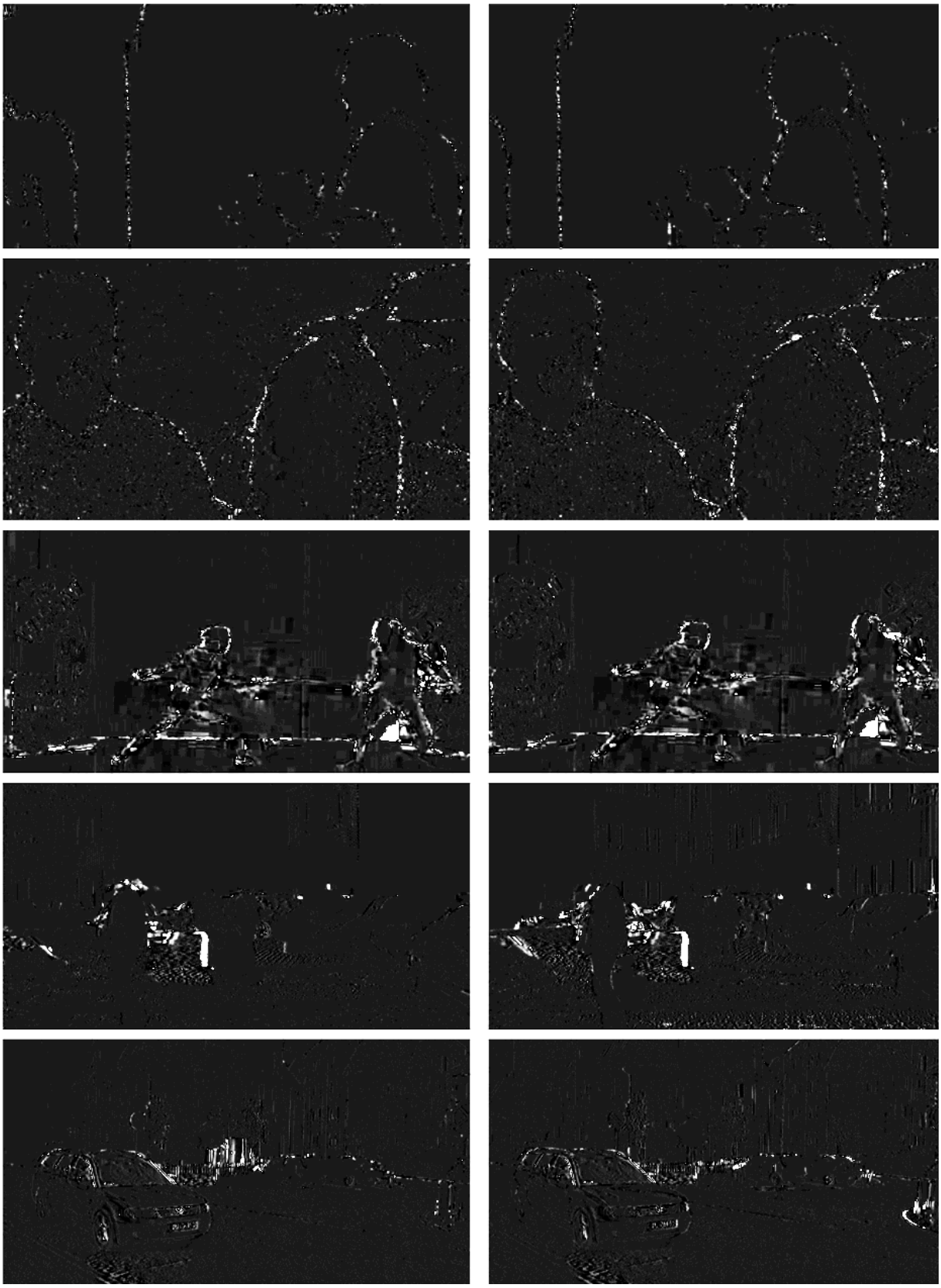


Figure 4. Logarithm of differences between virtual views synthesized using depth maps estimated using point-to-point matching and point-to-block matching with 3×3 block (left) / 5×5 block (right). Test sequences (top to bottom): Painter, Frog, Fencing, Carpark, Street. The contrast of the image was increased to increase the visibility of differences.

amount of noise present in a multi-view sequence is significant.

A gain in quality was also observed for all sequences in cases of decreased number of depth levels tested during the estimation process. Such estimation is much less computationally expensive, which indicates that the method could also be very effective in visual systems with limited resources (e.g., a typical PC with two internet cameras). Such systems almost always operate using compressed input video streams, therefore, the abovementioned efficiency of the proposal in such cases further increases its usability in such real-life configuration.

The need for further research on depth estimation methods adapted to be used with compressed input views can be seen also in the current state of the development of the new MPEG Immersive Video codec [Boy21] for virtual and augmented reality applications. This soon-to-be video encoding standard includes Geometry Absent profile [MPEG21] which can be used to send a subset of input views to the decoder, where the depth estimation process is performed to synthesize any viewport of the three-dimensional scene.

6. ACKNOWLEDGMENTS

This work was supported by the Ministry of Education and Science of Republic of Poland.

7. REFERENCES

[Ach12] Achanta R., Shaji A., Smith K., Lucchi A., Fua P., and Süsstrunk S. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. 2012 IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, pp. 2274-2282, 2012.

[Alc12] Alcantarilla P. F., Bartoli A., and Davison A. J. KAZE features. Proceedings of European Conference on Computer Vision, pp. 214–227, 2012.

[Bay08] Bay H., Ess A., Tuytelaars T., and Van Gool T. Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding, vol. 110, Issue 3, pp. 346-359, 2008.

[Boi19] Boissonade P., and Jung J. [MPEG-I Visual] Improvement of VVS1.0.1. ISO/IEC JTC1/SC29/WG11/MPEG2019/m46263, Marrakesh, Jan. 2019.

[Boy21] Boyce J., Doré R., Dziembowski A., Fleureau J., Jung J., Kroon B., Salahieh B., Vadakital V., and Yu L. MPEG Immersive Video Coding Standard. Proceedings of the IEEE, early access. 2021.

[Cha18] Chai Y. and Cao X. Stereo Matching Algorithm Based on Joint Matching Cost and Adaptive Window. 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, pp. 442-446, 2018.

[Che21] Chen R., Chen F., Xu G., et al. Precision analysis model and experimentation of vision reconstruction with two cameras and 3D orientation reference. Scientific Reports, No. 11, 3875, 2021.

[Dom16] Domański M., Dziembowski A., Grzelka A., Mieloch D., Stankiewicz O., and Wegner K. Multiview test video sequences for free navigation exploration obtained using pairs of cameras. Document ISO/IEC JTC1/SC29/WG11, MPEG M38247, 2016.

[Dom17] Domański M., Stankiewicz O., Wegner K. and Grajek T. Immersive visual media – MPEG-I: 360 video, virtual navigation and beyond. 2017 IEEE International Conference on Systems, Signals and Image Processing (IWSSIP), Poznań, pp. 1–9, 2017.

[Doy18] Doyen D., Boisson G., and Gendrot R. [MPEG-I Visual] New Version of the Pseudo-Rectified Technicolorpainter Content. Document ISO/IEC JTC1/SC29/WG11 MPEG/M43366, Ljublana, 2018.

[Dzi16] Dziembowski A., Domański M., Grzelka A., Mieloch D., Stankowski J., and Wegner K. The influence of a lossy compression on the quality of estimated depth maps. 2016 International Conference on Systems, Signals and Image Processing (IWSSIP), Bratislava, pp. 1-4, 2016.

[Dzi20] Dziembowski A., Mieloch D., Stankiewicz O., Nikonowicz J., Domański M. Noise modeling in TMIV. Document ISO/IEC JTC1/SC29/WG11 MPEG/M54895, Online, June 2020.

[Fan16] Fang L., Xiang Y., Cheung N. M., and Wu F. Estimation of virtual view synthesis distortion toward virtual view position. 2016 IEEE Transactions on Image Processing, vol. 25, no. 5, pp. 1961–1976, May 2016.

[Gar19] Garus P., Jung J., Maugey T., and Guillemot C. Bypassing Depth Maps Transmission for Immersive Video Coding. 2019 Picture Coding Symposium (PCS), Ningbo, pp. 1-5, 2019.

[God20] Godbolt M. Optimizations in C++ compilers. Communications of the ACM, vol. 63, no. 2 pp. 41-49, 2020.

[Kol02] Kolmogorov V., and Zabih R. Multi-camera Scene Reconstruction via Graph Cuts. Proceedings of the 7th European Conference on Computer

Vision-Part III (ECCV '02), London, pp. 82-96, 2002.

[Köp13] Köppel M., Makhlouf M. B., Müller M., and Ndjiki-Nya P. Temporally consistent adaptive depth map preprocessing for view synthesis. 2013 Visual Communications and Image Processing (VCIP), Kuching, Malaysia, pp. 1-6, 2013.

[Laf16] Lafruit G., Domański M., Wegner K., Grajek T., Senoh T., Jung J., Kovács P., Goorts P., Jorissen L., Munteanu A., Ceulemans B., Carballeira P., García S., and Tanimoto M. New visual coding exploration in MPEG: Super-MultiView and Free Navigation in Free viewpoint TV. 2016 Proceedings of the Electronic Imaging Conference: Stereoscopic Displays and Applications, San Francisco, pp. 1–9, 2016.

[Leo07] Leontaris A., Cosman P. C., and Reibman A. R. Quality Evaluation of Motion-Compensated Edge Artifacts in Compressed Video. 2007 IEEE Transactions on Image Processing, vol. 16, no. 4, pp. 943-956, Apr. 2007.

[Mie20a] Mieloch D., Stankiewicz O., and Domański M. Depth Map Estimation for Free-Viewpoint Television and Virtual Navigation. 2020 IEEE Access, vol. 8, pp. 5760-5776, 2020.

[Mie20b] Mieloch D., Dziembowski A., and Domański M. [MPEG-I Visual] Natural Outdoor Test Sequences. Document ISO/IEC JTC1/SC29/WG11 MPEG/M51598, Brussels, Jan. 2020.

[MPEG20] Software manual of IV-PSNR for Immersive Video. Document ISO/IEC JTC1/SC29/WG04 MPEG VC, N0013, Online, Oct. 2020.

[MPEG21] Test Model 8 for MPEG Immersive Video. Document ISO/IEC JTC1/SC29/WG04 MPEG VC, N0050, Online, Jan. 2021.

[Mül11] Müller K., Merkle P., and Wiegand T. 3-D Video Representation Using Depth Maps. 2011 Proceedings of the IEEE, vol. 99, no. 4, pp. 643-656, 2011.

[Sal18] Salahieh B., Marvar B., Nentedem M., Kumar A., Popvic V., Seshadrinathan K., Nestares O. and Boyce J. Kermit test sequence for Windowed 6DoF Activities. Document ISO/IEC JTC1/SC29/WG11 MPEG/M43748, Ljublana, Slovenia, Jul. 2018.

[San17] Sankowski W., Włodarczyk M., Kacperski D., and Grabowski K. Estimation of measurement uncertainty in stereo vision system, Image and Vision Computing, vol 61, pp. 70-81, 2017.

[Say15] Saygili G., Van der Maaten L., Hendriks E. A. Adaptive stereo similarity fusion using confidence measures. Computer Vision and Image Understanding, vol. 135, pp. 95-108, 2015.

[Shi17] Shin Y., and Yoon K. Adaptive spatiotemporal similarity measure for a consistent depth maps. 4th International Conference on Computer Applications and Information Processing Technology (CAIPT), Kuta, pp. 1-4, 2017.

[Sta14] Stankiewicz O., Domański M., and Wegner K. Analysis of noise in multi-camera systems. 3DTV Conference 2014, Budapest, 2014.

[Suo12] Suominen O., Gotchev A., and Hannuksela M. Transform domain similarity measures in stereo matching. 2012 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), Zurich, pp. 1-4, 2012.

[Tan12] Tanimoto M., Tehrani M. P., Fujii T., and Yendo T. FTV for 3-D Spatial Communication. 2012 Proceedings of the IEEE, vol. 100, no. 4, pp. 905-917, 2012.

[Tan17] Tang Z., Grompone von Gioi R., Monasse P., and Morel J. A Precision Analysis of Camera Distortion Models. IEEE Transactions on Image Processing, vol. 26, no. 6, pp. 2694-2704, 2017.

[Tip13] Tippetts B., Jye Lee D., Lillywhite K., and Archibald J. Review of stereo vision algorithms and their suitability for resource-limited systems. Journal of Real-Time Image Processing, vol. 11, no. 1, pp. 5-25, 2013.

[Weg09] Wegner K., and Stankiewicz O. Similarity measures for depth estimation. 2009 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, Potsdam, pp. 1-4, 2009.

[X265] MulticoreWare Inc. x265 HEVC Encoder/ H.265 Video Codec. Available on http://x265.org

[Zeg20] Zeglazi O, Rziza M, Amine A, Demonceaux C. Structural Similarity Measurement Based Cost Function for Stereo Matching of Automotive Applications. Journal of Imaging, vol. 6, no. 8, 2020.

[Zha19] Zhang C., He C., Chen Z., Liu W., Li M. and Wu J. Edge-Preserving Stereo Matching Using Minimum Spanning Tree. IEEE Access, vol. 7, pp. 177909-177921, 2019.

3D Scanning of Semitransparent Amber with and without Inclusions

Adam L. Kaczmarek, Jacek Lebiedź
Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, ul. G. Narutowicza 11/12, 80-233 Gdansk, Poland
adakaczm@pg.edu.pl, jacekl@eti.pg.edu.pl

Jakub Jaroszewicz, Wojciech Święszkowski
Warsaw University of Technology
Faculty of Materials Science and Engineering
ul. Wołoska 141, 02-507 Warsaw, Poland
jakub.jaroszewicz@pw.edu.pl
wojciech.swieszkowski@pw.edu.pl

ABSTRACT

This paper is concerned with making 3D scans of semitransparent ambers with and without inclusions. The paper presents results of using a variety of devices applied for this purpose. Equipment used in the experiments includes a 3D laser scanner, a structured light scanner, a stereo camera, a camera array and a tomograph. The main object used in the experiment was an amber with a fossil of a lizard. The paper shows possibilities of acquiring the 3D structure of fossils embedded in semitransparent material which interfere with the measurement performed by 3D scanning equipment. Moreover, the paper shows the application of results of 3D scanning as the 3D scan of a lizard was reconstructed in a virtual reality cave making it possible to visualize in detail its shape and texture.

Keywords

3D scanning; scanning of amber; scanning of semitransparent material; VR caves

1 INTRODUCTION

The paper is dedicated to the problem of acquiring 3D scans of ambers. Ambers are fossils which can be semitransparent or not transparent. They can also contain inclusions such as parts of plants or insects. These fossils may have heterogeneous structure and a single amber stone may contain parts differing in their density and hue. Features of amber cause that it is particularly problematic to obtain 3D scans of this kind of specimens. One of the purposes of obtaining a 3D scan of an amber is the possibility to precisely present it to a potential purchaser. It includes both unprocessed amber stones, jewelry and other ornaments. Apart from that 3D scans of the most valuable ambers can be used to create virtual museums making it possible for virtual visitors to watch exhibits.

This paper presents the results of acquiring 3D scans with the use of different kind of scanning equipment. Authors of this paper performed experiments with scanning amber with the use of Light Detection and Ranging (LIDAR), a structured-light 3D scanner, a stereo camera, a camera array and a tomograph. Ambers scanned in the experiments are exhibits displayed in municipal Amber Museum located in Gdansk, Poland. One of the most precious exhibits provided by the museum is amber with fossilized lizard. It is one of only few such objects in the world. Moreover, as a part of the research presented in this paper the results of scanning was used to reconstruct the fossilized lizard in virtual reality cave making it possible to precisely visualize its shape.

Original contributions of this paper includes application of a camera array for obtaining 3D scans of semitransparrent amber with inclusions, results of performing a scan using a tomograph on a fossil of an ancient lizard embeded in a amber and visualizing the result in a virtual reality cave in cooperation with a muzeum of amber. The structure of the paper is the following. Section 2 is a literature overview. Section 3 and 4 describe experiments and Section 5 presents their results. Section 6 shows the application of 3D scans to virtual reality cave. The last section presents the conclusion of the paper.

2 RELATED WORK

Ihrke et al. presented a detailed survey on 3D scanning of semi-transparent objects [Ihr10a]. They identified nine classes of transparency. Eight of these classes refer to levels of light transport in objects including opaque, translucent and fully transparent objects. The ninth class are inhomogeneous objects consisting of parts with various transparency. Amber can be foremost classified as an inhomogeneous object. Light passing through amber is affected by many factors including

sub-surface scattering, reflections and absorption. Ihrke et al. described also suggestions for selecting the most suitable 3D scanning methods depending on the transparency of scanned objects. They considered performing 3D scans of objects such as glass, gas flows and fire.

Scanning transparent objects is also the subject of research of Wu et al. [Wu18a]. They developed a method for automatically and directly reconstructing complete 3D models for transparent objects based only on their appearances in a controlled environment. The environment is designed using affordable and off-the-shelf products, which include a LCD monitor, two turntables, and two cameras. This setup can work in fully automatic fashion, removing the needs for manually adjusting object positions and calibrating the cameras. The weakness of this approach is the assumption that the transparent object is homogeneous, which is rarely true in the case of amber. Interesting results were also brought by the research of the same team on the simultaneously recovering the 3D shape of both the wavy water surface and the moving underwater scene [Qia18a]. This approach exploits multiple viewpoints by constructing a portable camera array. However, the assumption is that the water is homogeneously transparent.

Research was also performed regarding 3D scanning of amber with inclusion. In particular, the usage of confocal laser scanning microscopy was analyzed by Clark et al. (scanning of trichomes) [Cla10a] and Zaharenko et al. (scanning of a spider) [Zak19a]. Selden took advantage of cameras, microscopes and X-ray computed tomography for imaging fossil spiders [Sel17a]. Kamp et al. showed possibilities of using X-ray microtomography with amber [Kam14a]. Li et al. analyzed amber in the context of its characteristics as a martial [Li15a].

To the best of our knowledge there were no research regarding usage of camera arrays for scanning amber with inclusions. Moreover, research papers concerned with scanning amber focus on analyzing its structure and content in the context of geological or biological research. These studies are not performed with the purpose of reconstructing amber inclusions in virtual reality cave which we present in this paper.

Equipment for 3D scanning

Laser and structured-light scanners

Laser 3D scanners produce separate clouds of points for every position of scanning. Each of these points is described by its three coordinates. They are calculated by timing the round-trip time of a pulse of light (time-of-flight laser range finder) or by analysis of the locations of the laser dots on the 2D image recorded by a camera embedded in a scanner camera related to the position of the camera and the laser emitter (triangulation). Acquiring a full specimen requires scanning it from all sides with some angular resolution. The created local clouds of points should then be combined into one global cloud by matching corresponding to each other distinctive points (so called reference points) identified in scanned shape for each local cloud [Kyo13a]. Then, the global cloud can be displayed directly [Lev85a, Wim06a, Dis18a] or to avoid inconsistencies it should be converted into polygon (e.g. triangle) mesh by special software.

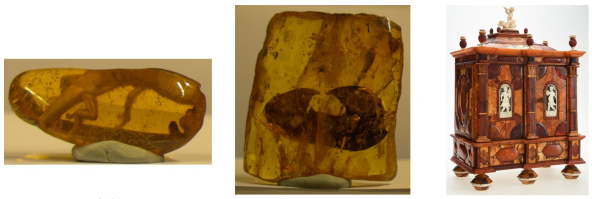
Pure laser scanners produce only geometrical shape of object represented by a point cloud or a polygon mesh without information about color. In order to enrich the scan by color an extra camera is usually needed for assigning color for every point of the cloud or for capturing a texture applied to a larger area of the surface.

Analogically to laser scanners, structured-light 3D scanners produce also separate local clouds of points for each view of a specimen, where every point is described by three coordinates. Positions of cloud's points are calculated on the base of assessment of deformation of the multiresolutional patterns projected on the specimen and taken by an embedded camera located on the side of the projector. If the structured light is white the same camera can be used for register a color of scanned object. Otherwise, an additional camera that records the color in white light is needed. The global cloud and polygon mash are created in the same way as for laser scanners.

Microcomputed tomograph

Microcomputed tomography (μCT) is a nondestructive technique that generates cross-sectional images of a sample using an X-ray source. Using these 2D representations, it is possible to generate a 3D reconstruction of the sample and perform analysis on the material structure. Amber with fossil inclusions is well suited for μCT, both in terms of typical size and composition. The method is non-destructive and requires minimal preparation, generating 3-dimensional reconstructions that can be sectioned and viewed from numerous angles, essentially permitting digital 'dissection' of the specimen within the amber. In recent years various forms of μCT have proved particularly well-matched for imaging inclusions in amber [Die07a, Keh14a, Mor16a] and have been successfully applied in fossil lizards [Pol02a, Daz16a]. A μCT scanner looked inside the amber without damaging the fossils, allowing study researchers to digitally piece together tiny bones and examine soft tissue. The 3D images of the detailed preservation provided insight into the anatomy and ecology of ancient lizards.

Unfortunately, the tomograph is unable to register the color of the specimen. Therefore, if the geometric form itself is not satisfactory, then the reconstructed solid should be colored separately. We stopped at the non-colored reconstruction 3D.



(a) (b) (c)

Figure 1: Amber exhibits used in experiments (a) Gierłowska's lizard (b) Amber with part of a leaf (c) Zernebach's Cabinet (photo by Michał Kosma Szczerek, courtesy of the Amber Museum, a branch of the Museum of Gdansk

Stereo cameras

A pair of cameras resembling a pair of eyes makes it possible to record 3D visual information of the surrounding area. Such a pair of cameras aimed in the same direction is called a stereo camera. A stereo camera produces two images taken from different points of view. Locations of the same objects is different in these images unless these objects are too distant from cameras. The difference in locations of objects in two different images is called a disparity. The closer the object is to a stereo camera, the greater is a disparity. Disparities are retrieved from a pair of images as a result of processing these images by a stereo matching algorithm which are designed to identify areas of images corresponding to the same objects.

A stereo camera consists of a reference camera and a side camera. A reference camera is a point of view of the camera set. Disparities for a series of points visible in a reference image are saved in the form of a disparity map. Disparity maps can be converted to depth maps which contain values of distances between a reference camera and objects visible in images. Converting disparity map to a depth map requires taking into account parameters of a stereo camera such as focal length of lens and the distance between constituent cameras called the baseline.

Disparity maps and depth maps obtained with the use of stereo cameras may contain errors which are incorrect values occurring in these maps. This issue is related to the selection of an appropriate stereo matching algorithm. There is a large variety of such algorithms. They differ in the quality of results, speed, the amount of required computer resources and other parameters. There are also rankings of stereo matching algorithms considering these parameters. Leading rankings of these algorithms are provided by Middlebury Stereo Vision Page [Mid21a] and the KITTI Vision Benchmark Suite [Kit21a]. The former ranks more than 100 stereo matching algorithms and the latter takes into account over 140 algorithms. Experiments presented in this paper considered well-known algorithms with available implementations. Disparity maps can also be obtained using camera arrays. [Kac17b, Kac19c] include overview of this technology.

3 DATA ACQUISITION

Ambers used in experiments

Amber Museum located in Gdansk, Poland provided amber fossils and items decorated with amber which were used in the experiments. Objects subjected to experiments were Baltic ambers aged about 40 million years. Some of used fossils were ambers containing inclusions. We have experimented with acquiring 3D scans of both natural amber forms with inclusions (Gierłowska's lizard - amber with lizard fossil and amber with part of a leaf) and amber works of art made by former master craftsmen (Zernebach's Cabinet).

The Gierłowska's lizard is a semitransparent Baltic amber (succinite) with inclusion of a lizard *Succinilacerta succinea* of the family *Lacertidae*. It was found by Gabriela Gierłowska (it is named after its finder) in material obtained through rinsing from Holocene fossil beach sediments about 1 kilometer away from the shore in Gdansk's Stogi district in June 1997. Unfortunately, the animal of preserved length of 3.7 centimeters is incomplete. The front part of the head, dorsal fragment of the trunk and the rearmost fragment of the tail are missing [Kos97b]. The fossil is presented in Fig. 1(a).

An amber with part of a leaf is another item used in the experiments. Similarly to the Gierłowska's lizard this object is a semitransparent Baltic amber (succinate), but it is definitely bigger (3.5×4 cm). The specimen is shown in Fig. 1(b).

The cabinet also used in the experiments was made by a master of the Gdansk's amber guild Johann Georg Zernebach in 1724. This small box-shaped baroque piece of furniture has a shape of a cuboid with dimensions of $42 \times 31 \times 19$ cm. Its doors and drawers are made of amber, ivory, silver, mirror and wood [Kos08a]. The cabinet is presented in Fig. 1(c).

Laser and structured light scanners

Laser and structured light 3D scanners are widely used in museology. Many museum objects were archived using these devices. Although the difficulties that arise when scanning transparent and shiny objects are known, we decided to check these technologies also for amber. That is why we asked two companies dealing professionally with scanning to scan the pieces of amber presented above. One of them used a Surphaser laser scanner and a structured blue LED light ZEISS COMET L3D scanner, another one - a structured white light SmartTech Scan3D Surface scanner. The results of these tests were already described in [Leb17c], but it is worth presenting them briefly here.

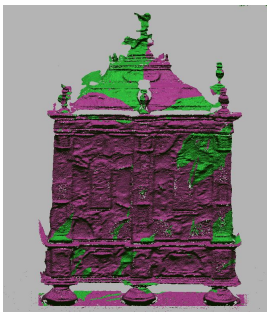


Figure 2: The laser scan of the Zernebach's Cabinet (made by Łukasz Piasecki using Surphaser scanner, courtesy of EC Test Systems) [Leb17c]

All laser 3D scanners use only a specific frequency of light. It means that process of scanning can catch only shapes visible to the used monochromatic laser light rays. The ray is reflected either from the amber exterior or from some internal amber surfaces (inclusions, fissures, structure changes etc.). If the ray enters the interior, its trajectory refracts on the external surface. For this reason, the ray measurement is disturbed. Furthermore, for complex specimens made of different kinds of amber (e.g. Zernebach's Cabinet) scanning may produce different results: a front surface for opaque fragments or a back surface (alternatively internal elements) for transparent parts [Leb17c]. This is clearly seen in Fig. 2, which presents result of use of the Surphaser laser scanner on the Zernebach's Cabinet.

Structured-light 3D scanners use rather polychromatic light (Fig. 3). Rays of a certain range of wavelength can reflect from external surfaces, while other rays can reflect from internal surfaces (inclusions, fissures, structure changes etc.). Such a diverse spectrum allow to acquire different, separate layers of amber. The heterogeneous structure of amber that has multilayer form can be reconstructed by a few correspondingly matched light rays reaching different layers of an amber and its inclusions. Unfortunately, today's scanners do not differentiate light rays relative to wavelength and this approach requires the construction of a new scanner and goes beyond the scope of described research. Existing

Figure 3: Process of structural light scanning of the Zernebach's Cabinet

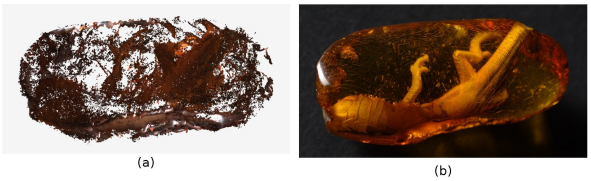


Figure 4: (a) The structured white light scan of the Gierłowska's lizard (made by Szymon Bloch using SmartTech Scan3D Surface scanner, courtesy of Scan 3D - authorized distributor of SmartTech) (b) The image corresponding to a scan (photo by Michał Kosma Szczerek, courtesy of the Amber Museum, a branch of the Historical Museum of Gdansk) [Leb17c]

Figure 5: A structured white light scan of a central part of the Zernebach's Cabinet (made by Szymon Bloch using SmartTech Scan3D Surface scanner, courtesy of Scan 3D - authorized distributor of SmartTech) [Leb17c]

scanners can register point clouds consisting of a couple of point set layers, but they are sparse and may have a lot of holes [Leb17c], as one can see in Fig. 4 (a), which presents result of use of the SmartTech Scan3D Surface scanner based on structured white light on the Gierłowska's lizard. Fig. 4 (b) presents an image corresponding to a scan.

Analyzing results of multilayer amber scanning it is worth remembering that the reflected rays can also refract passing between different parts of an amber and its inclusions. This means that the algorithm locating individual points of the amber interior should also calculate the phenomenon of refraction. Otherwise, obtained results can be inaccurate. They will consist of a few point clouds corresponding to sequent surfaces, but reconstructed shapes inside the amber can be deformed by refraction.

The glossy ambers are an additional problem in accurate scanning both for laser and structured light scanners. These scanners cannot register surfaces where specular reflection is and point clouds have holes in these places. Therefore only matt and opaque ambers can be scanned without much difficulty, as shown in Fig. 5 presenting a structured white light scan of a central part of the Zernebach's Cabinet made by the use of the SmartTech Scan3D Surface scanner [Leb17c].

Usage of tomograph

More precise way for the 3D acquisition of amber or its inclusions is provided by X-ray tomography [Dun11a,

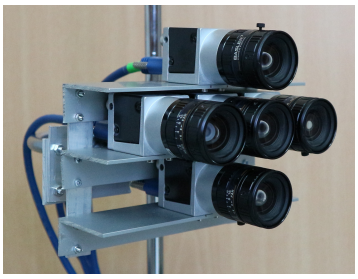


Figure 6: Real EBCA used in the experiments

Ste16a]. The basic disadvantage of this method is its huge cost but it has also some other limitations. For instance, metal parts in acquired objects make difficult to acquire amber jewelry. Fortunately, it is not a problem for natural amber forms. The size of acquired specimens is the next limitation. The dimensions of the tomographic equipment chamber limit the size of the examined object. Furthermore, the results produced by tomography are represented as memory consuming voxels instead of point clouds or easy to visualize polygon meshes generated by the other acquisition techniques.

The X-ray micro-CT observations were conducted at Warsaw University of Technology, Poland using a Zeiss Xradia XCT-400 system. Scans were performed with a polychromatic X-ray beam at an energy of 40 kV and power of 10 W. Sample to detector distance was set to 60 mm, and a source to sample distance of 80 mm was used. Tomographic slices were generated from 1200 rotational steps through 216-degrees of rotation, using a LFOV objective, and the exposure time during each projection was set to 4 s. Acquired images were binned ($2 \times 2 \times 2$) giving voxel size of 38 μm.

Usage of cameras

Equal Baseline Camera Array (EBCA) is a device similar to a stereo camera however it produces a higher quality of disparity maps [Kac17b, Kac19c, Kac15a]. EBCA consists of a central camera and four side cameras equidistant from the central one. Side cameras are placed around a central camera forming a configuration resembling a plus sign. The image of a real EBCA used in the experiments is presented in Fig. 6. The set consists of Basler acA2500-14uc cameras with resolution 2590×1942 px and 1/2.5" sensor size.

EBCA has functions of four stereo cameras combined with each other. The considered subsets are pairs of cameras which consists of a central camera and one of side cameras. The central camera is a reference one in every such a pair. The same reference camera in every pair cause that using EBCA can be regarded as performing four measurements of the same distance with the use of four stereo cameras containing different side cameras. Perceiving EBCA in this way has many advantages and makes it possible to obtain results that have over 26% less errors than the results acquired from a single stereo camera [Kac17b].

Images from EBCA need to be analyzed in order to retrieve 3D data similarly as in case of a stereo camera. This paper presents results of using Exceptions Excluding Merging Method (EEMM) in this process [Kac17b]. This method proved to be the best one for obtaining disparity maps with the use of Equal Baseline Camera Array (EBCA). When EEMM is used disparity maps are obtained from four stereo cameras from EBCA independently from each other. Afterwards, these maps are merged together in order to acquire a resulting disparity map that has a higher quality. EEMM defines functions for processing data from four disparity maps in order to minimize the error rate in the resulting map. The method is further described in Sect. 4 of this paper.

In the experiments two stereo matching algorithms were used i.e. Semi-Global Block Matching (StereoSGBM) and Graph Cut with Expansion Moves (GC Expansion). StereoSGBM is provided with the OpenCV library [Bra08a, Hir08a]. OpenCV is a widely used open-source programming library which includes a variety of algorithms for computer vision. It also contains implementations of four stereo matching algorithms. StereoSGBM was chosen because it is the only algorithm from OpenCV classified in Middlebury ranking and it has the highest score among OpenCV algorithms in the KITTI ranking. Another stereo matching algorithm used in the experiments is Graph Cut with Expansion Moves (GC Expansion) provided by Middlebury Stereo Vision Page [Boy01a]. It is an iterative algorithm which minimizes the energy function using Markov Random Field [Sch02a]. This algorithm was selected for tests because of previous research on the five camera array used in the experiments presented in this paper [Kac17b, Kac19c]. The research showed that GC Expansion produces high quality of results when it is used with the considered array.

Images obtained with the camera set presented in Fig. 6 were used both for testing the usage of EBCA and for testing the stereo vision technology based on two cameras. Results for a stereo camera were obtained by processing an image from a central camera and an image from the left camera selected from an entire set of five images provided by EBCA. Sample images of ambers obtained by EBCA for the experiments are presented in Fig. 7.

4 DATA PROCESSING

Processing laser and structured-light scan

The process of combining local point clouds obtaining as a result of "one-side" static scanning into a global cloud is very complicated and so far it cannot be fully

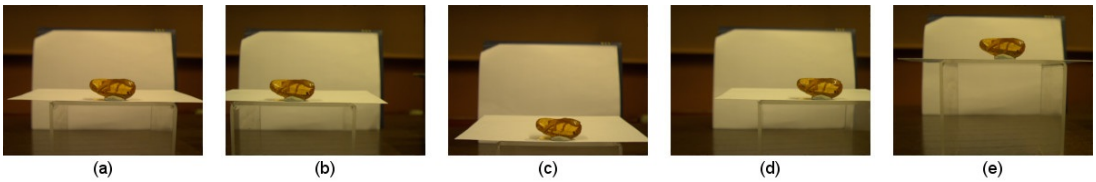


Figure 7: Images of amber with inclusion of a lizard taken with the use of (a) central, (b) right, (c) top, (d) left and (e) bottom cameras of EBCA

automated. Usually it takes the specialist more time than the process of registering points with scanner. Creating the global cloud relies on careful merging of individual local clouds. For this purpose, the corresponding reference points in combined local clouds are distinguished. Then they form the basis of combining [Kyo13a]. Setting reference points is not a trivial task. If the scanned object has many different landmark points (i.e. points in a shape that are located according to some mathematical or geometrical property) and consists of unique elements, then the choice of reference points as these landmarks is obvious. However, if the object is uniform and consists of large single-color areas with a small curvature, it is difficult to indicate the same points as reference points in aggregated scans taken from different sides. In this case, one can stick on the object special markers that will act as reference points. Unfortunately, museum curators refuse to stick such markers on their exhibits.

A separate problem is the conversion of point clouds into a polygon mesh, which is often more convenient for further processing than cloud. Fig. 8 shows point cloud of the upper part of the cabin obtained using structured blue LED light ZEISS COMET L3D scanner (a) and the triangle mesh generated by the ZEISS colin3D software (b). The number of points in such a cloud is usually very large, therefore such conversion needs simplification. The easiest method for it is decimation, consisting in leaving only every n-th point and omitting all other points. More sophisticated methods rely on leaving only points distinctive for a scanned shape (e.g. peaks, valleys, saddle points) [Leb17c].

In the case of transparent objects, the problem is also the separation of external surface and internal layers. Conversion of point cloud into a polygon mesh should be preceded by the separation of points into individual layers. This guarantees conversion to correct surfaces, which will be useful for the rendering mentioned as planned in Sect. 6. Automating this process is not always effective.

Processing tomograph data

Images were imported into Avizo Fire (ver. 2020.2) software platform for segmentation and 3D visualization. The powerful Watershed algorithm with interactive techniques [Rus03a] were used to provide a highly

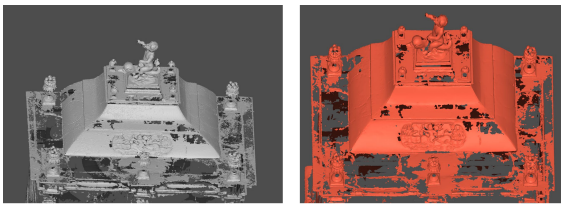


Figure 8: Visualization of the point cloud (a) and the triangle mesh (b) of the Zernebach's Cabinet (made by Łukasz Piasecki using structured blue LED light ZEISS COMET L3D scanner and the ZEISS colin3D software, courtesy of EC Test Systems)

effective strategy for segmenting complex 3D structures of the fossil lizard. The process was based on a marker-controlled algorithm which segments a gray level image based on extracted landscape image and set of markers which identify a subset of the regions of interest inside the data set. The resulting volume was manually programmed to separate the specimen from other objects embedded within the amber (mainly air bubbles). Direct Volume Rendering was used for visualizing 3D scalar fields.

Processing images

Amber fossils are visible only in parts of considered images. Therefore, only these parts were processed in order to acquire disparity maps. Remaining areas of images were not considered in calculations. The part of the image with the Gierłowska's lizard used in calculation has a size of 1000×800 pts and the size of the image of amber with a leaf was set to 1040×1030 pts.

Two stereo matching algorithms were used in experiments for obtaining disparity maps of amber. These are StereoSGBM and GC Expansion presented in Sect. 2. The obligatory input parameter of these algorithms is a range of disparities for which the algorithm verifies a match between a reference image and a side image. The selected range was wider than the real range of disparities of objects visible in images for which disparities are calculated. All other input parameters of stereo matching algorithms were set to default values or values used in examples provided with the software.

This paper presents the comparison of disparity maps obtained using a stereo camera with disparity maps obtained using EBCA. The results for a stereo camera were based on images from the central and the right camera included in the real EBCA used in experiments. StereoSGBM and GC Expansion algorithms were used both in tests with a stereo camera and tests with five camera set.

In case of using EBCA a method of taking advantage of this five camera set needs to be applied in order to acquire disparity maps that have a higher quality than disparity map based on a single stereo camera. In the experiments presented in this paper Exceptions Excluding Merging Method (EEMM) was used for retrieving 3D data on the basis of five image sets [Kac17b]. The usage of EEMM consists of two steps. In the first step, stereo cameras forming EBCA are used independently from each other to obtain disparity maps. Only those stereo cameras are considered for which a central camera is a reference one. Different stereo matching algorithms designed for stereo cameras can be used in this step. Disparity maps acquired from stereo cameras are input data to the second step of Exceptions Excluding Merging Method. In the second step maps are merged in order to obtain a resulting disparity map that has a higher quality than input maps.

Each point of a disparity map contains either a value of a disparity or a value indicating that a disparity for this point is unknown. The reason for the lack of disparities is such that a stereo matching algorithm was not able to determine the disparity on the basis of images from a stereo camera that the algorithm processed. It is mainly caused by limited visibility of the same objects in two different images from a stereo camera. When EBCA consisting of five cameras is used then four disparity maps are merged into a single resulting disparity map. Therefore, every disparity in a resulting map is based on input disparities whose quantity range from 0 to 4. Exceptions Excluding Merging Method specially defines functions for merging disparities with regard to the number of input values. Details of EEMM are described in [Kac17b].

5 RESULTS OF EXPERIMENTS

Laser and structured light 3D scanners

Fig. 2, 5, 8 present results of scanning of the Zernebach's Cabinet by different laser and structured light scanners. Fragments of the cabinet made with opaque and matt amber were scanned with satisfactory accuracy. However, the transparent ambers have already been "lost" and the scan only showed their back wall (e.g. door corners in the Fig. 2). In the case of the Gierłowska's lizard, the scan result is a scattered cloud of points that presents both the lizard skin and amber faces (Fig. 4a). The scan result in this form is unacceptable. It would require manual separation of both shapes - the lizard and the amber, which would require many hours of painstaking work of the operator. Widespread losses, especially in the cloud of points representing amber, would require manual replenishment, which in essence boils down to manual modeling of missing parts of amber shape.

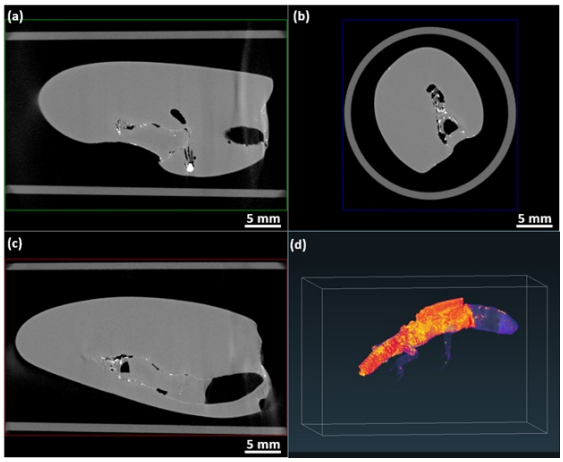


Figure 9: The μCT three perpendicular slices of the amber (a-c) and volume rendering of the Gierłowska's lizard hidden in amber

Tomograph

Three perpendicular slices of the amber are shown in Fig. 9 a-c, and the whole 3D volume renderings are displayed in Fig. 9d. The extraction of 3D objects and its visualization was one of the most important steps in the analysis of the pre-processed image data. The segmentation of lizard based on gray-scale gradient was not possible for volumetric approach. The lizard could not be sufficiently separated from the surrounding structures. Even by calculating the threshold for each slice separately, we failed. Therefore, we decided to use watershed algorithm with interactive techniques [Rus03a] and finally, the 3D model was exported from special segmentation software in 3D standard triangle language (STL) format.

Stereo camera and EBCA

Fig. 10 presents disparity maps obtained in the experiments with EBCA presented in this paper. The figure has a form of a table. The first raw corresponds to results of processing images of amber with a lizard. The second row presents results for amber with part of a leaf. The column containing images (a) and (f) shows results of using the StereoSGBM algorithm for a pair of images taken by a single stereo camera. The column with images (b) and (g) presents results for the same stereo matching algorithms used with the EEMM method and a set of five images. These are the results

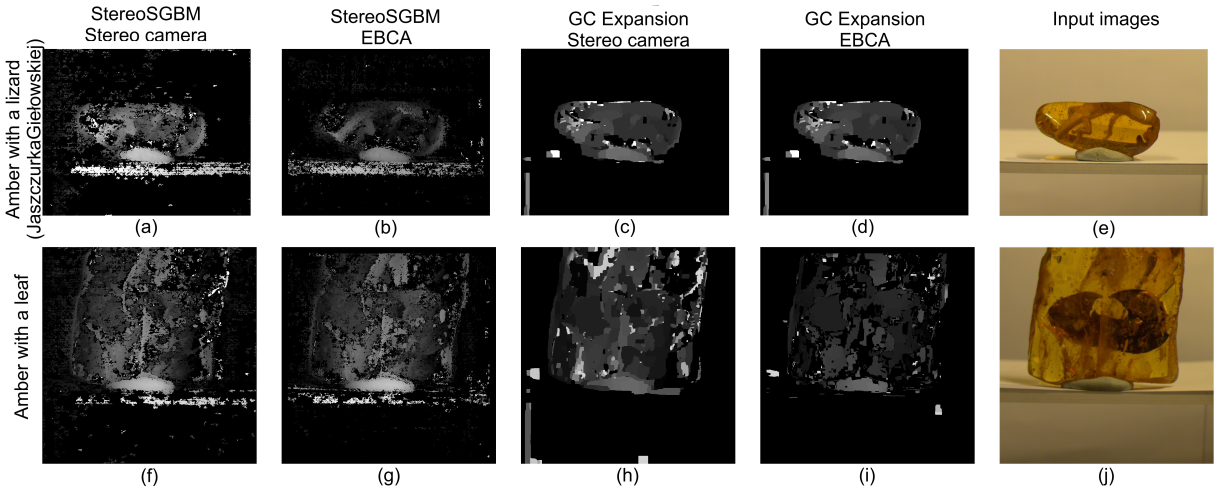


Figure 10: Results of using a stereo camera and Equal Baseline Camera Array for obtaining disparity maps of amber items

of executing the version of EEMM described in Sect. 4. Similarly, the third column presents results for the GC Expension algorithm used with a pair of images and the fourth column presents results for GC Expension applied to five image sets. The last column consists of raw images of ambers from which disparity maps were calculated.

The comparison of images presented in Fig. 10 shows differences between disparity maps obtained from two images and those obtained on the basis of five images. In both of these disparity maps there are areas with the same or similar value of a disparity depicted by a similar shade of gray in disparity maps. It can be noticed that when StereoSGBM algorithm is used edges of these areas are more irregular in case of disparity maps based on two images than in case of maps from five images. Moreover, two image disparity maps contain more parts in these areas with unknown value of disparity. These parts are more filled with disparity values in five image disparity maps. When GC Expansion is used than taking advantage of five images instead of two ones reduces the number of disparities for which the stereo matching algorithm obtained incorrect values. These disparities are visible in maps (c) and (h) as the brightest areas. Disparity maps based on five images contain only the most credible values of disparities. Instead of incorrect values of disparities these maps contain black values indicating that the disparity is unknown.

6 3D VISUALIZATION OF ACQUIRED AMBER SPECIMENS

Successful acquisition of 3D structure of amber specimens allows us to visualize the reconstructed objects, but the semitransparent nature of amber requires the use of advanced mathematical modeling and graphic rendering methods. A uniform 3D voxel grid, or a 3D tree such as an octree or a k-d tree can be used for modeling of acquired amber objects [Leb17c]. Such structures divide the object into parts and allows to define some quantities like the light transmittance for each part. Using the finite element method we can model ray tracing by solving a system of light transmittance equations corresponding to these individual parts. Such a method can simulate complex optical phenomena such as global illumination, subsurface scattering, caustics and internal reflections and offers very high quality of visualization [Leb17c].

A ray tracing engine for semitransparent materials is now being developed. It will allow the visualization of amber objects in virtual reality environments. We hope that the developing engine will be an attractive tool for curators of the Amber Museum and will allow them to prepare inviting museum exhibitions. An example of the expected capabilities of the engine is the virtual reconstruction of the Amber Room in the virtual reality cave located in the Immersive 3D Visualization Lab (Fig. 11) [Leb14a, Leb16b, Leb17c].

Precise reconstruction of the Gierłowska's lizard shape obtained thanks to the tomography allowed us to visualize the lizard itself as a zoological fossil [Leb20d]. Unfortunately, other methods do not yet provide adequate accuracy for reconstruction in VR. So we can see the three-centimeter lizard in the virtual reality cave with $100\times$ magnification. We can walk around it and view it from any angle. Rendezvous in virtual reality with a lizard that is 44 million years old and has been enlarged to the size of a crocodile leaves an indelible impression (Fig. 12). Particularly noteworthy are the lizard's surprisingly slender fingers.

Figure 11: Visualization of the Amber Room in a virtual reality cave

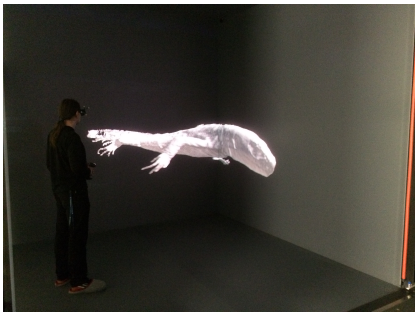


Figure 12: 3D reconstruction of the Gierłowska's lizard in a virtual reality cave with 100× magnification

7 CONCLUSIONS

The paper presents various classic scanning approaches applied to amber acquisition. The use of laser and structured-light scanners, microcomputed tomograph, stereo cameras and camera arrays were tested. The advantages and disadvantages of each of the approaches were described. None of the methods gave satisfactory results. The best results were obtained for tomography, but it is the most expensive and time-consuming method. In addition, it cannot cope with colors. Laser and structured light scanning as well as stereo cameras and EBCA methods capture the color correctly, but due to the translucency and multilayer structure of amber (anterior and posterior surfaces as well as inclusions and blemishes), the resulting reconstruction is burdened with significant inaccuracies. Scanning amber is a difficult task that requires further research. Amber as a structure of heterogeneous transparency, also changing abruptly (e.g. inclusions) and often with a glossy surface, seems to be one of the most difficult challenges in the acquisition of a 3D image. The conducted research has shown that it is necessary to develop special methods dedicated to amber scanning. At the moment, work is underway to develop proprietary amber scanning methods. An important element of this research is also the development of a mathematical model for the description of heterogeneity in semitransparent materials such as amber.

8 ACKNOWLEDGMENTS

This work was supported in part by ministry subsidies for local research and the special research infrastructure support grant 86/E-359/SPUB/SP/2019.

9 REFERENCES

[Boy01a] Y. Boykov, O. Veksler, R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Machine Intell.*, 23(11):1222–1239, Nov. 2001.

[Bra08a] G. R. Bradski, A. Kaehler. *Learning Opencv*. First edition. O'Reilly Media, 2008.

[Cla10a] N. Clark, C. Daly. Using confocal laser scanning microscopy to image trichome inclusions in amber. *J. of Paleontol. Techniques*, 8:1–7, 01 2010.

[Daz16a] J. D. Daza, E. L. Stanley, P. Wagner, A. M. Bauer, D. A. Grimaldi. Mid-cretaceous amber fossils illuminate the past diversity of tropical lizards. *Science Advances*, 2(3), 2016.

[Die07a] M. Dierick, V. Cnudde, B. Masschaele, J. Vlassenbroeck, L. Van Hoorebeke, P. Jacobs. Micro-ct of fossils preserved in amber. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, Proc. of the 10th Int. Symp. on Radiation Physics, 580(1):641–643, 2007.

[Dis18a] S. Discher, L. Masopust, S. Schulz, R. Richter, J. Döllner. A Point-Based and Image-Based Multi-Pass Rendering Technique for Visualizing Massive 3D Point Clouds in VR Environments. *J. of WSCG*, 26(2):76–84, 2018.

[Dun11a] J. A. Dunlop, D. Penney, N. Dalüge, P. Jäger, A. McNeil, R. S. Bradley, P. J. Withers, R. F. Preziosi. Computed tomography recovers data from historical amber: an example from huntsman spiders. *Naturwissenschaften*, 98(6):519–527, Jun. 2011.

[Hir08a] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Machine Intell.*, 30(2):328–341, Feb. 2008.

[Ihr10a] I. Ihrke, K. N. Kutulakos, H. P. A. Lensch, M. Magnor, W. Heidrich. Transparent and specular object reconstruction. *Comput. Graph. Forum*, 29(8):2400–2426, 2010.

[Kac15a] A. L. Kaczmarek. Improving depth maps of plants by using a set of five cameras. *Journal of Electronic Imaging*, 24(2):023018, 2015.

[Kac17b] A. L. Kaczmarek. Stereo vision with equal baseline multiple camera set (ebmcs) for obtaining depth maps of plants. *Computers and Electronics in Agriculture*, 135:23–37, 2017.

[Kac19c] A. L. Kaczmarek. 3d vision system for a robotic arm based on equal baseline camera array. *Journal of Intelligent & Robotic Systems*, 2019.

[Kam14a] T. van de Kamp, T. Rolo, T. Baumbach, L. Krogmann. Scanning the past - synchrotron x-ray microtomography of fossil wasps in amber. *Entomologie heute*, 26:151–160, 11 2014.

[Keh14a] C. Kehlmaier, M. Dierick, J. Skevington. Micro-ct studies of amber inclusions reveal internal genitalic features of big-headed flies, enabling a systematic placement of metanephrocerus aczél, 1948 (insecta: Diptera: Pipunculidae). *Arthropod Systematics and Phylogeny*, 72:23–36, 04 2014.

[Kit21a] KITTI Vision Benchmark Suite, http:www.cvlibs.netdatasetskittiindex.php.

[Kos08a] B. Kosmowska-Ceranowicz. Glowing stone: Amber in polish deposits and collections. *Przegląd Geologiczny (Geological Rev.)*, 56(8/1):604–610, 2008.

[Kos97b] B. Kosmowska-Ceranowicz, R. Kulicka, G. Gierłowska. A new find of lizard in baltic amber (in polish). *Przegląd Geologiczny (Geological Rev.)*, 45(10):1028–1030, 1997.

[Kyo13a] T. Kyöstilä, D. C. Herrera, J. Kannala, J. Heikkilä. Merging overlapping depth maps into a nonredundant point cloud. *Image Analysis*, SCIA, LNCS, 7944:567–578, 2013.

[Leb14a] J. Lebiedź, A. Mazikowski. Innovative Solutions for Immersive 3D Visualization Laboratory. *22nd Int. Conf. on Computer Graphics, Vis. and Computer Vision WSCG 2014 - Comm. Papers Proc.*, 315–319, 2014.

[Leb16b] J. Lebiedź, J. Redlarski. Applications of immersive 3d visualization lab. *24th Int. Conf. on Computer Graphics, Vis. and Computer Vision WSCG 2016 - Poster Papers Proc.*, CSRN 2603:69–74, 2016.

[Leb17c] J. Lebiedź, J. Redlarski, J. Rumiński. Virtual designs and reconstructions of amber works: Amber craftsman simulator. *2017 10th Int. Conf. on Human System Interactions* HSI, 246–249, July 2017.

[Leb20d] J. Lebiedź. Virtual reconstruction of Gierłowska's Lizard. *The Amber Magazine* 44:94–97, Mar. 2020, https://www.amber.org.pl/bursztynisko.

[Lev85a] M. Levoy, T. Whitted. The use of points as a display primitive. *Technical Report* 85-022, Computer Science Department, University of North Carolina at Chapel Hill, Jan. 1985, https:graphics.stanford.edupaperspoints.

[Li15a] H. Li, X. Wang, Y. Zhu. Identification characteristics for amber and its imitation. Proc. of the 5th Int. Conf. on Info. Engr. for Mech. and Mtls., 483–488, 2015.

[Mid21a] Middlebury Stereo Vision Page, http:vision.middlebury.edustereo.

[Mor16a] J.-D. Moreau, D. Néraudeau, V. Perrichot, P. Tafforeau. 100-million-year-old conifer tissues from the mid-Cretaceous amber of Charente (western France) revealed by synchrotron microtomography. *Ann. of Botany*, 119(1):117–128, 12 2016.

[Pol02a] M. J. Polcyn, J. V. Rogers II, Y. Kobayashi, L. J. Jacobs. Computed tomography of an anolis lizard in dominican amber: Systematic taphonomic, biogeographic, and evolutionary implications. *Palaeontologia Electronica*, 5, 01 2002.

[Qia18a] Y. Qian, Y. Zheng, M. Gong, Y.-H. Yang. Simultaneous 3D Reconstruction for Water Surface and Underwater Scene. *Proc. of the Eur. Conf. on Computer Vision* ECCV, 776–792, 2018.

[Rus03a] J. C. Russ. *The Image Processing Handbook*. Fourth Edition. CRC Press 2002.

[Sel17a] P. A. Selden, D. Penney. Imaging techniques in the study of fossil spiders. *Earth-Science Rev.*, 166:111–131, 2017.

[Sch02a] D. Scharstein, R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. of Computer Vision*, 47(1):7–42, 2002. Microsoft Research Tech. Rpt. MSR-TR-2001-81, Nov. 2001.

[Ste16a] F. Stebner, R. Szadziewski, P. T. Rühr, H. Singh, J. U. Hammel, G. M. Kvifte, J. Rust. A fossil biting midge (diptera: Ceratopogonidae) from early eocene indian amber with a complex pheromone evaporator. *Sc. Rpt.*, 6:34352, 2016.

[Wim06a] M. Wimmer, C. Scheiblauer. Instant Points: Fast Rendering of Unprocessed Point Clouds. *SPBG*, 129–136, July 2006.

[Wu18a] B. Wu, Y. Zhou, Y. Qian, M. Gong, H. Huang. Full 3D reconstruction of transparent objects. *ACM Trans. Graph.*, 37(4), Article 103, Aug. 2018.

[Zak19a] A. M. Zakharenko, K. S. Golokhvast. Using confocal laser scanning microscopy to study fossil inclusion in baltic amber, a new approach. *Physics and Technology of Nanostructured Mtls. IV* , 806:192–196. Trans. Tech. Pub., 7 2019.

LEAVEN - Lightweight Surface and Volume Mesh Sampling Application for Particle-based Simulations

Alexander Sommer[1] Ulrich Schwanecke[1]
alexander.sommer@hs-rm.de ulrich.schwanecke@hs-rm.de

[1] Computer Vision and Mixed Reality Group, RheinMain University of Applied Sciences Wiesbaden Rüsselsheim, Germany

ABSTRACT

We present an easy-to-use and lightweight surface and volume mesh sampling standalone application tailored for the needs of particle-based simulation. We describe the surface and volume sampling algorithms used in LEAVEN in a beginner-friendly fashion. Furthermore, we describe a novel method of generating random volume samples that satisfy blue noise criteria by modifying a surface sampling algorithm. We aim to lower one entry barrier for starting with particle-based simulations while still pose a benefit to advanced users. The goal is to provide a useful tool to the community and lowering the need for heavyweight third-party applications, especially for starters.

Keywords: mesh sampling, volume representation, surface representation, particle-based simulation

1 INTRODUCTION

In many fields of computer graphics, the plausible simulation of different physical phenomena is an important topic. As simulations hit interactive framerates, the urge for a unified simulation solver allowing to simulate various phenomena in one framework grew. A lot of methods like position-based dynamics [16] and their successors unified particle physics [14] and projective dynamics [3][19] use particles to represent all various simulation objects, like rigid-bodies, cloth, fluid, granular material, gases, and deformable solids. Further, in other non-real-time simulation methods, like Smoothed Particle Hydrodynamics (SPH) [18] or material point methods [9], particle representations play an important role. They are not only used for characterizing parts of a continuum but also in the efficient and correct handling of object- and domain boundaries [2].

We experienced that especially for beginners generating a particle representation of different geometrical objects is an entry barrier for getting started with particle-based simulations. There are a certain number of well-described algorithms for generating surface representations, as well as some algorithms for generating suitable volume representations (see Section 2).

Beginners often struggle to find a convenient and ready-to-use implementation or tool that is suitable for the requirements of particle-based simulation. The use of heavy-weight third-party software for sampling can also introduce new issues. These programs can be hard to access and learning how to work around the vast amount of functions can be unnecessarily time-consuming. Furthermore, data-export is often limited to undocumented binary data formats.

Therefore, we introduce LEAVEN, a Lightweight surfacE And Volume mEsh sampliNg application tailored to the needs of particle-based simulations. LEAVEN is an easy-to-use graphical application that enables beginners to sample a surface or volume of a 3D triangle mesh with only a few straightforward clicks. Furthermore, it leaves enough customization options to be valuable for experienced simulators. LEAVEN is made available as open-source and can be used as a library in other projects.

Our paper is organized as follows: first, we give a brief overview of sampling algorithms (Section 2), then we describe the algorithms used for sampling in our application (Section 3), followed by a detailed description of the developed application (Section 4) and its usage (Section 5). Our main contributions are:

- Giving a beginner-friendly introduction to particle sampling techniques
- Lowering the entry barrier for particle-based simulations
- Modifying a surface sampling method to generate volume samples that satisfy blue noise criteria
- Contributing an easy-to-use, lightweight, open-source standalone application for sampling the surface and volume of arbitrary closed triangle meshes for use in particle-based simulations

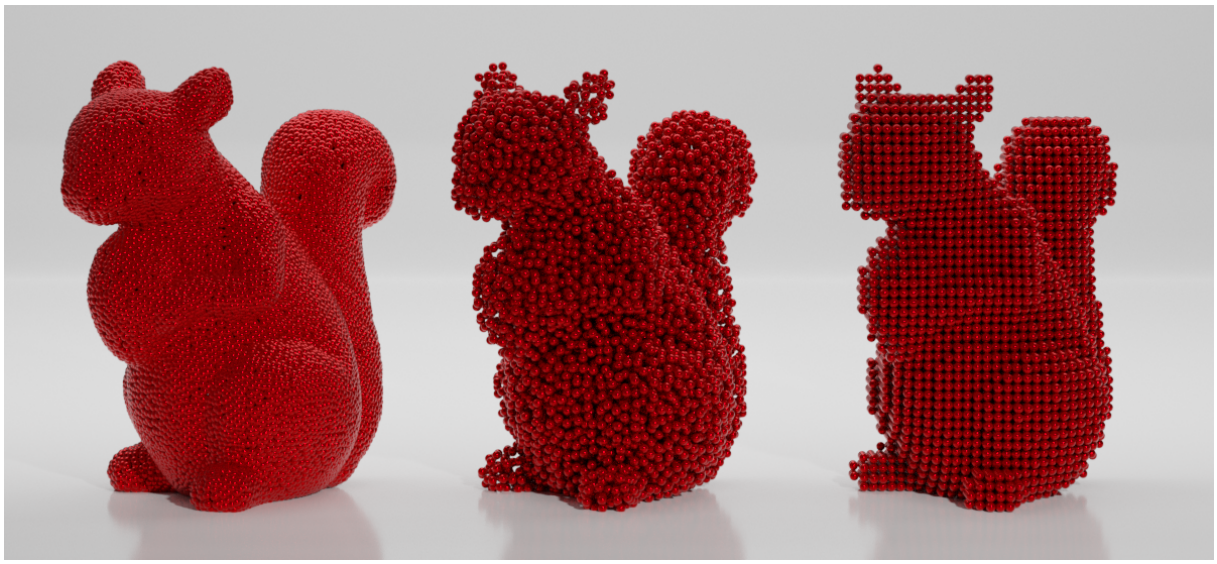


Figure 1: Different samplings in comparison. Surface sampling (left), volume sampling with randomly distributed particles (middle), and volume sampling with grid-based packing (right).

2 RELATED WORK

Various fields of computer graphics, such as texturing, remeshing, rendering, modeling, or animating fractures, have methods for sampling surfaces or volumes that are tailored to their specific needs. An important sampling criterion that many of these methods have in common is that the sampling behaves like blue noise. Blue noise distributions have been an important research topic [8][17][6]. A sampling that has blue noise characteristics is uniform and unbiased distributed in the spatial domain. Furthermore, when looking at the frequency domain with the Fourier transformed spatial information, there should be a lack of low-frequency noise and structural bias. One popular method for generating samplings that satisfy blue noise criteria is Poisson disk sampling [15][5]. In Poisson disk sampling, the samples are placed randomly on the valid spatial domain, which is the surface of an object for surface sampling or its volume for volume sampling. After placing the samples, a validation process guarantees that no two samples are too close to each other. Cline et. al. [7] introduced an optimized dart-throwing algorithm for generating Poisson disk point sets as a sampling of the surface of a 3D triangle mesh. Their algorithm needs sequential processing and has its applications in modeling and polygon remeshing. Wei et. al. [23] present a parallel dart-throwing algorithm by drawing suitable samples from a dense point cloud on the object's surface. This parallel version can achieve interactive frame-rates even for a large number of samples and is therefore well suited for real-time applications. Bowers et. al. [4] extended this algorithm to work not only with the Euclidean but also with a fast approximation of the Geodesic norm, as a measurement of sample point distances. This method is well suited for interactive texturing applications or other tasks where quick sampling is needed. Yuksel [24] describes an approach of generating surface samplings without using a form of dart-throwing but rather assigning weights to possible samples. He uses a greedy algorithm to eliminate the points with the highest weight. Wang, T. et. al. [21] propose another rapid parallel surface sampling method. They use blue-noise pattern planes that they progressively project on the mesh surface using ray tracing. This method has its applications especially when fast sampling is required. A novel data-driven approach by Wang, Z. et. al. [22] promises to achieve even better performance than the classic geometrical approach on triangle meshes.

In contrast to the previously mentioned techniques, Jiang et. al. [11] use different methods inspired by SPH simulations. They are not only generating surface samplings but also volume samplings in that way. Their idea is to treat each sampling point as an SPH particle and establish a constant density in SPH kernel estimation functions by applying correction forces. Adams et. al. [1] describe an adaptive volume sampling method for particle-based fluid simulation. They use a sampling condition to put more focus on geometrically complex simulation regions.

3 ALGORITHMS

The main algorithms used in LEAVEN can be divided into surface- and volume sampling algorithms. In particle-based simulations, surface representations are typically used for boundary handling and sometimes for two-way coupling, while volume samplings are used for initial particle positions or rigid-body representations in unified solvers. They both have in common that they need a discrete 3D representation

Model	BBox Scaling	Surface Sampling		Random Volume Sampling		Grid Volume Sampling	
		time	particles	time	particles	time	particles
Bunny	2.0	13ms	9122	293ms	14875	100ms	22206
Squirrel	3.0	30ms	14889	573ms	31795	109ms	48012
Armadillo	6.0	87ms	53921	5096ms	145403	2559ms	215141
Dragon	12.0	595ms	233355	23444ms	943703	6710ms	1445523

Table 1: Timing results for different meshes and varying particle numbers resulting from different scales of the bounding box and a fixed particle radius of 0.02m for volume sampling / minimum distance for surface sampling.

as an input. 3D geometry in computer graphics is often represented as polyhedrons, with a polygon mesh defining the surface of the object. We use triangle meshes as input since these meshes are most common, easy to work with, and accessible. For volume sampling, the input mesh describing the surface of an object must be closed to have a distinguishable inside and outside. Figure 1 shows different samplings of the same 3D object. On the left the surface sampling algorithm is applied, where particles can get as close as half the particle diameter. Shown in the middle is our approach to generating random blue noise samples within a volume, with a minimum particle spacing equal to the particle diameter. A grid-based volume sampling is shown on the right. Table 1 shows timing results for the three algorithms available in LEAVEN on different standard meshes.

3.1 Surface Sampling

Akinci et. al. [2] describe how boundary particles can be a suitable boundary representation in SPH fluid simulation and how to calculate a two-way fluid-rigid coupling. Later this practice has been used for boundary handling in position-based methods like position-based fluids [13], or for the simulation of gases and granular material [10].

There are some basic requirements for a good surface representation with particles. First of all, the sampling should be dense enough that no simulation particle can tunnel through holes between boundary particles. Furthermore, the particles should be spread out uniformly on the object's surface to guarantee an even force distribution to prevent artifacts.

To generate such a sampling of the surface of an object, we use a uniform sampling algorithm on arbitrary triangle meshes introduced by Bowers et. al. [4]. The method builds upon a grid cell sampling approach to generate Poisson disk samples on arbitrary manifold surfaces [23]. The goal of the sampling process is to find a set of sample points (particles) $\mathbb{S}$ that is randomly but uniformly distributed on the surface ∂V of a 3D Volume V. Thereby all particles should have a minimum distance d to each other, i.e.

$$\text{dist}(\boldsymbol{s}_i, \boldsymbol{s}_j) \geq d, \ \forall \boldsymbol{s}_i, \boldsymbol{s}_j \in \mathbb{S}$$

For most cases in particle-based simulation, d equals the radius r of the simulation particles. The distance metric $\text{dist}(\cdot,\cdot)$ measuring the spatial separation can be Euclidean or Geodesic. The Euclidean distance measures the shortest line segment between two points in 3D space. The Geodesic distance measures spatial separation with respect to the object's surface as a submanifold. This can be beneficial for achieving a good sampling of complex objects with thin features. For a detailed explanation of Geodesic distances and their fast approximation, we refer the reader to [4].

The algorithm starts initially by calculating a much larger set $\mathbb{P}$ with candidate positions on the surface ∂V without bothering about the distances between them. The candidate samples are found by picking a random triangle from the triangle mesh with a probability proportional to the triangle's area. On this triangle, an arbitrary position $\boldsymbol{p}$ is generated by choosing two random values $\tau_1, \tau_2 \in [0,1]$ for its barycentric coordinates

$$u = 1 - \sqrt{\tau_1}, \quad v = \tau_2 \sqrt{\tau_1}, \quad w = 1 - u - v$$

inside the triangle, which leads to a random position

$$\boldsymbol{p} = u\boldsymbol{a} + v\boldsymbol{b} + w\boldsymbol{c},$$

where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ are the three vertices of the triangle. In this manner, the initial set $\mathbb{P}$ is filled with a large number of candidate positions. In the application LEAVEN, we generate $\rho_S \cdot \frac{A(\partial V)}{\pi r^2}$ positions in this way, where $A(\partial V)$ is the surface area, which is the sum of all triangle areas and ρ_S is a controllable density parameter. We use a default value of $\rho_S = 40$. After this, the bounding box containing V is sliced into grid cells with a uniform side length of $\frac{d}{\sqrt{3}}$. This leads to a cell diagonal of d. For each position, $\boldsymbol{p} \in \mathbb{P}$ the corresponding grid cell id is calculated and the whole data array containing $\mathbb{P}$ is sorted by the grid cell id. Each cell that contains at least one position $\boldsymbol{p}$ is a valid surface cell. Cells without candidate positions are much likely located outside or inside the volume.

Since the majority of cells is empty a hash table is a suitable lookup structure for finding valid cells and their positions. The cell id is used as the hash key. In contrast to [4], we use a spatial hashing function with bit-wise `xor`:

$$(i \cdot p_1 \ \texttt{xor} \ j \cdot p_2 \ \texttt{xor} \ k \cdot p_3) \ \% \ \texttt{size(hash\_table)}$$

where

$$p_1 = 73856093, \ p_2 = 19349663, \ p_3 = 83492791$$

as described in [20], where i, j, k are the cell indices, to prevent hash collisions. Each hash bucket contains the cell id, a pointer to the first position $\boldsymbol{p}$ that is associated with this cell id in the concurrent sorted data array, and if already chosen, the sampled position. In the end, each valid surface cell will contain at most one sample.

Algorithm 1 Uniform surface sampling

```
1:  generateCandidateSetP()
2:  sortSetPByCellId(setP)
3:  insertSetPInHashMap(setP)
4:  for trial t = 1,...,n do
5:      for all valid cells i do
6:          if cell i is already sampled then
7:              break
8:          if t-th candidate point p_{i_t} doesn't exist then
9:              break
10:         conflict = false
11:         for all neighbor cells j do
12:             if j not a valid cell then
13:                 break
14:             if j has no sample s_j yet then
15:                 break
16:             if dist(p_{i_t}, s_j) < d then
17:                 conflict = true
18:         if conflict == false then
19:             sample for cell i: s_i = p_{i_t}
20:             put s_i in S
```

The main sampling algorithm consists of several trials n, which is another controllable parameter. A higher number of trials n makes it more likely to find a sample for each valid cell but needs more computation time. We employ a default value of $n = 10$. In each trial $t = 1,\ldots,n$, each valid cell is processed. If the cell already contains a valid sample it is skipped. If there are no more candidate positions in $\mathbb{P}$, the cell can also be neglected. Otherwise, neighboring cells are checked to see if any already sampled position is too close to the current candidate position $\boldsymbol{p}_{i_t}$. If this is not the case $\boldsymbol{p}_{i_t}$ is added to the set of samples $\mathbb{S}$. The whole sampling algorithm is shown in Algorithm 1. For a parallel version and further details, we refer the reader to [4].

3.2 Volume Sampling

In contrast to surface sampling, volume sampling is used to define initial positions of simulation particles, whether it may be free-moving particles like fluid, gas, or granular material, or a rigid-body representation in a unified solver like unified particle physics [14]. Therefore the requirements for the sampling are different. For initial positions the particles mustn't be primarily in an invalid state, meaning that the particles are not allowed to overlap. Overlapping initial positions can lead to high correction forces/position changes, causing the simulation to explode.

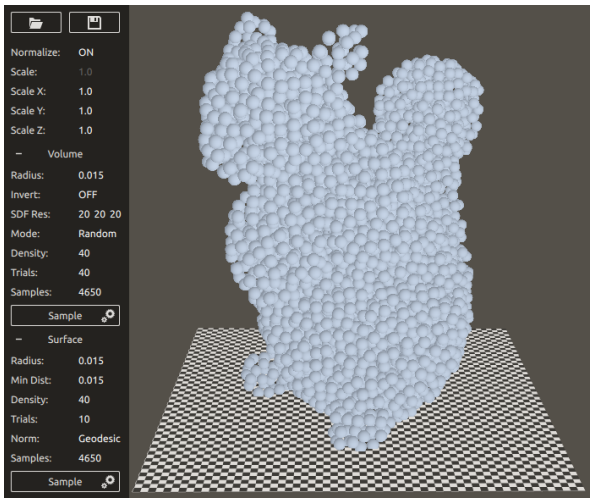


Figure 2: LEAVEN's application window with a triangle mesh sampled with 4650 randomly distributed particles through the volume.

Particles that represent a rigid-body should maintain a constant distance from each other during simulation. Therefore it makes sense to sample them closely and uniformly inside the volume V. The goal is to find a set of sample positions $\mathbb{S}$, that is spaced out on an equidistant grid with a minimal side length of $2 \cdot r$, where r is the particle radius. For deciding which positions are inside the Volume V we use a signed distance field (SDF). It is defined as a signed distance function $\Psi : \mathbb{R}^3 \to \mathbb{R}$,

$$\Psi(\boldsymbol{p}) = \text{sgn}(\boldsymbol{p}) \inf_{\boldsymbol{p}^* \in \partial V} \|\boldsymbol{p} - \boldsymbol{p}^*\| \quad \text{with}$$

$$\text{sgn}(\boldsymbol{p}) = \begin{cases} -1 & \boldsymbol{p} \in V \\ 1 & otherwise \end{cases}$$

measuring the Euclidean distance of a point $\boldsymbol{p} \in \mathbb{R}^3$ to the nearest point $\boldsymbol{p}^*$ on the object surface ∂V. Generating such a parametric SDF representation of an arbitrary closed Volume V is a topic on its own and beyond the scope of this paper. In LEAVEN we use a grid-based SDF with hierarchical cell size and polynomial degree refinement based on polynomial fitting as described in [12]. The general idea behind this is to fit polynomial functions in cells of axis-aligned hexahedral grids defined by the input meshes. The algorithm iteratively refines the SDF either by spatial subdivision or cell-by-cell adjustment of the degree of the polynomial.

After generating the SDF, the bounding box containing V is divided into grid cells similar as described in Section 3.1 but with a cell size of $2 \cdot r$. For each cell i one sampling particle is added to the cell center at position $\boldsymbol{p}_i$. Then the signed distance function is evaluated as position $\boldsymbol{p}_i$. When $\Psi(\boldsymbol{p}_i) < 0$ the position is inside the Volume V and therefore it is a valid sampling position $\boldsymbol{s}_i$, that can be added to the sampling set $\mathbb{S}$ (see Algorithm 2).

Figure 3: Volume and surface samplings used in a unified simulation framework simulating a granular material and a rigid-body (red) moved by an excavator.

Algorithm 2 Grid-based volume sampling

```
generateSDF()
for all cells i do
    position p_i = cellCenter
    if Ψ(p_i) < 0 then
        sample s_i = p_i
        put s_i in 𝕊
```

Particles that represent parts of fluids, gases, or granular materials, in general, change their distance to each other during the simulation. Sampling them in regular patterns, like previously, can introduce disturbing visual artifacts at the beginning of a simulation. Therefore we modify the concept of Bowers et. al. for surface sampling [4] explained in Section 3.1 to present a novel approach for generating random samples inside a 3D volume that satisfies blue-noise criteria.

Our modified sampling method divides the bounding box of V into uniform grid cells with a side length of $\frac{2 \cdot r}{\sqrt{3}}$ resulting in N cells. We fill the set of candidate positions $\mathbb{P}$ by picking uniformly distributed positions within the whole bounding box and only adding the positions $\boldsymbol{p}$ to $\mathbb{P}$ that lie inside the volume (i.e. $\Psi(\boldsymbol{p}_i) < 0$). In this way, we generate $\rho_V \cdot N$ samples, where the number of samples generated is proportional to the number of grid cells in the bounding box. The proportionality constant ρ_V is a controllable density parameter. We recommend this parameter to be equal to the number of trials n. The rest of the sampling process is the same as for the surface sampling algorithm with Euclidean distance norm (see Algorithm 1).

4 APPLICATION

As mentioned before, the main goal with LEAVEN is to provide an easy-to-use tool suitable for beginners and advanced users in particle-based simulation. In the following, we will briefly explain the settings the user can manipulate in the application[1].

Figure 2 shows the LEAVEN user interface (UI). On the righthand side of the UI is the viewer. It displays the result of the sampling process. The depicted 3D model can be manipulated by arcball rotation, zooming, and panning. The lefthand side of the UI contains the settings for the sampling process. When mesh normalization is turned on, the mesh is uniformly scaled to fit inside a bounding box with a side length of 1. The mesh can be scaled further up or down by a factor. It is also possible to scale each axis individually, for example, to easily generate rectangular boxes from a unit cube for non-cubic simulation domains.

For volume sampling, the radius of the sampling particles can be specified. When sampling is inverted, the volume between the bounding box and the mesh outside is sampled. The SDF Resolution defines the accuracy of the parametric volume representation. For objects with finer details, a higher resolution is needed. The grid mode samples the particles uniformly and aligned on a grid inside the volume. Random mode generates randomized sampling inside the volume that satisfies blue noise criteria as described in Section 3.2. In random mode the controllable density parameter ρ_V and the number of trials n can be set.

[1] https://github.com/a1ex90/Leaven

For surface sampling, the radius of the displayed sampling particles can also be specified. This only affects the visual appearance of the particles in the application's preview display and should not be confused with the *Min Dist* parameter, which controls the minimum distance d between particles that influences the sampling (see right side of Figure 2). For boundary handling, a good rule of thumb for choosing d is to use the same size as the simulation particle radius r. Furthermore, the controllable density parameter ρ_S and the number of trials n can be chosen (see Section 3.1). It is also possible to switch between Geodesic or Euclidean distance norm for sampling particle distances. For more fine details in the mesh, the geodesic norm should be the one in favor.

5 USAGE EXAMPLE

Figure 3 shows a simulation scene where our sampling has been applied to many various objects. The scene has been simulated with the unified particle physics [14] variant of position-based dynamics [16]. A granular material with friction, behaving like sand, is simulated. Initial particle positions (sand-colored) are sampled with LEAVEN's volume sampling random mode on. The rigid-body, a squirrel, is simulated within the same framework with a two-way coupling with the granular material. It is sampled with grid-based volume sampling (red). The excavator (yellow dots), as well as the domain boundaries (blue dots), are surface sampled with a minimal distance between samples equal to the particle radius.

6 CONCLUSION

With this paper we contribute an easy-to-use application for representing surfaces and volumes by a set of particles. We used a state-of-the-art surface sampling algorithm [4] and present a novel approach for generating volume samples that satisfy blue noise criteria by using an SDF representation of the volume. Furthermore, we give a beginner-friendly introduction to the field of sampling techniques tailored for the needs of particle-based simulation.

REFERENCES

[1] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. *ACM Trans. Graph.*, 26(3):48–es, July 2007.

[2] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4), July 2012.

[3] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4):154:1–154:11, July 2014.

[4] John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.*, 29(6), December 2010.

[5] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07, page 22, New York, NY, USA, 2007. ACM.

[6] Jiating Chen, Xiaoyin Ge, Li-Yi Wei, Bin Wang, Yusu Wang, Huamin Wang, Yun Fei, Kang-Lai Qian, Jun-Hai Yong, and Wenping Wang. Bilateral blue noise sampling. *ACM Trans. Graph.*, 32(6), November 2013.

[7] David Cline, Stefan Jeschke, K. White, Anshuman Razdan, and Peter Wonka. Dart throwing on surfaces. *Computer Graphics Forum*, 28:1217 – 1226, 08 2009.

[8] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, January 1986.

[9] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph.*, 37(4), July 2018.

[10] Markus Ihmsen, Arthur Wahl, and Matthias Teschner. High-Resolution Simulation of Granular Material with SPH. In Jan Bender, Arjan Kuijper, Dieter W. Fellner, and Eric Guerin, editors, *Workshop on Virtual Reality Interaction and Physical Simulation*. The Eurographics Association, 2012.

[11] Min Jiang, Yahan Zhou, Rui Wang, Richard Southern, and Jian Jun Zhang. Blue noise sampling using an sph-based method. 34(6), October 2015.

[12] Dan Koschier, Crispin Deul, Magnus Brand, and Jan Bender. An hp-adaptive discretization algorithm for signed distance field generation. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 07 2017.

[13] Miles Macklin and Matthias Müller. Position based fluids. *ACM Trans. Graph.*, 32(4), July 2013.

[14] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4), July 2014.

[15] Michael Mccool and Eugene Fiume. Hierarchical poisson disk sampling distributions. *Proceedings - Graphics Interface*, 05 1992.

[16] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109 – 118, 2007.

[17] Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph.*, 23:488–495, 08 2004.

[18] Barbara Solenthaler and Renato Pajarola. Predictive-corrective incompressible sph. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, New York, NY, USA, 2009. ACM.

[19] Alexander Sommer, Ulrich Schwanecke, and Elmar Schoemer. Chebyshev's method on projective fluids. *Journal of WSCG*, 28:132–136, 2020.

[20] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomeranets, and Markus Gross. Optimized spatial hashing for collision detection of deformable objects. *VMV'03: Proceedings of the Vision, Modeling, Visualization*, 3, 12 2003.

[21] Tong Wang and Reiji Suda. Fast generation of poisson-disk samples on mesh surfaces by progressive sample projection. *Proc. ACM Comput. Graph. Interact. Tech.*, 1(2), August 2018.

[22] Zong-Sheng Wang, Jung Lee, Chang Geun Song, and Sun-Jeong Kim. Data-driven point sampling with blue-noise properties for triangular meshes. In *Proceedings of the 2020 3rd International Conference on Computer Science and Software Engineering*, CSSE 2020, pages 77–82, New York, NY, USA, 2020. Association for Computing Machinery.

[23] Li-Yi Wei. Parallel poisson disk sampling. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, New York, USA, 2008. ACM.

[24] Cem Yuksel. Sample elimination for generating poisson disk sample sets. *Computer Graphics Forum*, 34, 05 2015.

Model-Based Tracking on Conveyor Belts: Evaluation and Practical Results in the Automotive Industry

Fabian Scheer
Daimler Protics GmbH
Fabian.scheer@daimler.com

Moritz Loos
Daimler Protics GmbH
Moritz.loos@daimler.com

Markus Neumann
Daimler Protics GmbH
Markus.n.neumann@daimler.com

ABSTRACT

Model based tracking (MBT) of painted cars in the automotive mass production on conveyor belts with robots is a challenging task. Many disturbing sources that have an impact on the MBT exist, like the influence of the localized work illumination, the synchronization of the MBT to the conveyor belt, reflections in the paint, variants of the cars and the complexity of the used CAD models. By having such complex systems the mere assessment of the accuracy and stability of MBT approaches by literature can be hard. A real world application is necessary for a better understanding. Therefore, we present the evaluation of MBT for a robotic gap measurement system on painted cars. The influence of local lighting and car paint is analysed in detail regarding the MBT accuracy. To reduce complexity considering the car variants on a production line, we evaluated the MBT with different model setups and show the influence on the MBT results. Regarding MBT in a complex calibrated system that runs twenty-four-seven, a broken or slightly displaced camera should not have a huge impact like a loss of production. For this reason, we present a method to exchange a camera within minutes and without a loss of the overall accuracy. The method relies on a separate test specimen and is evaluated in detail. The presented evaluations can help researchers and the industry to better understand and assess the influence and correlations of different error sources or disturbing factors for the usage of MBT in complex conveyor belt based robotic applications.

Keywords

Application, evaluation, model-based tracking, coated cars, synchronization to conveyor, robotic production line

1. INTRODUCTION

By using MBT in complex systems a lot of correlations exist between different influencing factors, like the reflectivity of considered objects, the lighting situation, the optimal selection of the used CAD model or the synchronization to other system components or machines. During research in the literature we found that such relationships are often sparsely considered for MBT approaches and that there is a lack of detailed evaluations of MBT in action. Therefore, we present evaluations of different factors that have an influence to the results of MBT in a complex system. The industrial use case presented in this paper is the automatic measurement of gap points on painted cars on a conveyor belt with a light weight robot. To avoid a crash of the robot with the car, an MBT approach is used that measures the car. On this basis, the robot can correct its teached coordinates with the actual tracking results. Such automated vision based systems play a more and more important role in the context of future manufacturing systems. Various international initiatives like the industrial internet consortium, industry 4.0 or smart factories support these ambitions. Against this background, efficient and pragmatic solutions, e.g. maintenance models, for the every day use of such systems are necessary. Based on this motivation, we present a fast method to handle the exchange of broken or displaced cameras without the need of initial measurements for the total system. This method is evaluated in detail and the results are presented and discussed. The presented methods and evaluations in this paper can help researchers and the industry to better assess the behaviour and accuracy of MBT in complex systems and to better understand the impact of different influencing factors and their correlations.

2. RELATED WORK

[VP05] gives an overview of monocular model-based tracking approaches in a survey. The authors give a categorized overview to support the selection process which approach may be suitable for a use case, but it is a very rough overview without many details. In-depth evaluations that consider different influencing factors are missing.

[Gar18] summarises state of the art frameworks and datasets for the evaluation of six degrees of freedom object trackers. They limit their method to the criteria of stability, robustness to occlusion and accuracy during challenging interactions.

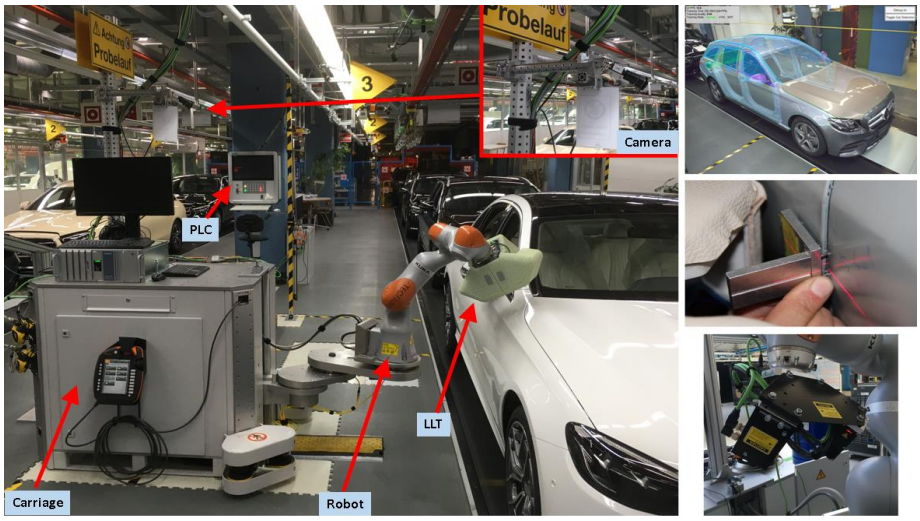


Figure 1: System Components. The camera mount is zoomed. Top right: Overlay of CAD data to video image. Middle right: template for robot teaching. Bottom right: Laser line tool without protective cover.

Several evaluation techniques use fiducials like [Hin11] or a checker board [Wu17]. Therefore, the quality of ground truth information is limited.

In [Hod17] a summary of the state of the art can be found for the evaluation of 6D pose estimation for textureless objects. They also use fiducials to create ground truth data.

In our former work [Sch20], MBT was evaluated with devices of the measurement technology and an detailed overview of the influence and correlations of several factors was given. A robotic conveyor belt based gap measurement application was shown, but limited to body shells. In this paper our work is extended by considering painted cars that are completely assembled. Section 3.2 recapitulates the used application. Different gloss paints with light reflections are challenging for the MBT. Therefore, different lighting situations in correlation with the car's gloss paint, the optimal CAD model selection and the accuracy of the total system will be evaluated. Detailed results for the gloss paints will be given. Furthermore, a fast method to exchange a camera in the system will be shown and evaluated in detail. It merely introduces a small, acceptable error regarding the system's total accuracy.

In this paper, the MBT approach of [Wue07] that is based on [Com06] and [Vac04] was used for all evaluations. The authors evolved the approach and reached a high degree of maturity. The results in [Sch20] prove that with this MBT a high degree of accuracy and robustness in industrial applications can be achieved. Since we also focus on an industrial use case, this was important in the selection process for the MBT. Only one MBT approach is considered in this paper, due to the great expense for all presented evaluations. To solve the initialization problem, [Wue07] relies on a real camera perspective that closely matches a specified CAD perspective. If both perspectives match, a continuous tracking takes place for subsequent frames and the CAD model can be augmented on the video image (see Figure 1,top right).

3. System and workflow overview

Our system uses MBT to track painted cars on a conveyor belt. Tracking results are sent to a light weight robot to navigate to predefined points on the outer shell of a car where gaps are measured. By using the MBT results the robot avoids collisions with a car.

3.1 Motivation

Industrial robot applications rely on an initial setup and configuration stage. In this stage, all components are in a defined state. This is necessary to calibrate the different components and put the system into operation. For our use case, a painted car is put on the conveyor belt in an initial position. In this position measurement points on the outer shell of the car are taught to the robot. During production, the painted cars are driven onto the conveyor belt by workers. Therefore, the pose of the cars deviates from the pose of the car in the initial position. Thus, without any pose correction the robot could collide with the cars on the conveyor belt during production. Furthermore, the tire pressure of the cars can lead to the effect that the robots are not measuring the exact gap position on every car. To avoid the described problems a pose correction for every car on the conveyor belt is needed.

3.2 System setup and workflow

In Figure 1 the setup of the industrial application is shown. To run the MBT, an industrial PC is used that is integrated into a carriage. On top of the carriage a robust and adjustable camera mount is installed that holds a video camera. A uEye UI-3000SE-C-HQ camera with global shutter, 12 mm lense and a resolution of 4110x3006 was used for all evaluations in this paper. The light weight robot is also attached to the carriage. To get information about the conveyor

Figure 2: Tests with light turned on, belt parallel lights turned off and light in the station turned off. Top row: Unmodified images. Bottom row: Images with automatic color and contrast correction.

belt movements a rotary encoder is attached under the conveyor belt. A light barrier detects if a car enters the station. All components are connected to a programmable logic controller (PLC) that controls the production station. The PLC on the other hand is connected to the production system and provides information about type, color and variant of the actual car to the system. With this information the correct CAD model can be loaded for the MBT.

To setup the MBT and to teach the robot, the same car is used. The car is brought into a reference position that is fully visible in the view of the camera. This position is called the virtual trigger (VT). Then the painted car is measured with the MBT and the pose is stored as P_{ref}. The increment value of the rotary encoder Inc_{VT} is also stored. During production the passing cars are measured by the MBT at the VT position. From the VT an earlier position in the opposite conveyor direction is calculated and called "first fit". The first fit position lies directly behind the light barrier and is used to start the matching of the edges of the CAD model with the edges detected in the video frames to obtain a continuous tracking for the following frames. When cars during the production pass the VT, the pose P_{VT} is measured by the MBT and the transformation between P_{ref} and P_{VT} is calculated with $P_{diff} = P_{ref}^{-1} * P_{VT}$ and send to the PLC. P_{diff} determines the shift and twist of the actual car in comparison to the reference car. This information is used further to correct the teached coordinates of the robots measurement points for every car. In this way collisions between the cars and the robot are avoided.

3.2.1 Synchronization of robot, conveyor belt and model-based tracking

By using a rotary encoder attached under the conveyor belt the covered distance can be calculated by a conversion factor. The rotary encoder delivers increment values of the belt that can be read by the PLC. If a car enters the station, the light barrier detects it and sets an increment counter to zero. From then on, the PLC sends the actual increment value Inc_{act} every 4 milliseconds (ms) to the MBT system. When Inc_{act} is greater than Inc_{VT}, the car has reached the VT and a timestamp t_{act} is stored in the MBT system. The MBT system sets a timestamp for every calculated pose. Thus, the last pose P_{t1} before timestamp t_{act} and the first pose P_{t2} after t_{act} can be used together with the time difference t_{diff} to calculate the interpolated pose P_{VT} at the virtual trigger position. With $P_{VT}/t_{diff} = factor$, the slerp function can be used to interpolate between the poses. In this way P_{VT} at time t_{act} is calculated and can be compared directly with the reference pose P_{ref}. Because the increment values are merely sent every 4 ms, an additional error of approximately 0.3 mm can occur with a conveyor speed of 75mm per seconds that was used in the experiments.

In the configuration stage of the production station the transformation of the car to the robot base and the conveyor belt direction is measured with a high precision measurement device like a Faro laser tracker [Far21] or a Creaform metra scan [Cre21]. By using these information together with the pose correction P_{diff} and the conveyor increments including the conversion factor, the robot can predict the exact car position and respective measurement points. Thus, the robot can approach each calibrated measurement point on the car correctly without any collisions.

3.3 Error sources

Several error sources have an influence on the accuracy of the total system in this complex application. A detailed evaluation regarding the single error sources can be found in [Sch20]. We sum up the most important ones under an assumption of 2 σ:

- MBT errors: 0.7 mm ± 1.5 mm
- MBT-conveyor synchronization:1.7 ± 1.5 mm
- Robot to conveyor synchronization: ± 0.5 mm

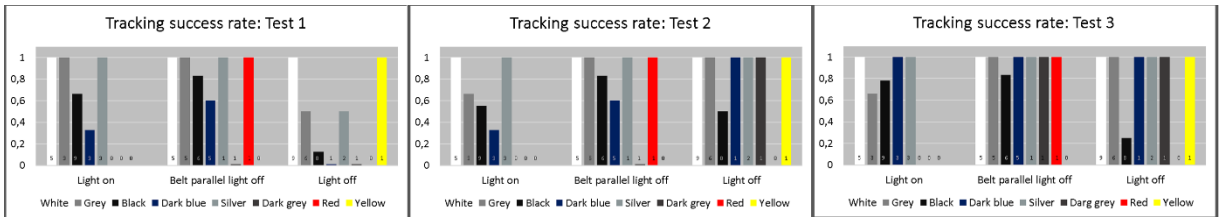


Figure 3: Results of the light tests for different gloss paints.

- Robot calibration of gap points: ± 0.5 mm
- Production tolerance of test object ± 0.5 mm
- Calibration conveyor direction
- Calibration robot to test object

In addition, several other factors have an influence on the MBT results, like the localized work illumination, the different colors of the car paint or the selection of the CAD parts that are used for MBT.In the following sections these factors are evaluated in detail. An accuracy analysis for such a complex system is difficult, since many factors have an influence on the result. In [Sch20] separate factors are evaluated. But at least the accuracy of the final system output is essential. Therefore, we also present an evaluation of the total accuracy of the system. Thereby, the laser line tool at the robot head is used to determine how precise the robot can approach the several measurement points on the outer shell of painted cars.

3.4 Influence of illumination

In [Sch20] the influence of lighting and temperature over time on MBT for body shells was evaluated. For the use case of painted cars the influence of lighting is bigger due to reflections or the influence of car paint to the edge detection and matching of the MBT. As first evaluation criterion, we used the tracking quality. This is an output of the used MBT that describes how many edges of the CAD model could be matched to the edges in the respective camera image. A factor of 0.8 for examples denotes that 80 percent of the CAD model edges could be matched.

The localized work illumination for production facilities is built up considering a norm. Mostly fluorescent tubes are used. Fluorescent tubes that are aligned parallel to the conveyor belt can have a negative influence to the MBT. The reflections of the tubes can cause the detection of an edge by the MBT that is similar to edges coming from gaps between car parts. If these edges lie in the surrounding area of a correct edge a mismatch can occur. The effect is visible in Figure 2. A light reflection lies in the direct surrounding of the gap between the car hood and the car wing. Another observed issue was that white areas, e.g. walls, in the video image can outshine image regions. This can be seen in Figure 2. The background wall outshines the contour edges of the car roof. This can lead to a bad initial match of the MBT edges. Another problem concerning a correct MBT matching process is the bad contrast between the black colored belt and the underbody for cars with dark or black gloss paint. To overcome the described problems we tested different settings in our evaluation. First, we tested three light situations: All lights turned on, belt parallel lights turned off and all lights turned off. We call this test “test 1”(see Figure 2). Then we modified the camera images with an automatic color and contrast correction. This is “test 2”. In “test 3” we modified the tracking parameters. Thresholds were modified to detect more edges in the video image and the initial tracking value was lowered. The initial tracking value relies on the tracking quality value that describes in percentage from when on a matching can take place, e.g. try a match if 50 percent of the edges match. The results are listed in Figure 3.

For this test we defined a tracking success rate. Only results with a tracking quality greater than 0.55 and a jitter smaller than 5 mm were considered as a successful tracking. The decision was based on the observation that with results smaller than 0.55 the MBT more often runs into local minima. The success rate is marked on the left scale and describes how many of the evaluated cars could be successfully

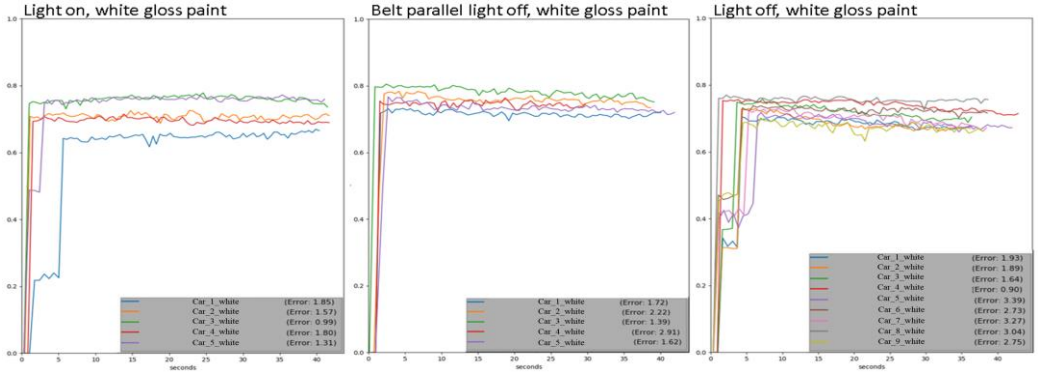


Figure 4: Tracking quality results for cars with white gloss paint over time in different lighting situations.

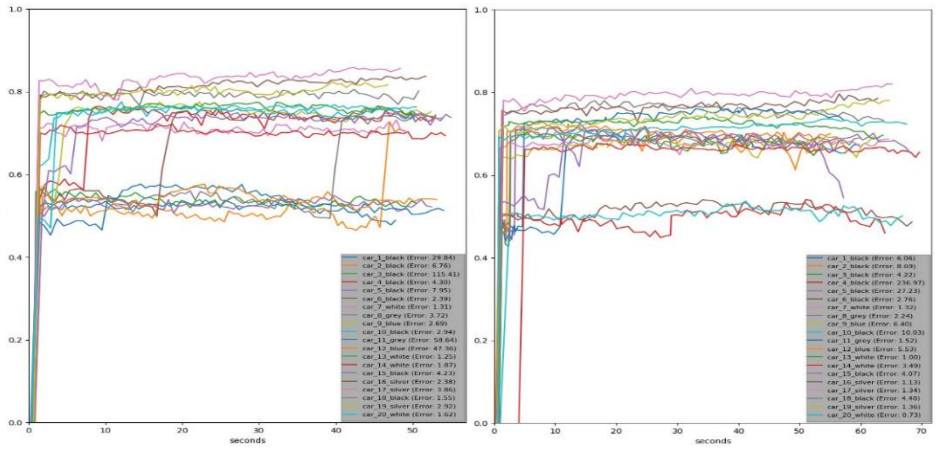


Figure 5: Results over time for different models: Left: Common parts. Right: Full model with all variants.

tracked after the criteria described above. The gloss paint of the cars is depicted in the bottom line and corresponds to the bars. The numbers on the bottom of the bars show how many cars per color were considered. This means: If 5 cars were considered per gloss paint and all are tracked successfully the bar goes up to a value of one. If only three out of five are tracked successfully the bar goes up to 0.6.

The most significant findings are that the automatic color and contrast correction improves the success rate for the situation where all lights are turned off. Otherwise, no big differences can be found between "test 1" and "test 2". The best overall result can be found when the belt parallel lights are turned off in "test 3". Further tests considered the behavior of the MBT over time. In Figure 4 the results of the tracking quality for a moving white car can be seen for a period of 35 seconds. The best results were also achieved for the lighting situation with belt parallel lights turned off. This lighting achieves the best matching rates between CAD and video image edges from the beginning of the period. Under the other light situations the MBT needed some time until the matching yields the best rates and the tracking quality is generally lower. Due to our findings we kept the light and tracking parameter settings that achieved the best results for our further evaluations.

In Figure 4 we also listed an average error. We estimated for all poses of a car during the period a best fitting line with RANSAC and computed the average deviation along this line. This can serve as a measure to check how stable the MBT calculates the car poses, because due to the conveyor belt setup it can be expected that the cars poses lie on a straight line according to the belt direction.

Throughout the evaluation we found that completely black cars are a challenge for the MBT. Especially the low contrast to the conveyor belt can have a disturbing influence on the MBT. Due to this, we conducted another evaluation with additional lights. Four spotlights were used to illuminate the gap between the belt and the underbody and another spotlight behind the car that illuminates the background to improve the contrast (see Figure 7). Furthermore, the gamma correction of the video camera was turned on to further improve the contrast. For black cars we found that the additional lighting improves the tracking quality value, meaning more edges could be matched. Approximately 10 percent more edges could be found (see Figure 7, top row). By turning on the gamma correction with no additional lighting we achieved nearly the same tracking quality values (see Figure 7, middle row). Turning the light and the gamma correction on, a tracking quality value of 0.79 was achieved (see Figure 7, middle row). Since additional

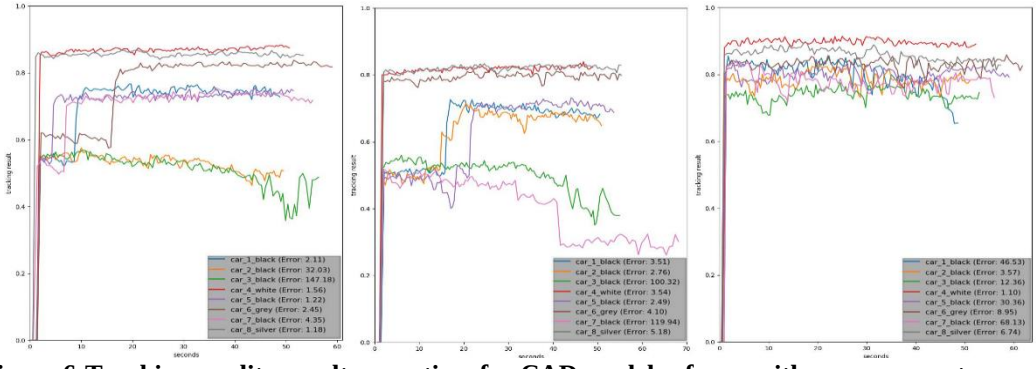


Figure 6:Tracking quality results over time for CAD models of cars with common parts over all variants (left), of a half car model (mid) and of the outer contour of common parts(right).

lights tend to outshine image regions a test for white gloss paint was needed to exclude any unwanted side effects. For such a car we achieved opposite results. With gamma correction a result of 0.84 was obtained. With lighting turned on the tracking quality dropped to 0.78 (see Figure 7, bottom row).

Figure 7: Images of the test with additional lighting and gamma correction turned on.

With outshined surfaces MBT has problems in finding edges correctly. This test was also conducted for other gloss paints. In general, we found that the gamma correction improves contrasts and reduces outshined surfaces in the images in a way that the MBT can achieve a better matching. We also found that the gamma correction has a similar effect on the tracking quality than the color and contrast correction. Therefore, we just use the gamma correction and avoided the additional color and contrast correction. Since we don't want to use special adjustments of the MBT parameters, the camera settings and the lighting situation for every possible gloss paint, the best solution to work with is one setting that fits it all. According to the results above the best solution regarding a cost-benefit analysis can be achieved with the belt parallel lights turned off, the gamma correction of the camera turned on and the usage of the optimized parameters for the MBT.

3.5 Evaluation of different CAD models

Considering all car variants produced on a production line, there is a high complexity of the used CAD models. Reducing this complexity would lead to a more manageable situation for the usage of MBT in industrial settings. Therefore, another evaluation was conducted that compares the usage of CAD models with all variant parts, of common parts over all variants only and of the outer contour of common parts only. Common parts are for example the chassis, the hood, the doors, the trunk, the roof, the windows, the front shield, the car wings, etc..

In the production every station meter costs. Reducing the station length by shortening the view space of the camera would reduce such costs. On that account, we evaluated the usage of a half car model only. This evaluation was conducted at the beginning of all of our tests. Therefore, the settings described in the previous section 3.4 were not used. This can be seen in the generally lower tracking quality values and higher jitter around the fitted RANSAC vector for all tests in this evaluation (see Figure 6).

In Figure 6 the tracking quality over time is shown for common parts that can be used for all car variants (left), a half car model (middle, the front half of the car was used) and for the outer contours (only contour edges used for the MBT) of common parts of all car variants. Considering the tracking quality value, the number of matched edges for the outer contour achieves the best results, followed by the common parts. The results for the half car model are the worst. But setting these values in correlation to the fitted average vector with RANSAC, the situation looks different. The poses of the common parts test slightly vary around the fitted average vector. For the half car model the deviations to the RANSAC vector increase and for the outer contours we found a big jitter and even big jumps in the single poses in relation to the straight fitted vector of poses with RANSAC. The tests also contained outliers as can be seen on the deviations in form of errors to the RANSAC vector. The tests were conducted before the illumination tests in section 3.4. By using the results of section 3.4, we observed that such outliers are drastically reduced. Finally, the common parts achieved the best results in terms of stability and robustness, even though a smaller amount of edges can be matched.

In another test, common parts are compared to the CAD model that contains the full car variants CAD data, for example with special parts for a car line like the sill, driving mirror, radiator grill etc.. In Figure 5 the results are shown. The results are ambivalent. For some cars the tracking quality with the common parts are better and for other cars the full CAD model gets better results. A handful of cars benefit from the full CAD model, but don't represent the majority of cars. With the common parts slightly more outliers are present considering RANSAC. Since the results aren't that clear and overall both method generate stable and robust average results, we decided to continue with the method of using common parts. This is because the model preparation, including part search in the database, data export and converting the data into the necessary formats takes time. Under a cost-benefit analysis, using the common parts makes sense in

	Quantity	P1 X	P1 Y	P2 X	P2 Y	P3 X	P3 Y	P4 Y	P4 Z	P5 Y	P5 Z
Black	27	0,99 ± 4,21	-1,1 ± 3,79	1,76 ± 4,32	-3,13 ± 3,52	-1,29 ± 6,78	-1,78 ± 4,05	-1,88 ± 3,79	0,52 ± 6,41	0,33 ± 4,37	1,44 ± 3,68
White	68	-0,82 ± 5,06	0,7 ± 2,91	-1,2 ± 3,90	-1,54 ± 2,81	-3,24 ± 5,21	-0,41 ± 2,91	0,71 ± 2,97	-1,02 ± 6,12	0,74 ± 2,62	0,41 ± 3,04
Obsidian black	55	0,12 ± 4,14	-0,77 ± 4,16	0,91 ± 4,42	-2,91 ± 3,28	-1,99 ± 4,89	-1,42 ± 3,54	-1,7 ± 4,24	1,4 ± 6,14	0,59 ± 4,00	1,77 ± 3,10
Diamant white	5	-1,12 ± 4,43	0,25 ± 1,01	-1,69 ± 4,41	-1,92 ± 1,76	-3,33 ± 5,13	-1,19 ± 2,40	0,51 ± 2,34	0,21 ± 5,39	0,12 ± 2,25	0,66 ± 2,92
Graphite grey	12	-0,91 ± 5,57	-0,03 ± 2,66	-0,36 ± 4,43	-2,53 ± 2,35	-2,86 ± 7,08	-1 ± 2,49	-1,31 ± 3,44	1,65 ± 6,89	0,78 ± 2,39	2,27 ± 3,82
Covansit blue	13	-0,1 ± 5,10	-0,98 ± 5,41	0,3 ± 4,89	-2,29 ± 4,58	-1,68 ± 7,86	-0,6 ± 5,07	-2,14 ± 6,52	2,39 ± 6,36	1,41 ± 5,25	1,06 ± 3,44
Hightech silver	37	-2,62 ± 4,31	1,08 ± 2,62	-1,2 ± 2,40	-1,49 ± 2,26	-3,64 ± 4,00	-0,23 ± 2,43	0,04 ± 3,32	0,2 ± 7,03	1,3 ± 2,62	0,87 ± 2,79
Selenit grey	40	-2,82 ± 5,00	0,63 ± 3,22	-1,13 ± 3,62	-1,95 ± 3,01	-4,55 ± 5,24	-0,29 ± 3,50	-0,56 ± 3,07	0,37 ± 6,17	1,28 ± 3,80	1,96 ± 2,67

Table 1: Results for the total system accuracy for different gloss paints.

regard to an industrial application. Furthermore, this approach is safeguarded by the evaluation in the next section, where the accuracy of the total system is evaluated. But if an application or approach has problems reaching given accuracy requirements, our findings confirm that it may be better to use the full CAD model for MBT of painted cars.

3.6 Evaluation of the system accuracy

The determination of the accuracy of a complex system can be challenging. Correlations exist between single error sources and further influencing factors, like lighting, CAD model selection or the gloss paint. In section 3.3 the single error sources are described that were evaluated in [Sch20]. For painted cars further factors have to be considered, whose influence was evaluated in the previous sections. With these evaluations researchers and the industry can better understand the relationship and possible effects of single components, influencing factors and their complex interaction. But in the end the total error of a complex system is essential to assess the accuracy, robustness or applicability.

Therefore we use the same method to determine the total accuracy of the system as described in [Sch20]. The laser line tool (LLT) that is mounted on the robot head to measure gaps is also used to measure the accuracy of how exactly the robot approaches a measurement point. The LLT consists of two laser line scanners. They are arranged in a special mount and specifically aligned to create one large laser scan line. In Figure 1 the LLT is shown. If the robot measures a gap point at a car the deviation of the gap midpoint to the LLT midpoint can be measured (the output is a coordinate on the laser line and the distance to the surface). This error is given in 2D LLT coordinates and serves as a metric to assess the total accuracy of the system. With this evaluation criterion it can at least be determined how precise the robot can reach a single measurement point and gives us an assessment of the operational fitness of the MBT for this kind of application scenario.

The measurement points are taught to the robot in the initial position. The LLT is aligned perpendicular to the normal of each gap and precisely aligned with a special template (see Figure 1). Little bridges on the backside guarantee a tight fit into a gap. Then the large laser line is orthogonally aligned to the template by an expert and the robot's position is stored. To overcome possible errors in this manual process, a software offset based on the measurements is calculated. It corrects systematic deviations and ensures a tight orthogonal alignment of the LLT's laser line to a gap. With the car in the reference position for the MBT, the direction of the conveyor belt and the transformation between the robot base and the painted car is measured with a high precision measurement device, like a Faro laser tracker. All these values together with the conveyor belt increments for the reference position are used by the robot in the conveyor belt synchronous mode to predict the position of a car for the actual belt increment values. The MBT measures the actual car at the VT (see section 3.2) and calculates a pose correction in comparison to the reference car. With this pose correction and the predicted car position the calibrated measurement points can be corrected in robot coordinates for the considered car.

To evaluate if the accuracy is suitable to avoid collisions of the robot with the cars and to fulfill the accuracy requirements, an evaluation for five measurement points on 261 cars was conducted. Measurement point P1 is at the middle of the gap between the car front door and the car wing. P2 is on the middle of the gap between the back and the front door. P3 is at the middle of the gap between the back door and the rearward car wing. P4 is at the middle of the gap between the hood and the car wing. And the last measurement point P5 is at the middle of the horizontal bottom gap of the tank cap. The results are shown in Table 1. As explanation: The origin of the car coordinate system is at the center of the front suspension, whereby the X-axis points along the cars main axis to the back wheels, the Y-axis points to the right front wheel and the Z-axis points up. Since the LLT has to be aligned orthogonally to a gap to perform measurements and delivers 2D results, vertical gaps are measured with X and Y coordinates, according to the cars coordinate system, and horizontal gaps are measured with Y and Z coordinates. We evaluated the mean value and the standard deviation of the difference between the measured gap and the measured LLT midpoint. In sum 261 cars were considered. In Table 1 only 257 are shown, because the additional cars had four separate colors, whereby no mean value and standard deviation could be calculated. In Table 1 the results are listed under an assumption of a Gauss distribution with 2 σ. Thus, approximately 95% of all measurements should lie in

	Cars clustered by brightness					
	Bright	White	Bright without white	Dark	Black	Dark without black
Car count	112	73	39	149	82	67
P1 X	5,04	5,02	4,27	5,41	4,23	5,66
P1 Y	2,80	2,82	2,70	4,11	4,05	3,87
P2 X	3,48	3,94	2,39	4,72	4,45	4,19
P2 Y	2,60	2,75	2,28	3,45	3,37	3,30
P3 X	4,81	5,20	3,90	6,36	5,62	6,67
P3 Y	2,75	2,90	2,42	3,86	3,73	3,70
P4 Y	3,16	2,93	3,34	4,22	4,10	4,22
P4 Z	6,49	6,10	6,96	6,40	6,28	6,53
P5 Y	2,65	2,61	2,55	4,09	4,12	3,92
P5 Z	2,99	3,03	2,84	3,25	3,31	3,17

Table 2: Results for the total system accuracy grouped by brightness.

the range μ ±2 σ. Generally, the standard deviation in the X direction are higher, because of the errors from the conveyor belt synchronization with the robot and with the MBT. Another finding is that for the Z coordinate of P4 the deviation for all considered gloss paints is constantly higher. This is partially caused by the error in the z coordinate for the point itself and in addition by the synchronization to the conveyor belt in x direction. Since P4 is on the curved hood, errors in the synchronization have an influence on the position where the gap is measured. This results in higher deviations of the Z coordinate. But nevertheless, the mean values are mostly in the range of up to two millimeters. Further findings are, that darker colors have a slightly higher standard deviation. This is plausible, because darker colors are more challenging for the MBT. From Table 1 it may be not that clear, but if the results for different colors are grouped by brightness, the difference gets much clearer. This can be seen in Table 2. The results represent the values for two times the standard deviation (2 σ). 261 car were considered. Referring to the colors in Table 1, black, obsidian black, graphite grey, covansit blue and selenit grey were assigned to the cluster "dark". White, diamant white and hightech silver were assigned to the cluster "bright". In sum 12 different gloss paints were considered. By separating colors like pure white or pure black, that are more challenging to the MBT (white gloss paint due to the outshining), and showing clusters without these cases, other findings are that for the bright colors without white the deviation slightly decreases; but for dark colors without black in the opposite case a slight tendency is recognizable that black may not be the most challenging color. Other dark colors seem to have a slightly more increasing influence on the standard deviation.

Considering the mean values, the total system achieves results, mostly varying in the range of up to two mm. In comparison of all values the point P3 achieves the worst results over all colors for the X coordinate. Since this point is one of the last measured points, rotational errors in the system could increase the results. The results for the tank cap with P5 however foil this assumption. P5 is measured as the last point, but neither shows very high results for the mean value, nor the standard deviation. Thus, we don't expect the MBT or the measured belt direction and transformation between the car and the robot as the reason for this effect. The most likely cause seems to be that during the manual calibration of the point by the robot, an error was introduced.

Overall the total system achieves results for the five measurement points and for all gloss paints that are suitable for automatic gap measurements. For 80 coordinate values listed in Table 1, 71 were under an total error of 8 mm, under an assumption of a Gaussian distribution with μ ±2 σ. Six values lied under 9 mm and three under 10 mm. From these nine values above 8 mm, six are related to the X coordinate of point P3. The possible reasons for this behavior were described above. Finally, the laser scan line has a total length of several centimeters. Under these circumstances, the evaluation showed that even with the worst case accuracy of up to 10mm in some cases, gaps can be measured by the LLT without any difficulty.

4. Evaluation of a pragmatic camera maintenance concept

The setup of the system requires the measurement and teaching of a reference car. In case of a camera defect, this procedure has to be repeated in a production free time slot. It is also necessary for a camera position change. Normally, the occurrence of the second case is rather unlikely due to the robust and tight camera mount, but if an object or person crashes with the camera mount, it cannot be completely excluded that the position of the camera slightly changes. To overcome these limitations we present a method to compensate such errors without the need of a new initial measurement. Therefore, we use a special test specimen that is shown in Figure 8. During the car measurement in the reference position the test specimen is setup and also measured with the MBT. If no camera position change occurred, the measurement of the test specimen by the MBT can also be conducted after the initial measurements. With these measurements, the transformation between the reference car and the test specimen can be calculated, as shown in Figure 8 (left). If the camera is exchanged due to a defect or if the camera is displaced, the specimen is set up again and measured with the MBT. By using the transformation T_{Spec}^{Cam2}, the former calculated transformation between the car and the specimen T_{Spec}^{RefCar} and the pose of the actual car during production $T_{Current}^{Cam2}$, the transformation between the actual car and the car in the reference position out of

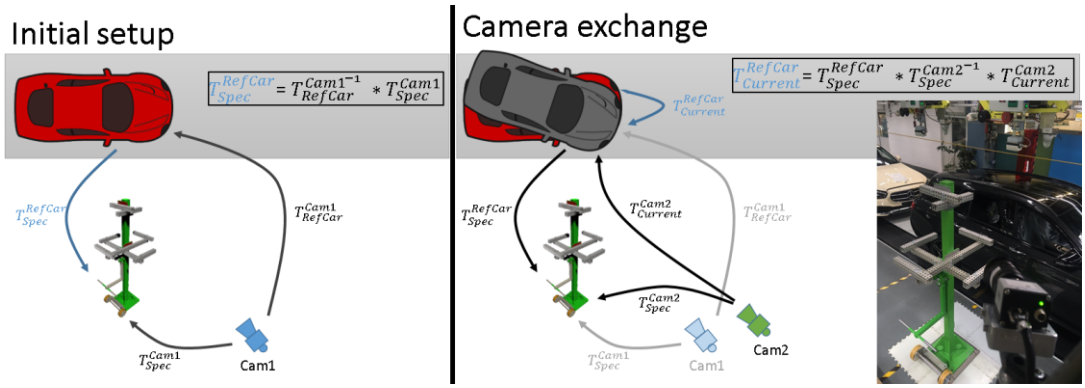


Figure 8: Transformations for the usage of an additional test specimen for the pose correction calculation.

sight of the new or modified camera can be calculated. This transformation can be further used as the pose correction for the robot (see Figure 8, right).

4.1 Evaluation camera exchange

The approach to handle a camera exchange or camera displacement in a pragmatic way without the need for time consuming initial measurements was evaluated in several tests. The CAD model of the specimen was created by a 3D scan with a high precision GOM 3D scanner [GOM21].

4.1.1 Repeatability of the installation

	X	Y	Z	RotX	RotY	RotZ
μ	1071,58	-998,44	2202,91	44,03	55,82	61,35
σ	0,10	0,14	0,14	0,005	0,007	0,005
Max diff to μ	0,21	0,29	0,26	0,011	0,017	0,010

Table 3: Results for repeated setup of the specimen.

The specimen is mounted on the ground of the factory. Therefore, a pinned fitting was used to overcome problems that may arise by screw tightening. In the test the specimen was setup and dismantled ten times and measured with the MBT. We calculated mean, standard deviation and maximum difference to the mean value. Results are shown in Table 3 and show that with this installation method the standard deviation is very low and the biggest deviations are in the range of 0.2 to 0.3 mm for the translation and about 0.01° for the rotation. Thus, the setup and mounting of the test specimen is not a big error source.

4.1.2 Verification of new pose correction

	X	Y	Z	RotX	RotY	RotZ
Mean difference	0,030	-0,138	0,346	-0,004	-0,012	-0,006

Table 4: Mean of pose correction differences.

To compare the new pose correction calculation with the test specimen with the old pose correction calculation, another test was conducted that investigates the influence of the further measurement of the specimen with the MBT. First the specimen was measured with the MBT. Then, the pose correction for 27 cars on the belt was calculated with both methods under the usage of the same camera. Then the difference between the pose correction transformations was calculated. In Table 4 the mean of these differences is shown. The results only differ in the first decimal place for the translation and in the second decimal place for the rotation values. Thus, in a direct comparison the pose correction with the help of the test specimen introduces a very small error.

4.1.3 Camera exchange and position change

In the next step, three tests with different uEye UI-3000SE-C-HQ cameras were conducted to test the camera exchange. First, camera 3 (cam3) was used and 30 cars were measured in the same way described in section 3.6. These measurements were used as a reference to compare the further test results. Instead of five measurements points we neglected the tank cap and used four points only. These were the same as in the evaluation in section 3.6. In test 1.1 cam3 was exchange with cam1 and then 20 cars were measured. Then cam1 was exchanged with cam2 and 22 cars were measured. Afterwards cam2 was exchanged with cam3 again and 23 cars were measured. In test 2.1 cam1 was used again and we simulated a crash of an object or person with the camera mount by rotating the camera a bit within the ball joint of the mount. 20 cars were measured subsequently. Then, cam1 was exchanged by cam2 and 20 cars were measured. After this, cam2 was exchanged by cam3 and 20 cars were measured. For test 3.1 cam3 was used and the camera position was modified somewhat stronger. Therefore, the cameras were rotated with the ball joint and also translated by moving the horizontal and vertical elements of the camera mount for some centimeters. With this setting 19 cars were measured. In test 3.2 the procedure was repeated and the camera position was further changed. Then, 12 cars were measured. The test results are shown in Table 5. We calculated the mean value and then the difference to the mean value of the reference test (see above). Therefore, the results in Table 5 are given as μdiff ±2 σ. The results show the difference of the mean of the new method with test specimen in comparison to the calculation method from section 3.6 and two times the standard deviation. For the mean difference, the majority of the results lie in the range of up to 1 mm. Only a few are in the range of 2 to 4 mm. The results for two times the standard deviation are comparable to the results of section 3.6 with a very slight tendency of increasing values. Since the laser line of the LLT is several centimeters long,

	Test 1.1 (Cam1)	Test 1.2 (Cam2)	Test 1.3 (Cam3)	Test 2.1 (Cam1)	Test 2.2 (Cam2)	Test 2.3 (Cam3)	Test 3.1 (Cam3)	Test 3.2 (Cam3)
Status	Cam change	Cam change	Cam change	Pos change	Cam change	Cam change	Pos change	Pos change
Cars	20	22	23	20	20	20	19	12
P1 X	2.86 ± 7.14	2.14 ± 6.61	2.23 ± 5.03	0.57 ± 6.56	2.44 ± 4.01	1.87 ± 5.56	0.04 ± 7.89	0.16 ± 6.30
P1 Y	1.47 ± 3.07	0.84 ± 3.27	0.49 ± 2.66	1.09 ± 2.52	1.67 ± 3.20	0.68 ± 3.21	1.15 ± 2.31	0.31 ± 3.21
P2 X	1.62 ± 4.57	3.15 ± 6.85	0.93 ± 4.17	0.07 ± 4.50	1.37 ± 4.13	1.28 ± 4.87	0.37 ± 4.87	0.32 ± 3.96
P2 Y	3.08 ± 3.04	1.26 ± 3.43	0.47 ± 2.85	1.34 ± 2.40	0.63 ± 3.27	0.79 ± 2.93	1.29 ± 3.19	0.51 ± 1.78
P3 X	2.82 ± 6.79	0.62 ± 6.42	1.18 ± 4.37	0.12 ± 6.02	0.33 ± 4.39	2.38 ± 4.72	0.65 ± 7.21	0.61 ± 5.74
P3 Y	3.95 ± 2.91	1.97 ± 3.78	0.90 ± 2.94	1.17 ± 2.45	0.37 ± 2.98	0.48 ± 2.88	1.17 ± 3.54	0.98 ± 2.57
P4 Y	0.07 ± 3.13	0.21 ± 3.79	0.26 ± 2.98	0.76 ± 3.14	1.61 ± 4.26	0.73 ± 3.31	0.96 ± 3.42	0.14 ± 3.44
P4 Z	2.74 ± 2.78	0.15 ± 2.74	0.33 ± 2.64	0.42 ± 2.66	0.55 ± 3.53	1.08 ± 3.30	0.04 ± 2.30	0.53 ± 2.28

Table 5: Results for camera exchange and displacement tests.

the results show that the MBT and this concept to handle maintenance cases of the camera is suitable for the kind of automatic gap measurement application described in this paper. Camera exchanges or smaller position and rotation changes of the camera can be handled by the concept of measuring a test specimen with the MBT to correct calibrated relations of the system components. With this concepts a camera exchange can be performed within minutes, instead of some hours that are needed for initial measurements.

5. CONCLUSION

In this paper we presented the evaluation of model-based tracking (MBT) for robotic gap measurements on painted cars. Different lighting situations in correlation with the car's gloss paint were evaluated and analyzed. In this context image and lighting optimizations were tested to maximize the matching quality of the MBT. In this regard, another evaluation considering the optimal CAD model selection for the MBT was presented. We showed the industrial use case of using MBT to track painted cars on a conveyor belt and to use the results to correct learned robot measurement points. With this method collisions between the robot and the cars can be avoided. The applicability of MBT for such an industrial use case was proven by an evaluation of the total accuracy of the system. Therefore, the gap measurement tool of the robot that consists of a laser line was used. It was shown that the majority of measurement points could be approached by the robot with an accuracy of up to 8mm. Furthermore, a method to exchange a calibrated camera in the system within minutes was presented. The method relies on a separate test specimen and was evaluated in detail. The results showed, that with this method an efficient camera maintenance model can be implemented, that merely introduces a small and acceptable error in regard of the total system accuracy. The presented methods and evaluations can help researchers and the industry to better assess and understand the influence and correlations of different factors on MBT in complex industrial use cases.

6. REFERENCES

[VP05] Lepetit, V., Fua, P. (2005). Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. Foundations and Trends in Computer Graphics and Vision.

[Wue07] Wuest H., Wientapper F., Stricker D. (2007) Adaptable Model-Based Tracking Using Analysis-by-Synthesis Techniques. In: Kropatsch W.G., Kampel M., Hanbury A. (eds) Computer Analysis of Images and Patterns. CAIP 2007.

[Com06] Comport, A., Marchand, E., Pressigout, M., Chaumette, F.: Real-time markerless tracking for augmented reality: the virtual visual servoing framework. IEEE Trans. on Visualization and Computer Graphics 12(4), 615–628 (2006)

[Sch20] Scheer,F., Neumann,M., Wirth,K., Ginader,M., Oezkurt,Y., Mueller,S.: Evaluation of model-based tracking and its application in a robotic production line. Journal of WSCG, 2020.

[Vac04] Vacchetti, L., Lepetit, V., Fua, P.: Combining edge and texture information for realtime accurate 3d camera tracking. In: Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR) (2004)

[Hin11] Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N.,Lepetit, V.: Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes. IEEE International Conference on Computer Vision (ICCV), 2011

[Gar18] Garon,M., Laurendeau, D., Lalonde, J-F.: A Framework for Evaluating 6-DOF Object Trackers. In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. Computer Vision – ECCV 2018.

[Wu17] Wu, P-C., Lee, Y-Y., Tseng, H-Y., Ho, H-I., Yang, M-H., Chien, S-Y: [Poster] A Benchmark Dataset for 6DoF Object Pose Tracking. 186-191. 10.1109/ISMAR-Adjunct, 2017.

[Hod17] Hodaň,T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., Zabulis, X.: T-LESS An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects, IEEE Winter Conference on Applications of Computer Vision (WACV), 2017

[Cre21] Creaform Metra Scan: https://www.creaform3d.com, 2021

[Far21] FARO Europe GmbH & Co. KG, Faro arm, Faro laser tracker: https://www.faro.com, 2020

[Gom21] GOM GmbH, ATOS, https://www.gom.com, 2020.

Relevant Independent Variables on MOBA Video Games to Train Machine Learning Algorithms

Juan Guillermo López Guzmán
Pontifical Xaverian University
110121, Bogotá, Bogotá DC
juang.lopezg@javeriana.edu.co

Cesar Julio Bustacara Medina
Pontifical Xaverian University
110121, Bogotá, Bogotá DC
cbustaca@javeriana.edu.co

ABSTRACT

Popularity of Multiplayer Online Battle Arena (MOBA) video games has grown considerably, its popularity as well as the complexity of their playability, have attracted the attention in recent years of researchers from various areas of knowledge and in particular how they have resorted to different machine learning techniques. The papers reviewed mainly look for patterns in multidimensional data sets. Furthermore, these previous researches do not present a way to select the independent variables (predictors) to train the models. For this reason, this paper proposes a list of variables based on the techniques used and the objectives of the research. It allows to provide a set of variables to find patterns applied in MOBA videogames. In order to get the mentioned list, the consulted works were grouped by the used machine learning techniques, ranging from rule-based systems to complex neural network architectures. Also, a grouping technique is applied based on the objective of each research proposed.

Keywords

Pattern Recognition, recommender systems, Machine Learning, MOBA, League of Legends.

1 INTRODUCTION

Videogames are already present in our everyday life. They have reached a massive audience from all ages due to their wide variety of genders and thematics. Those genders evolve trought time and originate new ones, for instance, Real Time Strategy games (RTS) originates at the early 2000's a different sub-gender known as Massive Online Battle Arena (MOBA) [1].

There are many games that could be considered as MOBA predecessors. Although, the first videogame formally classified as a MOBA was Defence of the Ancients (DotA). It was created in 2003 by players of the RPG videogame Warcraft III. DotA is based on the Warcraft III lore, however, it has specific gameplay characteristics. These characteristics have become intrinsic to MOBA gender:

Each match starts with two adversary teams, typically conformed of five players. Players work together as a team to achieve the ultimate victory condition which is to destroy their enemy's base whilst protecting their own [1]. Usually, both teams have main structures that are located at the opposite corners of the battlefield. The first team to destroy the adversaries' main structure wins the match [1]. Destroying other structures within the adversary team's base may provide other benefits. Defensive structures, which are usually automatic "towers", are located in place to prevent the base destruction. Each team is assisted by relatively weak computer-controlled units, called "minions", that periodically spawn in groups at both bases, marching down predefined paths (called "lanes") toward their enemy base. While their charges are counterbalanced by the adversary team's minions. Players can aid them which turns the minions into a useful army for striking the adversaries' defenses [2]. There are typically three (3) "lanes" on the battlefield that are the main ways of getting from one base to another. The lanes are known as top, middle and bottom lane, or, in gamer shorthand – "top", "mid" and "bot". Between the lanes is an uncharted area called "jungle" [3]. The "jungle" is a territory to neutral monsters that are hostile to both teams and appear in marked locations on the map known as "camps" [4]. Defeating neutral monsters brings various benefits to the players and their team, such as growth in power, buffs, or assistance in pushing the lane [5].

In 2009 and 2012 respectively born the most popular MOBA videogames: League of Legends (LoL) and

Defence of the Ancients II (DotA II). They maintain the gameplay described above: Two teams of five players have to destroy the nexus of the enemy team located in the different side of a predefined map. Fighting against defensive structures, player and non-player characters.

Both achieved great popularity in the e-sports scenario. For instance, 2019 League of Legends World Championship got an online audience greater than one hundred million through several electronic platforms as Twitch [7]. Also, it got an in-site audience near to forty thousand people on the Shanghai Olympic stadium.

Due to players face several real-time decisions with a wide spectrum of possibilities MOBAs are very complex videogames [8]. Mentioned complexity and their popularity have attracted a lot of researchers' attention from various knowledge fields as psycology, marketing, computer science and artificial intelligence. Figure 1 shows the growth in the number of reviewed papers by year as evidence of the researchers' interest since 2014.

The above complexity had allowed the application of a wide spectrum of machine learning techniques from regression models [6] to different neural networks architectures [7]. Technique's evolution will be presented in Section 2.

There are several objectives related to MOBA's research. For instance: how game dynamics differentiate one player from another [7], how the chosen avatar has impact in the odds to win a game [8]. Every found objective will be described in Section 3.

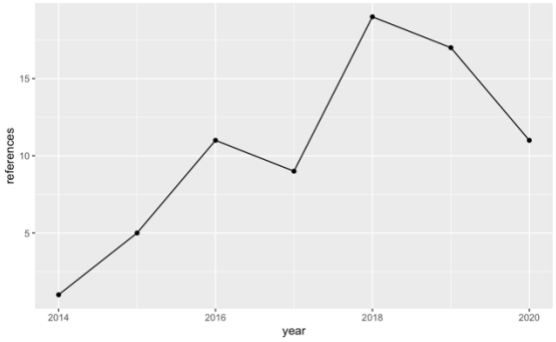


Figure 1. Number of reviewed references per year.

One important decision is how to define what items a player should buy in order to maximize his chance to win a game. As mentioned above, every decision in a MOBA videogame has a wide spectrum of feasible solutions turning them into complex optimization problem. Each player could buy until six items in each game from a pool higher than eighty options, for that reason, the number of possible builds is around 2.2e11 combinations.

According to the number of possible decisions that players can take, there are many variables to consider. In consequence the present work contains the sections describe below: Section 2 describes the most common machine learning techniques used on MOBA videogames with a special emphasys on what independent variables have been used in each technique. Section 3 groups the consulted researches by his objective and what independent variables have been used in each one. Section 4 presents a summary of the results and introduce the proposed list of relevant variables. Finally, Section 5 has main conclussiones and describes future works.

2 MOBA'S MACHINE LEARNING TECHNIQUES LISTING

An exhaustive research was performed in several academical repositories to collect researches related to MOBA videogames. Researches where investigators have used a machine learning technique was reviewed in detail to identify what techniques they were using and, the question investigators were trying to answer within the research.

Every reviewed paper uses one or more techniques to solve its objective. These techniques were tabulated and clustered into categories. It helped to find the most popular technique categories: Heuristics, neural networks, regression models and trees-based models. Figure 2 shows their evolution through time.

Most popular technique categories will be described in further sub-sections. As mentioned before, for each category will be listed the independent variables used by. Finally, there will be a sub-section to describe the methods and independent variables used in other categories.

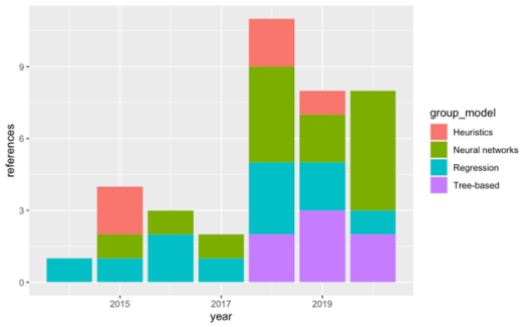


Figure 2. Number of reviewed references grouped per year and technique category.

2.1 Heuristics

During early years, heuristics were the predominant models which mainly use rule-based systems. For instance, a decisions' tree that will describe a rational agent who works like a novice players tutor [9] which

provide a rational agent the actions to execute whether the human player are dead, absent or under attack.

This category also contains: To develop an Artificial Intelligence MOBA player [10], adjusting the difficulty of a rational agent based on player performance [11], to suggest the avatar to choose in order to maximize the odds to win a game [8] [12] and which items should be bought during a game [13].

Table 1 presents the independent variables used as predictors in heuristic models and their number of appearances in different papers.

variable	type	number of appearances
Attacked by tower	boolean	1
Champion range	float	1
Creep missing percentage of health points	float	1
Distance between champions and creep	float	1
Distance between champions and tower	float	1
In turret	boolean	1
Items	vector	1
Partner AFK	boolean	1
Partner alive	boolean	1
Partner close	boolean	1
Partner dead	boolean	1
Partner recalling	boolean	1
Safe in ally turret	boolean	1
Teammate around	boolean	1
Tower range	float	1

Table 1. Independent variables used in Heuristics Models.

2.2 Regression Models

Another relevant category during early years is regression models. They were applied mainly to predict which team will win a game. Then, as a classification problem the more used method has been logistic regression [6] [14] [15] [16] [17]. Regression models have also been used to explain how the way players and teams move influences the game result [18] or how game result is influenced by bought items [19]. Finally, researchers have used generalized linear models to model the performance on professional teams during the 2018 League of Legends world championship [20].

Regression models have used 65 different independent variables. Those that were used in more than one model were shown in Table 2.

2.3 Tree-Based Models

A recently popular category has been tree-based models as simple classification trees [21] and behavior trees [22]. However, most used methodology has been random forest:

- to predict the learning ratio for a new player [16].
- to determine whether a skill or a usable item is available to be used [23].
- to predict the game outcome [24] [25] and
- to suggest the avatar to be used into a game [17] [26].

They have used 38 independent variables. Table 3 shows the variables that have been used into more than one model.

variable	type	number of appearances
Chosen hero	vector	4
Gold	integer	4
Creeps	integer	3
Assists	integer	2
Damage to heroes	integer	2
Deaths	integer	2
Gold difference between teams	integer	2
Items	vector	2
KDA Ratio	float	2
Kills	integer	2
Roles	vector	2

Table 2. Independent variables used in Regression Models.

variable	type	number of appearances
Chosen hero	vector	3
Kills	integer	3
Bans	vector	2
Creeps	integer	2
Gold difference between teams	integer	2
Roles	vector	2

Table 3. Independent variables used in Based-Tree Models.

2.4 Neural Networks

Recently, neural networks had become the predominant techniques during the last years. The variety of architectures in this group is remarkable. There could be found simple artificial neural networks [7] [27] [28] [19] [29], or more complex architectures: recurrent neural networks [30] [31] [32], convolutional neural networks [23], deep neural networks [33] [34] [35], fully-connected neural networks [26] [35], discriminative neural networks [36] and transformers [37].

Into the reviewed papers which used a neural network model were identified 42 independent variables. Table 4 presents those used into more than one paper.

variable	type	number of appearances
Chosen hero	vector	4
Items	vector	3
Gold	integer	2
Gold difference between teams	integer	2
Kills difference between teams	integer	2
Tower kills difference between teams	integer	2

Table 4. Independent variables used in Neural Network Models.

2.5 Other Techniques

In addition to the most popular categories, there are reviewed papers focused on to develop visualization methods to improve the comprehension about game patterns. They could be used by players [38] [39], coaches [40] [41] [42] [43] and spectators [44].

Also, classification models based on linear hyperplanes where support vector machines are the most popular [6] [45] [23] [17] accompanied by factorial [15].

Other categories are: Boosting [15] [46] [24] [17], clusterization [18] [47] [13] [29], Bayesians [6] [30] [15], behavioral [7] [48] [49] [50], bio-inspired [45] [51] [52], network analysis [53] [43] [54], mixed models [55] [56], recommender system [57] [27], fuzzy logic [58] [59], singular value decomposition [60], auto-regressive [61] and sequence analysis [50].

Summarizing the reviewed researches it is possible to notice: a) the intrinsic complexity of MOBA videogames let researchers to use a wide spectrum of machine learning techniques and b) there are variables which are repeated transversally in several researches, even when the used tecniques are evolve through time there are variables like the chosen hero present in three of the four tables shown.

3 Objective Categories

As previous section clustered the reviewed papers into categories based on the applied machine learning techniques, this section will tabulate and cluster researches based in the paper objectives. Figure 3 describes ths most popular categories.

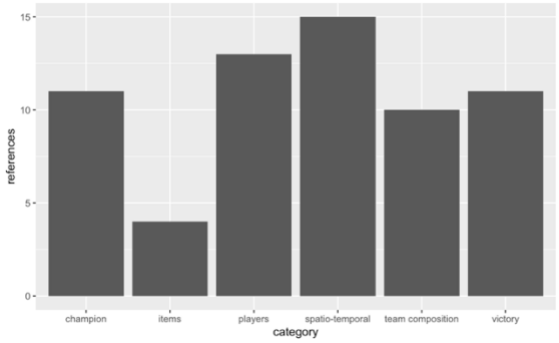


Figure 3. Number of researches per category.

Each category will be explored in detail during the further sub-sections. Besides, aligned with the main goal for each category will be analyzed the used independent variables.

3.1 Spatio-Temporal

Most popular category was called spatio-temporal. It refers to researches that have sought patterns related to how the game is developed in different map coordinates or different time slices:

- To detect an experienced player based in the way that he navigates across the game map [18] [39] [41] [43].
- To develop rational agents which determine the best way to navigate the game map at a specific moment of the game [9] [10] [35].
- To analyze the advantage of one of the teams at a specific time during the game [28] and specifically when this advantage generates an irreversible snowball effect [38].
- To improve spectators experience by determining in which sectors of the game map the events most relevant to the game are taking place [46] [55].
- To predict the outcome of an encounter with two or more players [14] [61] [34].
- Finally, to determine whether an ability or an actionable item could be used [23].

Although, it was possible to identify 64 different independent variables that were used in the aforementioned research. Due to the variety of topics, only 3 were used in more than one work, these are presented in Table 5.

variable	type	number of appearances
Gold difference between teams	integer	3
Experience difference between teams	integer	2
Positions of avatars	tuples	2

Table 5. Independent variables used in Spatio-Temporal Research.

3.2 Players

The second category was called players. Players' category groups the research that analyze characteristics of the players. These characteristics do not depend on the game condition, on the contrary, they tend to be present into every player's game. In this category we found objectives such as:

- To adjust automatically the level of difficulty for any player based on his historical performance [58].
- To predict the outcome of a game based on the historical performance of the players [62] [31] [50]
- To categorizing players according to their historical performance and design analysis strategies to improve it [40] [41] [42] [43] [29].
- To analyze the learning speed of a new player [16].
- To predict based on their gaming habits if a player is at risk of quitting the game [48], the best way to match players based on their

historical performance [33] [63] and,
- how the player's social network within the video game affects the outcome of his games [53].

For this category, a total of 51 different independent variables were identified. Table 6 shows those used in more than one research.

variable	type	number of appearances
Gold	integer	5
Creeps	integer	3
Assits	integer	2
Deaths	integer	2
Items	vector	2
KDA Ratio	float	2
Kills	integer	2
Roles	vector	2
Start time	timestamp	2
Winning	boolean	2

Table 6. Independent variables used in Players Research.

3.3 Champions

The researches that study the impact of the avatar selected by the players have been grouped into the champions category. Due to it is the way in which avatars are known within the MOBA League of Legends. The main objective in these researches has been to design champion recommendation systems [30] [27] [8] [12] [51] [17] [64] [60] [36] [26] [32] and predict the outcome of a game [15] [45] [7].

Although these works use a total of 18 different independent variables, the only one that appears in all researches is 'chosen-hero': a vector that represents the champion selected by one or more players.

3.4 Victory

Another of the objectives found in selected researches is the prediction of the winning team in a game. This category uses variables that are known at the beginning of the game [15] [27] [60] [43] [31] [65] [24] [25], at any time during the game [14] [62] or variables known at the end of a game in order to understand their influence on the result [66] [54] [20]. Therefore, this category will be called victory.

Of the 49 independent variables used within this category, table 7 shows those that were used in more than one research.

3.5 Team Composition

A limitation commonly mentioned in research is how to introduce into the models the synergies that must exist between players of the same team. Related to this question the category of team composition was created for those papers that seek objectives related to the roles of each player within a team [6] [50], the analysis of player behavior in the composition of a team [7] [45] [67] [42] [68], predicting the outcome of a game considering synergies between players [15] [65] or optimizing the construction of a team of players [8] [32] [49] [52] [63].

variable	type	number of appearances
Chosen hero	vector	5
Gold	integer	3
Gold difference between teams	integer	3
Creeps	integer	2
Creeps per minute	float	2
Gold per minute	float	2
Kills	integer	2
Level	integer	2
Roles	vector	2

Table 7. Independent variables used in Victory Research.

A total of 45 different independent variables were identified in the works within this category. Table 8 presents those used in more than one research.

variable	type	number of appareances
Assists	integer	2
Chosen hero	vector	2
Creeps	integer	2
Kills	integer	2

Table 8. Independent variables used inTeam Composition Research.

3.6 Items

One of the least explored dimensions in the game is the purchase of items to equip each player's virtual avatar. In League of Legends, each player has the possibility of acquiring up to six objects throughout a game from a pool of more than 80 different objects, all with different benefits for the character. The researches that study this game characteristic were grouped within the category objects. These seek to develop systems that recommend to the player which are the objects to acquire during the game [13] [69] [19] [35].

In a similar way as in the champions category, the only variable that appears more than once in the item's category is 'items': a binary vector that identifies whether or not a champion has a specific item equipped.

It is important to mention that none of the researches in the item's category considers the player's resources when is generated a recommendation. It could be obsolete if player does not have enough gold to acquire

the suggested item.

4 Analysis of Results

In total, 165 different independent variables were identified. This number is a consequence of the variety of objectives identified in the reviewed researches, therefore, many of these variables are used only in one paper or one objective category.

The variables aggrupation presented could be useful for future researches, specifically, authors future work will be focus on: how to define the items that maximize the chance to win a game. In consequence, results related with the categories players, champions and items will be highlighted. These categories were selected based on the qualitative hypothesis described below:

i. Players: These researches describe characteristics and play styles that are present in every player's games. It indicates that there may be a basket of items a player acquires in every game.
ii. Champions: Each virtual avatar has different characteristics (speed, armor, attack speed, ability power, among others) just like each object, therefore, each champion can have a greater affinity with those objects that share similar characteristics.
iii. Items: The papers within this category share the objective of future work.

The set of independent variables identified (165) were mapped into the three chosen categories, as presented in Figure 4 only three variables are repeated in more than one category.

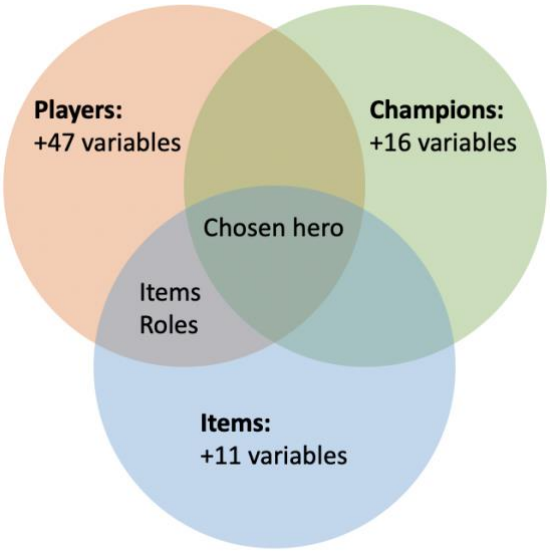


Figure 4. Independent variables that are repeated throughout the categories of interests.

'chosen_hero' variable is a binary vector that represents whether a player or a team has selected a specific champion. 'items' variable is also a binary vector that indicates whether a player or a team has equipped their champion with a specific object. 'roles' is a categorical variable that indicates the role of each player in the game.

As evidenced in [70], in order to keep the champions balanced, the developer company makes adjustments to champion and item attributes, it happens in the same way with the objects, for that reason, the relations and synergies in 'chosen_hero' and 'items' variables are constantly changin. Therefore, using these vectors of binary variables as predictors could quickly become obsolete after an update. However, both variables can be expressed based on the attributes of champions and items, so these predictors will not be linked to a specific champion / item with specific characteristics but to any champion / item whose attributes are similar to the vector of attributes trained by the model.

5 Conclusions

Thanks to the exhaustive papers review, built categories and variables match, presented in previous sections, it is possible to conclude:

i. Reviewed papers do not show a consistency in the independent variables use, even in researches with related objectives. It suggests variables were mainly chosen by the authors preferences and it is not related to previous researches or a specific methodology.
ii. It is possible to determine the chosen hero, the items and roles are the most commonly used variables and, for that reason, they could be considered as a baseline variables to machine learning applications in MOBA videogames.

Author's future work will be focused on the importance of bought items in the victory probability during a match. Below the acquire lessons that will be useful:

i. Variables mentioned (chosen hero, items and roles) will be used as baseline predictors to train machine learning models to determine the impact of items in the probability to win a match.
ii. Due to the periodic changes in the champion and items stats [71] future works will suggest representing the independent variables (chosen hero and items) in a way agnostic to the current game patch. So far, they were represented as binary vectors, in consequence, the stats updates made trained models less relevant with every patch.

iii. None of the researches reviewed consider into their models the player available resources, it means, suggestions made are optimal ones considering a player can buy any item he wants in any time during the game. Then, considering available resources when suggesting an item will be an exclusive contribution of future works.

REFERENCES

[1] L. Jackson, "How MOBAs Took Over Gaming," IGN Middle East, 1 August 2013. [Online]. Available: https://me.ign.com/en/pc/70142/feature/how-mobas-took-over-gaming. [Accessed 23 March 2021].

[2] Field Level Media, "Esports primer: League of Legends," Reuters, 24 March 2020. [Online]. Available: https://www.reuters.com/article/esports-lol-primer-idUSFLM1tj2jd. [Accessed 23 March 2021].

[3] J. Calixto, "Proving Grounds: The Geography of the MOBA Map," The Meta, [Online]. Available: https://killscreen.com/themeta/proving-grounds-geography-moba-map/. [Accessed 23 March 2021].

[4] A. Gies, "The Normal Person's Guide to Watching Competitive Dota 2 (2017 Edition)," Polygon, 2 August 2017. [Online]. Available: https://www.polygon.com/2017/8/2/16073588/the-normal-persons-guide-to-watching-competitive-dota-2-2017-edition. [Accessed 23 March 2021].

[5] Garamor, "Heroes of the Storm: How to Fully Utilize Mercenary Camps," Dignitas, 27 October 2017. [Online]. Available: http://team-dignitas.net/articles/blogs/Heroes-Of-The-Storm/11768/heroes-of-the-storm-how-to-fully-utilize-mercenary-camps. [Accessed 23 March 2021].

[6] C. Eggert, M. Herrlich, J. Smeddinck and R. Malaka, "Classification of player roles in the team-based multi-player game dota 2," in *International Conference on Entertainment Computing*, 2015.

[7] H. Wang, H.-T. Yang and C.-T. Sun, " Thinking style and team competition game performance and enjoyment," *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 7, no. 3, pp. 243-254, 2015.

[8] Z. Chen, T.-H. D. Nguyen, Y. Xu, C. Amato, S. Cooper, Y. Sun and M. S. El-Nasr, "The art of drafting: a team-oriented hero recommendation system for multiplayer online battle arena games," in *12th ACM Conference on Recommender Systems*, 2018.

[9] V. Do Nascimento Silva and L. Chaimowicz, "A tutor agent for moba games," in *XIV Brazilian Symposium on Computer Games and Digital Entertainment*, 2015.

[10] V. Do Nascimento Silva and L. Chaimowicz, "On the development of intelligent agents for moba games," in *14th Brazilian Symposium on Computer Games and Digital Entertainment*, 2015.

[11] M. P. Silva, V. Do Nascimento Silva and L. Chaimowicz, "Dynamic difficulty adjustment on MOBA games," *Entertainment Computing,* vol. 18, pp. 103-123, 2017.

[12] M. Z. A. Z. Aznin, N. M. Diah and N. A. S. Abdullah, "Expert system for dota 2 character selection using rule-based technique," in *International Visual Informatics Conference*, 2019.

[13] W. Looi, M. Dhaliwal, R. Alhajj and J. Rokne, "Recommender system for items in dota 2," *IEEE Transactions on Games,* vol. 11, no. 4, pp. 396-404, 2018.

[14] M. Schubert, A. Drachen and T. Mahlmann, "Esports analytics through encounter detection," in *MIT Sloan Sports Analytics Conference*, 2016.

[15] A. Semenov, P. Romov, S. Korolev, D. Yashkov and K. Neklyudov, "Performance of machine learning algorithms in predicting game outcome from drafts in dota 2," in *International Conference on Analysis of Images, Social Networks and Texts*, 2016.

[16] M. Aung, V. Bonometti, A. Drachen, P. Cowling, A. V. Kokkinakis, C. Yoder and A. Wade, "Predicting skill learning in a large, longitudinal moba dataset," in *IEEE Conference on Computational Intelligence and Games (CIG)*, 2018.

[17] I. Porokhnenko, P. Polezhaev and A. Shukhman, "Machine learning approaches to choose heroes in dota 2," in *24th Conference of Open Innovations Association (FRUCT)*, 2019.

[18] A. Drachen, M. Yancey, J. Maguire, D. Chu, I. Y. Wang, T. Mahlmann, M. Schubert and D. Klabajan, "Skill-based differences in spatio-temporal team behaviour in defence of the ancients 2 (dota 2)," in *IEEE Games Media Entertainment*, 2014.

[19] R. Smit, "A machine learning approach for recommending items in league of legends," 2019.

[20] A. R. Novak, K. J. Bennett, M. A. Pluss and J. Fransen, "Performance analysis in esports: modelling performance at the 2018 League of Legends World Championship," *International Journal of Sports Science & Coaching,* vol. 15, no. 5-6, pp. 809-817, 2020.

[21] X. Bang, W. Huiwen and R. Zhou, "What contributes to success in MOBA games? An empirical study of Defense of the Ancients 2," *Games and Culture,* vol. 14, no. 5, pp. 498-522, 2019.

[22] W. Michael and M. Deshen, "An analysis of artificial intelligence techniques in multiplayer online battle arena game environments," in *Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, 2016.

[23] J. Eichelbaum, H. Roony and H. Olaf, "Classification of Icon Type and Cooldown State in Video Game Replays," in *International Conference Image Analysis and Recognition*, 2018.

[24] R. Ani, V. Harikumar, A. K. Devan and O. S. Deepa, "Victory

prediction in League of Legends using Feature Selection and Ensemble methods," in *International Conference on Intelligent Computing and Control Systems (ICCS)*, 2019.

[25] S.-K. Lee, S.-J. Hong and S.-I. Yang, "Predicting Game Outcome in Multiplayer Online Battle Arena Games," in *International Conference on Information and Communication Technology Convergence (ICTC)*, 2020.

[26] S.-J. Hong, S.-K. Lee and S.-I. Yang, "Champion Recommendation System of League of Legends," in *International Conference on Information and Communication Technology Convergence (ICTC)*, 2020.

[27] L. Hanke and L. Chaimowicz, "A recommender system for hero line-ups in MOBA games," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2017.

[28] L. Yu, D. Zhang, X. Chen and X. Xie, "Moba-slice: A time slice based evaluation framework of relative advantage between teams in moba games," in *Workshop on Computer Games*, 2018.

[29] S. Kim, D. Kim, H. G. Ahn and B. Ahn, "Implementation of user playstyle coaching using video processing and statistical methods in league of legends," *Multimedia Tools and Applications,* pp. 1-13, 2020.

[30] A. Summerville, M. Cook and B. Steenhuisen, "Draft-analysis of the ancients: predicting draft picks in dota 2 using machine learning," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2016.

[31] X. Lan, L. Duan, W. Chen, R. Qin, T. Nummenmaa and J. Nummenmaa, "A player behavior model for predicting win-loss outcome in MOBA games," in *nternational Conference on Advanced Data Mining and Applications*, 2018.

[32] P. Goyal, A. Sapienza and E. Ferrara, "Recommending teammates with deep neural networks.," in *29th on Hypertext and Social Media*, 2018.

[33] A. Katona, R. Spick, V. J. Hodge, S. Demediuk, F. Block, A. Drachen and J. A. Walker, "Time to die: Death prediction in dota 2 using deep learning," in *IEEE Conference on Games (CoG)*, 2019.

[34] D. Ye, G. Chen, P. Zhao, F. Qiu, B. Yuan, W. Zhang, S. Chen, M. Sun, X. Li, S. Li, J. Liang, Z. Lian, B. Shi, L. Wang, T. Shi, Q. Fu, W. Yang and L. Huang, "Supervised Learning Achieves Human-Level Performance in MOBA Games: A Case Study of Honor of Kings," *IEEE Transactions on Neural Networks and Learning Systems,* pp. 1-11, 2020.

[35] A. Villa, V. Araujo, F. Cattan and D. Parra, "Interpretable Contextual Team-aware Item Recommendation: Application in Multiplayer Online Battle Arena Games," in *Fourteenth ACM Conference on Recommender Systems*, 2020.

[36] Q. Li, P. Xu, Y. Y. Chan, Y. Wang, Z. Wang, H. Qu and X. Ma, "A visual analytics approach for understanding reasons behind snowballing and comeback in moba games," *IEEE Transactions on Visualization and Computer Graphics,* vol. 23, no. 1, pp. 211-220, 2017.

[37] E. Carlini and A. Lulli, "A spatial analysis of multiplayer online battle arena mobility traces," in *European Conference on Parallel Processing*.

[38] A. Sapienza, H. Peng and E. Ferrara, "Performance dynamics and success in online games," in *IEEE International Conference on Data Mining Workshops (ICDMW)*.

[39] F. Block, V. Hodge, S. Hobson, N. Septhon, S. Devlin, M. F. Ursu, A. Drachen and P. I. Cowling, "Narrative bytes: Data-driven content production in esports.," in *ACM international conference on interactive experiences for TV and online video*, 2018.

[40] T. Gonçalves, P. Vieria, A. P. Alfonso, M. B. Carmo and T. Moucho, "Analysing player performance with animated maps," in *22nd international conference information visualisation (IV)*, 2018.

[41] A. P. Afonso, M. B. Carmo and T. Moucho, "Comparison of Visualization Tools for Matches Analysis of a MOBA Game," in *23rd International Conference Information Visualisation (IV)*, 2019.

[42] S. Ahmad, A. Bryant, E. Kleinman, Z. Teng, T.-H. D. Nguyen and M. S. El-Nasr, "Modeling Individual and Team Behavior through Spatio-temporal Analysis," in *Annual Symposium on Computer-Human Interaction in Play*, 2019.

[43] M. Anshori, F. Mar'i, M. W. Alauddin and F. A. Bachtiar, "Prediction result of dota 2 games using improved svm classifier based on particle swarm optimization," in *International Conference on Sustainable Information Engineering and Technology (SIET)*, 2018.

[44] H. Lie, D. Lukas, J. Liebig and R. Nayak, "A Novel Learning-to-Rank Method for Automated Camera Movement Control in E-Sports Spectating," in *Australasian Conference on Data Mining*, Singapore, 2018.

[45] F. F. do Nascimento, A. S. da Costa Melo, I. B. da Costa and L. B. Marinho, "Profiling successful team behaviors in League of Legends," in *23rd Brazillian Symposium on Multimedia and the Web*, 2017.

[46] S. Demediuk, A. Murrin, D. Bulger, M. Hitchens, A. Drachen, W. L. Raffe and M. Tamassia, "Player retention in league of legends: a study using survival analysis," in *Australasian Computer Science Week Multiconference*, 2018.

[47] S. A. Halim, Y. Indrianti, Y. Udjaja, J. Moniaga and B. A. Makalew, "The Repercussions of Game Multiplayer Online Battle Arena," in *International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, 2019.

[48] E. Kleinman, S. Ahmad, Z. Teng, A. Bryant, T.-H. D. Nguyen, C. Harteveld and M. S. El-Nasr, ""And then they died": Using Action Sequences for Data Driven, Context Aware Gameplay Analysis," in *International Conference on the Foundations of Digital Games*, 2020.

[49] L. M. Costa, A. C. C. Souza and F. C. M. Souza, "An approach for team composition in league of legends using genetic algorithm," in *18th Brazilian Symposium on*

Computer Games and Digital Entertainment (SBGames), 2019.

[50] T. Tsogbadrakh and M. Spradling, "Genetic Approach to Stable Partitioning in Online Role Based Hedonic Games," in *IEEE Congress on Evolutionary Computation (CEC)*, 2019.

[51] M. Mora-Cantallops and M.-Á. Sicilia, "Player-centric networks in League of Legends," *Social Networks,* vol. 55, pp. 149-159, 2018.

[52] M. Mora-Cantallops and M.-Á. Sicilia, "Team efficiency and network structure: The case of professional League of Legends," *Social Networks,* vol. 58, pp. 105-115, 2019.

[53] C. Ringer, J. A. Walker and M. Nicolaou, "Multimodal joint emotion and game context recognition in league of legends livestreams," in *IEEE Conference on Games (CoG)*, 2019.

[54] D.-H. Kim, C. Lee and K.-S. Chung, "A confidence-calibrated moba game winner predictor," in *IEEE Conference on Games (CoG)*, 2020.

[55] O. Cavadenti, V. Codocedo, J.-F. Boulicaut and M. Kaytoue, "What did i do wrong in my MOBA game? Mining patterns discriminating deviant behaviours," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016.

[56] N. P. H. Pratama, S. M. S. Nugroho and E. M. Yuniarno, "Fuzzy controller based AI for dynamic difficulty adjustment for defense of the Ancient 2 (DotA2)," in *Fuzzy controller based AI for dynamic difficulty adjustment for defense of the Ancient 2 (DotA2)*, 2016.

[57] M. Waltham and D. Moodley, "An analysis of artificial intelligence techniques in multiplayer online battle arena game environments," in *Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, 2016.

[58] T. D. Do, D. S. Yu, S. Anwer and S. I. Wang, "Using Collaborative Filtering to Recommend Champions in League of Legends," in *IEEE Conference on Games (CoG)*, 2020.

[59] Z. Cleghern, S. Lahiri, O. Özaltin and D. L. Roberts, "Predicting future states in dota 2 using value-split models of time series attribute data," in *12th International Conference on the Foundations of Digital Games*, 2017.

[60] I. Makarov, D. Savostyanov, B. Litvyakov and D. I. Ignatov, "Predicting winning team and probabilistic ratings in "Dota 2" and "Counter-Strike: Global Offensive" video games.," in *International Conference on Analysis of Images, Social Networks and Texts*, Cham, 2017.

[61] M. Hirth, K. Borchert, F. Allendorf, F. Metzger and T. Hoßfeld, "Crowd-based Study of Gameplay Impairments and Player Performance in DOTA 2," in *4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks*, 2019.

[62] L. Wang, Y. Tang and J. Liu, "WPQA: A Gaming Support System Based on Machine Learning and Knowledge Graph," in *Joint International Semantic Technology Conference*, Singapore, 2019.

[63] D. Gordeau and L. Archambault, "Discriminative neural network for hero selection in professional Heroes of the Storm and DOTA 2," *IEEE Transactions on Games,* 2020.

[64] L. Zhang, C. Xu, Y. Gao, Y. Han, X. Du and Z. Tian, "Improved Dota2 lineup recommendation model based on a bidirectional LSTM," *Tsinghua Science and Technology,* vol. 25, no. 6, pp. 712-720, 2020.

[65] N. Wang, L. Li, L. Xiao, G. Yang and Y. Zhou, "Outcome prediction of dota2 using machine learning methods," in *International Conference on Mathematics and Artificial Intelligence*, 2018.

[66] B. Xia, H. Wang and R. Zhou, "What contributes to success in MOBA games? An empirical study of Defense of the Ancients 2," *Games and Culture,* vol. 14, no. 5, pp. 498-522, 2019.

[67] J. G. Reitman, "Distributed cognition and temporal knowledge in league of legends," *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS),* vol. 10, no. 1, pp. 23-41, 2018.

[68] Z. Chen, Y. Yang, C. Tan, D. Cheng, A. Chen and Y. Zhuang, "What makes a good team? A large-scale study on the effect of team composition in Honor of Kings," in *The World Wide Web Conference*, 2019.

[69] V. Araujo, F. Rios and D. Parra, "Data mining for item recommendation in MOBA games," in *13th ACM Conference on Recommender Systems*, 2019.

[70] A. Kica, A. La Manna, L. O'Donell, T. Paolillo and M. Claypool, "Nerfs, Buffs and Bugs - Analysis of the Impact of Patching on League of Legends," in *International Conference on Collaboration Technologies and Systems (CTS)*, 2016.

About the Application of Autoencoders for Visual Defect Detection

Richárd Rádli
Faculty of Information Technology
University of Pannonia
Egyetem u. 10.
Veszprém, Hungary
radli.richard@virt.uni-pannon.hu

László Czúni
Faculty of Information Technology
University of Pannonia
Egyetem u. 10.
Veszprém, Hungary
czuni@almos.uni-pannon.hu

ABSTRACT

Visual defect detection is a key technology in modern industrial manufacturing systems. There are many possible appearances of product defects, including distortions in color, shape, contamination, missing or superfluous parts. For the detection of those, besides traditional image processing techniques, convolutional neural networks based methods have also appeared to avoid the usage of hand-crafted features and to build more efficient detection mechanisms. In our article we deal with autoencoder convolutional networks (AEs) which do not require examples of defects for training. Unfortunately, the manual and/or trial-and-error design of AEs is still required to achieve good performance, since there are many unknown parameters of AEs which can greatly influence the detection abilities. For our study we have chosen a well performing AE known as structural similarity AE (SSIM-AE), where the loss function and the comparison of the output with the input is implemented via the SSIM instead of the often used L1 or L2 norms. Investigating the performance of SSIM-AE on different data-sets, we found that its performance can be improved with modified convolutional structures without modifying the size of latent space. We also show that finding a model with low reconstruction error during training does not mean good detection abilities and denoising AEs can increase efficiency.

Keywords

autoencoder neural network, convolutional neural network, defect detection, unsupervised anomaly detection

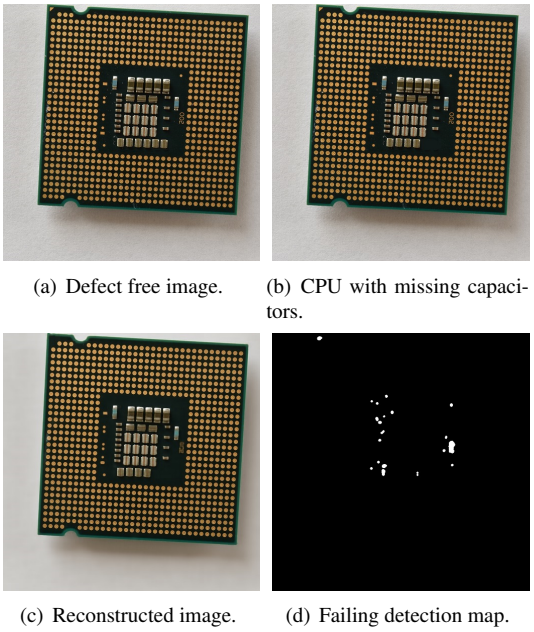
1 INTRODUCTION

There are several applications of neural networks in manufacturing systems [Wang2018], visual inspection of products is a critical step during their production. To have the greatest freedom in the handling and imaging of objects we need sophisticated methods to allow different distortions such as changes in lighting, position and orientation, especially in case of 3D objects. Due to this problem, in recent times, different deep learning solutions have been developed, autoencoder (AE) neural networks are very promising for anomaly detection in visual quality inspection, surveillance, or medical image analysis.

As we will discuss, there are several types of autoencoders, it is not clear which one suits best unsupervised defect detection and segmentation. Recent improvements in generative networks mainly focused on creating photo-realistic "natural" images as faces, nature, cityscapes, etc. In defect detection, while we have the advantage of the narrower image domain (we can train separate AEs for each class of products), reconstruction errors are much less tolerable, otherwise the succeeding detection phase may fail. Moreover, there is a large set of parameters which significantly influence detection abilities. Our purpose is to investigate the applicability of a well-performing autoencoder (namely SSIM-AE) for the fault detection on repetitive and partly repetitive structures. A great advantage of using AEs is that unsupervised training and detection is possible without making efforts to collect samples with anomalies. However, there is no guarantee that images with errors will not be reconstructed making detection unsuccessful since many times errors are reproduced in the decoded image. Fig. 1 shows a good example for failing anomaly detection with the otherwise well performing SSIM-AE [Bergmann2019]. Input image has a missing capacitor and the AE reproduced the image with the same defect.

First, we overview the different approaches for visual anomaly detection mainly focusing on AE networks, then, in Section 3, the SSIM-AE and its variants are introduced. Section 4 details the selected data-sets,

while in Section 5, we explain our experiments and discuss the performance of SSIM-AEs with different settings for the pixel-wise segmentation of different errors. Finally, we summarize our paper in the last section.



(a) Defect free image. (b) CPU with missing capacitors.

(c) Reconstructed image. (d) Failing detection map.

Figure 1: Illustration of failed detection of missing components from a CPU with SSIM-AE at resolution 512×512.

2 ALTERNATIVES FOR VISUAL INSPECTION

Optical defect detection approaches can be grouped in the following main categories: Template matching [Buniatyan2017], [Zhou2019] local feature based methods [Gai2016],[Xi2017], statistical texture models [Cogranne2014],[Cao2015], neural networks [Weimer2016], [Tuluptceva2019], [Tuluptceva2020], [Alaverdyan2020], [Nagy2021]. There can be overlap among these groups, and each category have some advantages, but there is no doubt that due to the generality, robustness and accuracy of deep learning convolutional AE approaches, they are getting more focus in recent years. In our article we discuss only AE networks having the main advantage of unsupervised training and the ability to handle various imaging conditions.

2.1 About Autoencoders

An autoencoder is a special kind of feed-forward neural network containing three main sequential components: the encoder network $E : \mathbb{R}^{k\times h\times w} \to \mathbb{R}^d$, the latent space $\mathbb{R}^d$, and the decoder network $D : \mathbb{R}^d \to \mathbb{R}^{k\times h\times w}$.

$$\hat{\mathbf{x}} = D(E(\mathbf{x})) = D(\mathbf{z}), \tag{1}$$

where $\mathbf{x}$ is the input image, $\mathbf{z}$ is the latent information, and $\hat{\mathbf{x}}$ is the output of the network.
AEs were originally used for dimensionality reduction, feature extraction, but nowadays they are very popular tools for generative modeling. The similarity of principal component analysis (PCA) and AEs is well know, moreover, in many scenarios AEs outperformed linear or kernel PCAs [Sakurada2014].
The low-level neural structure in AEs can be convolutional or can utilize extreme learning machines [Kasun2013].
They can also be categorized according to their regularization techniques. The loss function (J), to learn a specific task, can be generally formed this way:

$$\mathbf{J}(\mathbf{x},\theta) = \frac{1}{n}\sum \|\mathbf{x}-\hat{\mathbf{x}}\| + R(\mathbf{z}), \tag{2}$$

where $\theta = (w_E, b_E, w_D, b_D)$ are the weights and biases of the encoder and decoder networks and n is the number of elements. In sparse AEs R is based on the Kullback-Leibler divergence or on the L1 norm, the purpose is to make the most of the hidden unit's activations close to zero. Contractive AEs apply the Frobenius norm on the derivative of $\mathbf{z}$ as a function of $\mathbf{x}$ to make the model resistant to small perturbations; and in an information theoretic-learning autoencoder Renyi's entropy is used.
Variational AEs (VAEs) impose constraints on the latent variables in a different way, they estimate posteriori probability $p(\mathbf{z}|\mathbf{x})$ with an assumption of a prior knowledge $p(\mathbf{z})$ being a normal Gaussian distribution. Adversarial AEs (AAEs) use generative adversarial networks (GANs) to perform variational inference. VAEs and AAEs are both generative models, and both are based on maximum likelihood. The difference between VAEs and AAEs can be characterized that while VAEs apply explicit rules on $\mathbf{z}$, AAEs control its distribution implicitly.
Denoising autoencoder (DAE) is an extension of the basic autoencoder, which is trained to reconstruct corrupted input data. In [Vincent2010] not only Gaussian noise, but also salt-and-pepper noise and zero-masking noise was also investigated. C denotes the corrupting function generating $\tilde{\mathbf{x}}$. Eq. 1 now becomes:

$$\hat{\mathbf{x}} = D(E(C(\mathbf{x}))) = D(E(\tilde{\mathbf{x}})) = D(\mathbf{z}). \tag{3}$$

The applied loss puts an emphasis on the corrupted areas by proper weighting:

$$\mathbf{J}(\mathbf{x},\tilde{\mathbf{x}},\theta) = \alpha \sum_{i\in\mathscr{C}} (\mathbf{x_i}-\hat{\mathbf{x}}_\mathbf{i})^2 + \beta \sum_{i\notin\mathscr{C}} (\mathbf{x_i}-\hat{\mathbf{x}}_\mathbf{i})^2, \tag{4}$$

where $\mathscr{C}$ denotes the indexes of corrupted components. Feature matching autoencoder [Dosovitskiy] enforces the features of the input and its reconstruction to be equal:

$$\mathbf{J}(\mathbf{x},\tilde{\mathbf{x}},\theta) = w_{im}\sum(\mathbf{x}-\hat{\mathbf{x}})^2 + w_{ft}\sum(F(\mathbf{x})-F(\hat{\mathbf{x}}))^2, \tag{5}$$

where F is a feature extractor which can be fixed or also trained. In [Dosovitskiy], besides using features for evaluation, adversarial discriminator was also utilized.
We guide people, with more interest in the overview of AEs, to papers [Zhai2018] and [Yuan2019].

2.2 Autoencoders for Defect Detection

In general, trained AEs can generate very similar outputs to their inputs, if the later fits the trained model or with other words, the image models could learn the inherent features properly. That is if the difference between the two is above a threshold Th, we set the detection map to 1:

$$\mathbf{m} = \begin{cases} 1, & \text{if } ||\mathbf{x}-\hat{\mathbf{x}}|| > Th, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The applied $||.||$ can be various (e.g. L1, L2, SSIM [Bergmann2019]).
In [Beggel2019] they deal with the problem of training AEs when the training set is contaminated with a small fraction of outliers. Their proposed adversarial autoencoder imposes a prior distribution on the latent representation, placing anomalies into low likelihood regions, thus potential anomalies can be identified and rejected already during training.
[Tuluptceva2020] proposes perceptual deep autoencoders where relative-perceptual-L1 loss [Tuluptceva2019], robust to low contrast and noise, is applied for training and also to predict the abnormality for new inputs. The applied progressive growing technique helped the efficiency of training: in the beginning of the training, the loss function computes the dissimilarity between the low-resolution images using the features from the coarse layers of the network, whereas, as the training advances, the "level" of this information is increased by including deeper and deeper features. The addition of the new layers to the autoencoder with the gradual increase of the depth of the features entailed in the calculation of the perceptual loss was synchronized.
Siamese networks are good for supervised anomaly detection, since they can learn discriminating features efficiently [Nagy2021]. In [Alaverdyan2020] an unsupervised siamese convolutional autoencoder is proposed to detect anomalies in brain MRI images. Anomalies are detected by a one class SVM in latent space. The role of the siamese network is to regularize the latent space: $R(z)$ (Eq. 2) is the cosine distance of two independent samples in the two siamese branches in the latent space. Since each voxel of the MRI data is handled independently (with its own SVM), thus anomalies are learnt for every location.
The approach we utilize in our study is from [Bergmann2019], which applies structural similarity (SSIM) in the loss function. It was reported that it outperforms many other AEs and reaches the state-of-art on some data-set. It is discussed in details in the next section.

3 SSIM AUTOENCODERS FOR DEFECT DETECTION

In our understanding SSIM-AE [Bergmann2019] is a kind of feature matching AE (Eq. 5) with the following loss function:

$$\mathbf{J}(\mathbf{x},\theta) = \frac{1-SSIM(\mathbf{x},\hat{\mathbf{x}})}{2} = \frac{1-l(\mathbf{x},\hat{\mathbf{x}})^{\alpha}c(\mathbf{x},\hat{\mathbf{x}})^{\beta}s(\mathbf{x},\hat{\mathbf{x}})^{\gamma}}{2}. \tag{7}$$

l stands for the luminance measure, estimated by comparing the patches' mean intensities, c for the contrast responsible for the local variance, and s evaluates the covariance of patches. We set all three exponents to 1, and other inner variables (not detailed here) as follows: $c_1 = 0.01$, $c_2 = 0.03$, and the window size of the patches $K = 11$. SSIM is not only utilized in the loss but also in the comparison of the input and outputs:

$$\mathbf{m} = \begin{cases} 1, & \text{if } \frac{1-SSIM(\mathbf{x},\hat{\mathbf{x}})}{2} > Th, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Since SSIM already includes the luminance of pixel values, $w_{im} = 0$ in Eq. 5, colors are simply neglected. The SSIM index was originally designed to create an accurate perceptual quality metrics for the comparison of two $K \times K$ image patches [Wang2004]. In our case not the perceptual information makes it appealing to use it in the loss function but its efficient extraction of structural information: SSIM-AE is forced to represent, beside luminance, local variance and covariance. According to [Bergmann2019] SSIM-AE could significantly outperform the FM-AE of [Dosovitskiy] and a tested VAE. The code available at the Internet[1] is not fully identical to the description in their paper. In our experiments we followed the paper except for setting the stride of overlapping windows when fusing the cropped images for reconstruction (32 instead of 30).
The structure of the encoder is given in Table 1, the decoder has the opposite structure, as generally.

[1] https://github.com/plutoyuxie/AutoEncoder-SSIM-for-unsupervised-anomaly-detection-

Layer	Output Size	Kernel	Stride	Padding
Input	128x128x3			
Conv1	64x64x32	4x4	2	1
Conv2	32x32x32	4x4	2	1
Conv3	32x32x32	3x3	1	1
Conv4	16x16x64	4x4	2	1
Conv5	16x16x64	3x3	1	1
Conv6	8x8x128	4x4	2	1
Conv7	8x8x64	3x3	1	1
Conv8	8x8x32	3x3	1	1
Conv9	1x1xd	8x8	1	0

Table 1: Architecture of the encoder of SSIM-AE [Bergmann2019] for input image $128 \times 128 \times 3$. $d = 100$ for the texture images, $d = 500$ for the CPUs.

Layer	Output Size	Kernel	Stride	Padding
Input	128x128x3	3x3		
Conv1	64x64x32	3x3	2	1
Conv2	64x64x32	3x3	1	1
Conv3	64x64x32	3x3	1	1
Conv4	32x32x64	3x3	2	1
Conv5	32x32x64	3x3	1	1
Conv6	32x32x64	3x3	1	1
Conv7	16x16x128	3x3	2	1
Conv8	16x16x128	3x3	1	1
Conv9	16x16x128	3x3	1	1
Conv10	8x8x256	3x3	2	1
Conv11	8x8x256	3x3	1	1
Conv12	8x8x256	3x3	1	1
Conv13	1x1xd	8x8	1	0

Table 2: Architecture of the encoder of SSIM-AEe for input image $128 \times 128 \times 3$. $d = 100, 500$ for the texture images, $d = 500$ for the CPUs.

In our experiments we investigated several questions: Can the results be improved by increasing the number of layers but keeping the size of the latent space? How does SSIM-AE perform in case of less periodic structures? What happens if some components are added or removed from the original images?
We made two modifications to the original SSIM-AE:

1. In one modification (named SSIM-AEe) we increased the number of convolutional layers as depicted in Table 2. We kept the latent layers untouched.

2. In SSIM-DAE we followed Eq. 3 to build a denoising AE trained with masked blocks (see Section 5 for details). The purpose of our SSIM-DAE was to implement an implicit regularization of the network to reconstruct patches from a larger vicinity. There is low probability that during decoding both the target and source areas are all affected by defects. But either one is distorted $||\hat{x} - \tilde{x}||$ will have high value.

4 DATA-SETS

The visually observable defects can be various such as faulty color, contamination, geometrical distortion, missing parts, or superfluous parts. We assume that our objects are almost 2D (like the surface of a rectangular object without significant bulges or holes), and images have variations in translation, orientation, and lighting conditions.
In our study, we used five data-sets. Two of them, provided by [Bergmann2019], contain regular patterns of woven fabric textures. Each of these two sets include 100 defect-free images, and 50 test images with various defects. Ground truth images were also provided in binary maps. We have chosen these textures, since they are good representatives of roughly regular repetitive patterns. Since we focus on convolutional AEs, the radius of convolutions and the number of layers can have a great impact on the generation abilities depending on the texture size.
The other data-sets involve 60 genuine images of the backside of a CPU of size 37.50 mm $\times$ 37.50 mm. We separated 48 images for training purposes and the remaining 12 for various tests. Three different test cases were generated for the CPUs:

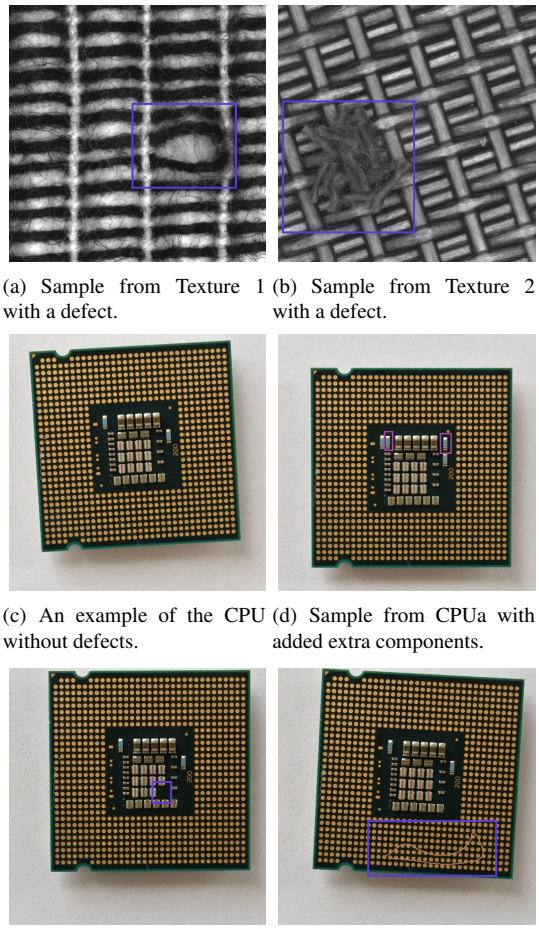
- CPUa: we added extra components with visually realistic appearance;
- CPUc: the test images are loaded with synthetic contamination;
- CPUm: some components (such as pins or capacitors) are removed.

Table 3 summarizes the mean and standard deviation of the defected areas for all 5 data-sets. In case of textures, anomalies are significantly larger.

	Texture 1	Texture 2	CPUa	CPUc	CPUm
mean	8.00	5.52	0.16	0.69	0.31
standard deviation	4.56	2.20	0.07	0.26	0.28

Table 3: Mean and standard deviation of defected areas (in percentage) in the different test sets.

The selection of the CPU images is twofold: first, such and similar images are typical in the production of electronic components; second, the image of the CPU contains regularly repetitive (outward areas with contacts) and regular but much less repetitive regions (central capacitor area). Besides, the CPU images contain more finer details than the textures.
All images are of size 512×512, their illustration with defects is in Fig. 2. Ground truth images were also provided as binary maps.



(a) Sample from Texture 1 with a defect.
(b) Sample from Texture 2 with a defect.
(c) An example of the CPU without defects.
(d) Sample from CPUa with added extra components.
(e) Sample from CPUm with a missing capacitor.
(f) Sample from CPUc with synthetic contamination.

Figure 2: Illustration of test images with various defects.

5 TRAINING, TESTING, AND EVALUATION

During training we followed the method and hyper-parameters described in [Bergmann2019]. They first resized images to 256 × 256, then image cropping was applied to the size of 128 × 128. Networks were trained on NVIDIA RTX 2070 GPU for 200 epochs with ADAM optimizer, initial learning rate was 2×10^{-4} with 10^{-5} weight decay. The latent space was set to $d = 100$ and $d = 500$ for Texture 1 and Texture 2, and $d = 500$ for the CPU, considering its structural complexity and finer details. Furthermore, the window size of the SSIM was adjusted to 11.

In [Bergmann2019], for both texture data-sets, they increased the variability of training data by generating 10,000 images using various augmentation procedures such as rotation, cropping, and flipping (both vertically and horizontally). In our study, we did the same, except in the case of the CPU tests, where no flip was involved to keep the asymmetric structure.

To train the SSIM-DAE we added noise to the input images: small, homogeneous masks were added to the 48 images of the original training data-set. The size of masks were set to 20×20, being greater than the size of one capacitor in the middle of the CPU; the color was set to the dominant color of the actual (128×128) crop. Each training image contained one masks, $\alpha = \beta = 1$ in Eq. 4.

For evaluation, we have chosen the standard receiver operating characteristic (ROC) curves and the area under curve (AUC) values. We defined the true positive rate (TPR) as the rate of pixels that were correctly classified as defect, and false positive rate (FPR) as the ratio of pixels that were misclassified as defect. The process of testing was accomplished by the reconstructions of image patches of 128 × 128, by moving over the test images (with stride of 32). Residual maps were created using SSIM (Eq. 8), then ROC curves and AUC values were computed thresholding with increasing SSIM values.

To monitor the reconstruction abilities of the different networks, we also computed the SSIM and mean squared error (MSE) of reconstructions for defect-free testing images (last columns of Table 5).

5.1 Results and Discussion

Four kinds of AEs (SSIM-AE, SSIM-AEe, SSIM-DAE, and SSIM-DAEe) were compared in 5 test cases (Texture 1, Texture 2, CPUa, CPUc, CPUm) as described before. Results are summarized in Table 4 and Table 5 (best values are in bold), ROC curves are illustrated in Fig. 4 and Fig. 5.

For Texture 1 and Texture 2 SSIM-AE performed very well, neither increasing the number of convolutions (SSIM-AEe), nor increasing the size of the latent space (from $d = 100$ to $d = 500$) could make (significant) achievements. In general, Texture 1 and 2 are missing fine details (see Fig. 2 a and b) making deeper convolutions useless.

In case of the CPU tests we got different results. Comparing SSIM-AE to SSIM-AEe we see that increasing the number of convolutional layers could increase the accuracy of reconstruction but it could not help the ability to detect anomalies (except for CPUm). That is if a network is better in the reconstruction of defect-less items, it does not mean it is also better to mishandle defected inputs (thus to detect anomalies).

Since the CPU images had more details, SSIM-AEe could learn the image models more accurately than SSIM-AE (even with the same latent space): it is shown by MSE and SSIM values, measured between error-less inputs and their reconstructions, in Table 5. Since CPUa contained small anomalies, detection went quite well (AUC is between 0.9822 and 0.9952), while CPUc had the worst results. Finally, we found that the larger number of layers and the denoising type training of SSIM-DAEe resulted in the best performance for all

CPU test cases (bold in table).

	Texture 1	Texture 2
SSIM-AE	**0.9490**	0.9710
SSIM-AEe	0.9445	0.9697
SSIM-AE, $d = 500$	0.9186	**0.9773**
SSIM-AEe, $d = 500$	0.9118	0.9709

Table 4: AUC values when testing AE methods on data-set Texture 1 and Texture 2.

	CPUa	CPUc	CPUm	SSIM	MSE
SSIM-AE	0.9930	0.9071	0.9262	0.7263	90.0472
SSIM-AEe	0.9822	0.8939	0.9557	0.8195	66.5257
SSIM-DAE	0.9914	0.9059	0.9651	0.7851	88.7726
SSIM-DAEe	**0.9952**	**0.9499**	**0.9815**	**0.9133**	**57.3972**

Table 5: Testing AE methods for different distortions on the CPU data-set. AUC values are given in the first three columns, SSIM and MSE is measured as the reconstruction error of error free test samples. SSIM is on the [-1,1] scale, MSE stands for Mean Squared Error.

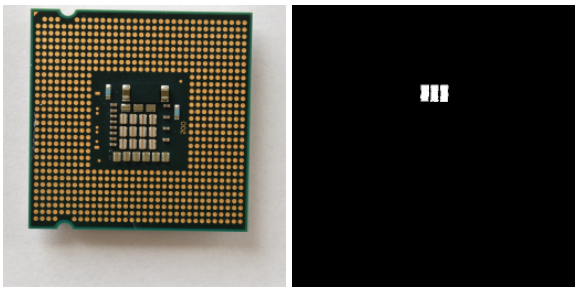
6 CONCLUSION

In case of visual quality inspection we are often not satisfied with the statistical definition of normal patterns and anomalies. For many products the strict topological structures should be retained during production, thus, the otherwise normal, missing or superfluous parts are not acceptable.
We investigated the performance of different SSIM autoencoders for defect detection on repetitive and semi-repetitive structures on 5 data-sets. By introducing modifications to the proposed AE of [Bergmann2019] we found, that:

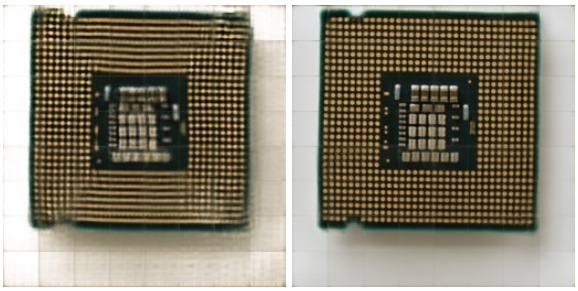
1. If an AE can reproduce (anomaly free) images with higher fidelity it does not strictly induce it is better in anomaly detection. It is an important observation, since AE's main application area is unsupervised anomaly detection, where no examples of defects are available to design and train optimal neural networks.

2. Unsupervised AEs are not reliable for (at least small) contamination or superfluous parts, DAEs may fit many defect detection tasks better.

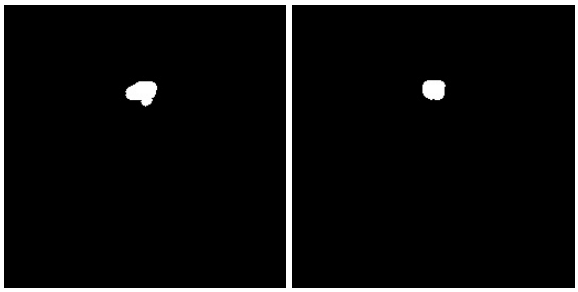
The proposed SSIM-DAEe outperformed other variants in our CPU experiments without the growth of the latent space, while in case of textures, the DAEe could not increase detection accuracy. (We also outperformed those feature matching and VAE approaches which are used as reference methods in [Bergmann2019]). Some illustrations of the results on CPU images are given in Fig. 3 for the SSIM-AE and SSIM-DAEe.
We believe current AEs are very far from their limit for defect detection, most recent regularization techniques consider the probability density functions with less emphasis on strict structural features.
No systematic design is known to optimally set the encoder, decoder, and latent parts, or the type of regularization. In future we plan to make more tests on the industrial data-sets of [Bergmann2021], and to build more structure awareness in the neural models investigating structural and contextual attention similar to [Yu2018].



(a) Input test image with missing capacitor and ground truth.



(b) Reconstruction with SSIM-AE and SSIM-DAEe.

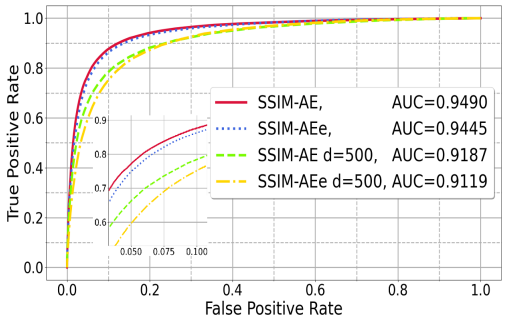


(c) Corresponding detection maps with SSIM threshold set to 0.8425

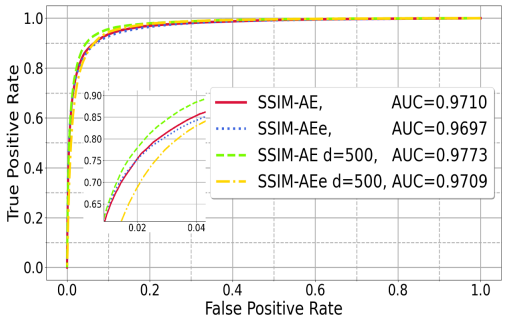
Figure 3: Illustration of the detection performance of SSIM-AE and SSIM-DAE for a CPU with missing capacitor.

7 ACKNOWLEDGMENTS

We are grateful to the NVIDIA corporation for supporting our research with GPUs obtained by the NVIDIA GPU Grant Program. We acknowledge the financial support of the projects 2018-1.3.1-VKE-2018-00048 under the ÚNKP-19-3 New National Excellence Program, 2020-4.1.1-TKP2020 under the Thematic Excellence Program, and the Hungarian Research Fund grant OTKA K 135729.

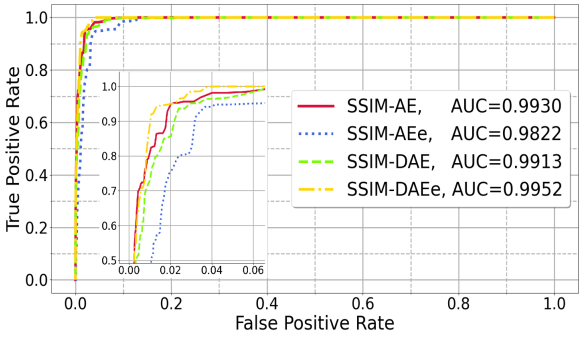


(a) AEs on Texture 1 at resolution 256 × 256.

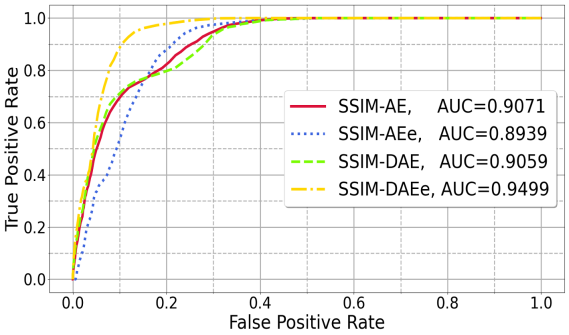


(b) AEs on Texture 2 at resolution 256 × 256.

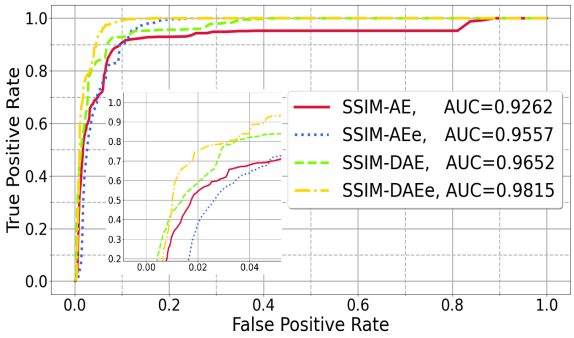
Figure 4: The performance of the AEs on Texture 1 and Texture 2.



(a) The performance of different AEs on CPUa.



(b) The performance of different AEs on CPUc.



(c) The performance of different AEs on CPUm.

Figure 5: The performance of different AEs on CPUa, CPUc, and CPUm tests.

8 REFERENCES

[Alaverdyan2020] Alaverdyan, Z., Jung, J., Bouet, R., Lartizien, C. (2020). Regularized siamese neural network for unsupervised outlier detection on brain multiparametric magnetic resonance imaging: application to epilepsy lesion screening. Medical image analysis, 60, 101618.

[Beggel2019] Beggel, L., Pfeiffer, M., Bischl, B. (2019, September). Robust anomaly detection in images using adversarial autoencoders. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 206-222). Springer, Cham.

[Bergmann2019] Bergmann, P.; Löwe, S.; Fauser, M.; Sattlegger, D. and Steger, C. (2019). Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders. VISAPP, pages 372-380.

[Bergmann2021] Bergmann, P., Batzner, K., Fauser, M., Sattlegger, D., Steger, C. (2021). The MVTec Anomaly Detection data set: A Comprehensive Real-World data set for Unsupervised Anomaly Detection. International Journal of Computer Vision, 1-22.

[Buniatyan2017] D. Buniatyan, T. Macrina, D. Ih, J. Zung, and H. S. Seung, Deep learning improves template matching by normalized cross correlation. arXiv preprint arXiv:1705.08593, 2017.

[Cao2015] Cao, J., Zhang, J., Wen, Z., Wang, N., and Liu, X. (2015). Fabric defect inspection using prior knowledge guided least squares regression. Multimedia Tools and Applications, 76:4141-4157.

[Cogranne2014] R. Cogranne and F. Retraint, Statistical detection of defects in radiographic images using an adaptive parametric model. Signal Processing, vol. 96, pp. 173-189, 2014.

[Dosovitskiy] Dosovitskiy, A., Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. arXiv preprint

arXiv:1602.02644.

[Gai2016] Gai, S. (2016). New banknote defect detection algorithm using quaternion wavelet transform. Neurocomputing, 196, 133-139.

[Kasun2013] Kasun L.L.C.,Zhou H.,Huang G.-B.,et al. Representational learning with ELMs for big data[J]. IEEE Intelligent Systems,2013,28(6):31-34.

[Nagy2021] A. Nagy and L. Czúni, Detecting Object Defects with Fusioning Convolutional Siamese Neural Networks. In Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP, pages 157-163

[Sakurada2014] Sakurada, M., Yairi, T. (2014, December). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis (pp. 4-11).

[Santana2016] Santana, E., Emigh, M., Principe, J. C. (2016, July). Information theoretic-learning auto-encoder. In 2016 International Joint Conference on Neural Networks (IJCNN) (pp. 3296-3301). IEEE.

[Tuluptceva2019] Tuluptceva, N., Bakker, B., Fedulova, I., Konushin, A. (2019, November). Perceptual image anomaly detection. In Asian Conference on Pattern Recognition (pp. 164-178). Springer, Cham.

[Tuluptceva2020] Tuluptceva, N., Bakker, B., Fedulova, I., Schulz, H., Dylov, D. V. (2020). Anomaly detection with deep perceptual autoencoders. arXiv preprint arXiv:2006.13265.

[Xi2017] Xi, J., Shentu, L., Hu, J., Li, M. (2017). Automated surface inspection for steel products using computer vision approach. Applied optics, 56(2), 184-192.

[Yu2018] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T. S. (2018). Generative image inpainting with contextual attention. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5505-5514).

[Yuan2019] Yuan, F.-N Zhang, L. Shi, J.-T Xia, X. Li, G. (2019). Theories and Applications of Auto-Encoder Neural Networks: A Literature Survey. Jisuanji Xuebao/Chinese Journal of Computers. 42. 203-230. 10.11897/SP.J.1016.2019.00203.

[Vincent2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P. A., Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(12).

[Wang2004] Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), 600-612.

[Wang2018] Wang, J., Ma, Y., Zhang, L., Gao, R. X., Wu, D. (2018). Deep learning for smart manufacturing: Methods and applications. Journal of Manufacturing Systems, 48, 144-156.

[Weimer2016] Weimer, D., Scholz-Reiter, B., Shpitalni, M. (2016). Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. CIRP Annals, 65(1), 417-420.

[Zhai2018] Zhai, J., Zhang, S., Chen, J., He, Q. (2018, October). Autoencoder and its various variants. In 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 415-419). IEEE.

[Zhou2019] X. Zhou, Y. Wang, C. Xiao, Q. Zhu, X. Lu, H. Zhang, J. Ge, and H. Zhao, Automated visual inspection of glass bottle bottom with saliency detection and template matching. IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 11, pp. 4253-4267, 2019.

Last page should be fully used by text, figures etc. Do not leave empty space, please.

Do not lock the PDF – additional text and info will be inserted, i.e. ISSN/ISBN etc.

The Influence of Coding Tools on Immersive Video Coding

Jakub Szekiełda Adrian Dziembowski Dawid Mieloch
adrian.dziembowski@put.poznan.pl

Institute of Multimedia Telecommunications, Poznań University of Technology
Polanka 3, 61-131 Poznań, Poland

ABSTRACT

This paper summarizes the research on the influence of HEVC (High Efficiency Video Coding) configuration on immersive video coding. The research was focused on the newest MPEG standard for immersive video compression – MIV (MPEG Immersive Video). The MIV standard is used as a preprocessing step before the typical video compression thus is agnostic to the video codec. Uncommon characteristics of videos produced by MIV causes, that the typical configuration of the video encoder (optimized for compression of natural sequences) is not optimal for such content. The experimental results prove, that the performance of video compression for immersive video can be significantly increased when selected coding tools are being used.

Keywords

Immersive video coding, immersive video systems, video compression.

1. INTRODUCTION

Many modern multimedia systems that include steps in which the video compression is applied can be described as codec-agnostic. It means that any video codec can be utilized, so the selection of the compression method is completely transparent to the rest of the system. Obvious examples of codec-agnostic video systems and methods are streaming-related methods (e.g., MPEG Dynamic Adaptive Streaming over HTTP [Sod11]) or simple simulcast compression required, e.g., in surveillance or free-viewpoint television systems [Sta18].

Besides these applications, recently a new trend of using existing compression methods as an internal processing tool in new video codecs can be seen. The latest examples are MPEG-5 LCEVC (Low Complexity Enhancement Video Coding) [Mea20] that introduces an enhancement layer which, when combined with a base video encoded with another existing video codec, produces an enhanced video stream, or VPCC (Video-based Point Cloud Coding) [Gra20] or MIV (MPEG Immersive Video) [Boy21] that utilize video compression for dynamic three-dimensional scenes and objects. In immersive video, user can change the viewpoint and is not limited to watch views acquired by cameras located around a scene. While the use of existing state-of-the-art compression methods makes it easier to develop new codecs for more and more new emerging technologies, configuration of internal coder is often not optimized for these applications, as the default configuration usually provides already satisfactory results.

This paper describes a different approach, in which adaptation of the internal coder configuration is performed, while the external one is not changed. The proposed experiments focus on the influence of HEVC configuration on immersive video coding performed by MIV. In MIV, some views (base views) are fully transmitted, while for others (additional views) only the non-redundant information is included in atlases – synthetic videos containing information from many input views (as a mosaic of patches – Fig. 2).

The content of atlases highly varies from the typical video sequences, motivating the need for using a non-standard set of coding tools and testing their configuration.

2. EXPERIMENTS METHODOLOGY

Overview

In the experiments (Fig. 1), the input views together with depth maps were processed in the TMIV (Test Model for MPEG Immersive Video) [MPEG21a] encoder. It outputs 4 atlases: 2 containing texture information (called T0 and T1) and 2 containing depth information with reduced resolution (G0 and G1). Then, each atlas was separately encoded with x265 video encoder [X265] with 5 QP values: 22, 27, 32, 37, and 42 for texture and 4, 7, 11, 15, and 20 for depth.

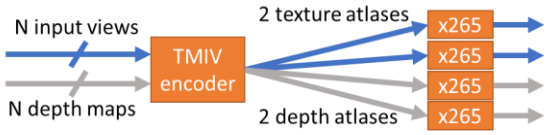


Figure 1. Scheme of an experiment.

The TMIV encoder works in two major configurations: "MIV Atlas" (A17, presented earlier) and "MIV View" (V17, Fig. 3). In the MIV View configuration, a subset of input views is transmitted within atlases. The remaining views are completely skipped thus some information is ignored (e.g., for Group sequence, only 8 of 21 input views are transmitted). We tested both configurations, thus each experiment was run twice.

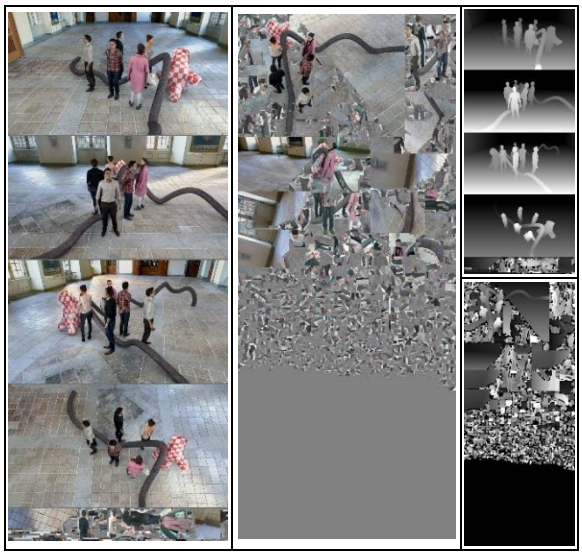


Figure 2. Atlases for Group sequence, MIV Atlas (A17). From left: atlas T0, T1, G0 (top), and G1.

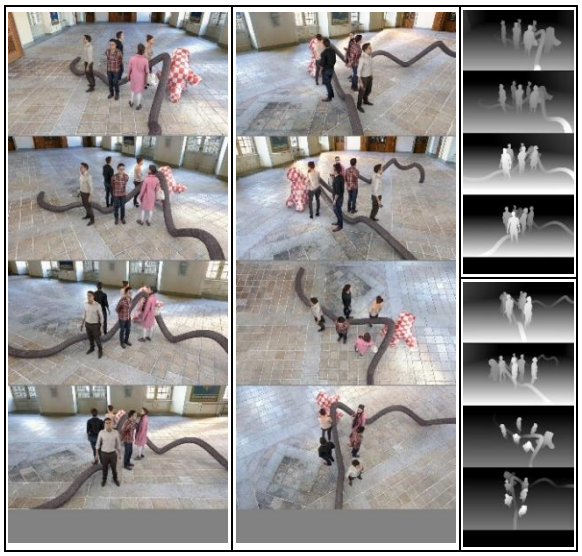


Figure 3. Atlases for Group sequence, MIV View (V17). From left: atlas T0, T1, G0 (top), and G1.

To preserve the readability of the paper, results obtained for all 5 QP values and all 14 test sequences were averaged.

We have decided to use the x265 encoder [X265] because of two main reasons. At first, it allows to flexibly change numerous encoding parameters, so we can easily analyze, which aspects of video encoding influence the immersive video encoding the most. Secondly, x265 is a very fast encoder, it is two orders of magnitude faster than HEVC Test Model (HM) [MPEG17] – the HEVC Test Model [Sze20].

In the experiments, 13 encoding parameters were tested. For some of them, several tests were performed resulting in 22 experiments in total. We have tested parameters, which potentially could improve the encoding efficiency of the immersive video:

1. b-adapt – flexibility of setting the GOP (group of pictures) structure,
2. bframes – maximum number of consecutive B-frames (bidirectional-predicted frames),
3. bframe-bias – probability of choosing B-frames,
4. lookahead-slices – number of threads used for frame cost calculation,
5. max-merge – maximum number of neighboring blocks analyzed in motion prediction,
6. me – motion search method (method of searching of corresponding blocks in previously-encoded frames),
7. no-early-skip – additional analysis of possible modes providing better quality in increased time,
8. rd – rate-distortion optimization level,
9. rdoq-level – level of rate-distortion analysis within quantization step,
10. rect – rectangular motion partitioning,
11. rect amp – rectangular motion partitioning with the possibility of asymmetric partitioning,
12. ref – number of L0 references,
13. subme – subpixel refinement level.

#	x265 parameter	default value	tested value	other changes
1	--b-adapt	2	0	
2			1	
3	--bframes	4	16	
4	--b-adapt	2	0	--bframes 16
5			1	
6	--bframe-bias	0	100	
7	--lookahead-slices	8	1	
8			4	
9	--max-merge	2	3	
10	--me	hex	dia	
11			umh	
12			star	
13	--no-early-skip	disabled	enabled	
14	--rd	3	1	
15			2	
16			5	
17	--rdoq-level	0	2	
18	--rect	disabled	enabled	--limit-modes
19	--rect --amp	disabled	enabled	
20	--ref	3	4	
21	--subme	2	3	
22	*optimal configuration*			

Table 1. Performed experiments.

Test sequences

The test set contained 14 miscellaneous high-resolution test sequences, including 7 synthetic, computer-generated sequences (Fig. 5):

1. ClassroomVideo, 16 full-360° 4K×2K cameras [Kro18],
2. Museum, 24 semispherical 2K×2K cameras placed on the sphere [Dor18],
3. Hijack, 10 parallel 4K×2K cameras with angle of view 180°×90° [Dor18].
4. Chess, 10 semispherical 2K×2K cameras placed on the sphere [Ilo19],
5. ChessPieces, 10 semispherical 2K×2K cameras placed on the sphere [Ilo20],
6. Kitchen, 25 perspective FullHD cameras placed in the 5×5 matrix [Boi18],
7. Fan, 15 perspective FullHD cameras placed in the 5×3 matrix [Dor20a],
8. Group, 21 perspective FullHD cameras placed on a sphere [Dor20b],

and 6 natural sequences, captured by real multicamera systems containing different numbers of perspective cameras (Fig. 4):

1. Fencing, 10 FullHD cameras placed on an arc [Dom16],
2. Carpark, 9 FullHD cameras placed linearly [Mie20],
3. Street, 9 FullHD cameras placed linearly [Mie20],
4. Hall, 9 FullHD cameras placed linearly [Mie20],
5. Frog, 13 FullHD cameras placed linearly [Sal18],
6. Painter, 16 cameras with resolution 2048×1088 placed in the 4×4 matrix [Doy17].

All sequences are commonly used in immersive video applications, e.g., within ISO/IEC JTC1/SC29/WG04 MPEG Video Coding group [MPEG21]. Each sequence has 17 frames.

Figure 4. Natural sequences. Left column: Frog, Hall, Painter; right: Fencing, Carpark, Street.

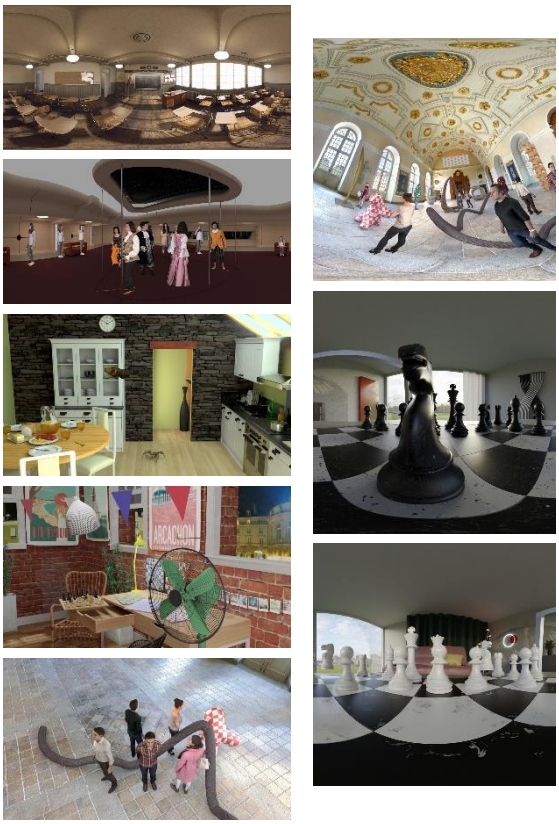


Figure 5. Computer-generated sequences. Left column: ClassroomVideo, Hijack, Kitchen, Fan, and Group; right: Museum, Chess, ChessPieces.

3. EXPERIMENTAL RESULTS

In total, 22 experiments were performed. In each, only one parameter was changed, while other parameters were set to default values. Each configuration was compared to the default configuration of the x265 encoder. Results of the encoding using default configuration are presented in Tables 2 and 3.

As mentioned earlier, all the results presented in Section 3 were averaged over 5 QP values and 14 sequences.

default configuration	A17		V17	
	bitrate [kbps]	PSNR [dB]	bitrate [kbps]	PSNR [dB]
avg G0	8583	69.86	8563	69.82
avg G1	10131	70.82	10174	70.76

Table 2. Encoding of depth atlases.

default configuration	A17		V17	
	bitrate [kbps]	PSNR [dB]	bitrate [kbps]	PSNR [dB]
avg T0	17781	42.36	17352	42.33
avg T1	10588	45.22	10531	45.26

Table 3. Encoding of texture atlases.

In consecutive subsections, only the most beneficial experiments are described. Results of all performed experiments are included at the end of the section.

Number of B-frames

In experiment #3, the influence of maximum number of consecutive B-frames was tested. By default, this number is set to 4. We have tested 16 B-frames, so the whole tested sequence was treated as one GOP.

bframes 16	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg G0	-0.22%	-0.06%	-0.21%	-0.07%
avg G1	0.42%	-0.07%	-0.21%	-0.05%

Table 4. Encoding of depth atlases, --bframes 16 (default: 4).

bframes 16	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg T0	-2.41%	-0.06%	-2.13%	-0.07%
avg T1	-0.54%	0.10%	-1.88%	-0.07%

Table 5. Encoding of texture atlases, --bframes 16 (default: 4).

The results obtained for texture and depth atlases are different – an increase of the number of B-frames allows to significantly decrease the bitrate of texture atlases, while for depth the difference is much smaller.

In the immersive video, the cameras are usually stationary, as the change of viewing position is made by the final viewer himself. Therefore, increasing the B-frames number allows to increase the coding efficiency of the video. This assumption is true both for textures and perfect (usually computer-generated) depth maps. The depth maps estimated for natural content are often not stable, thus a smaller number of B-frames is a better solution in this case.

Moreover, for the A17 configuration of the TMIV, the coding gain for atlases T1 and G1 is much smaller than for atlases T0 and G0. For V17 such a difference cannot be spotted.

The reason of this phenomenon is the different characteristics of the first and second atlas. The first atlas in the A17 configuration contains full base views while the second one contains many smaller patches, so it is much less temporally consistent. In the V17 configuration, both atlases contain similar information – full views.

Motion search method

In experiments #10, #11, and #12, different methods of motion search were tested. The default method in the x265 decoder is "hex" – hexagon-shaped search. We have tested three other methods: "dia" – diamond search, "umh" – uneven multi-hexagon, and "star" with 8 directions of searching. An exhaustive search ("full") was not tested because it is ridiculously slow and impractical.

Diamond search performs worse than the default hexagonal search (see Tables 20 and 21). More sophisticated search methods ("umh" and "star") allow to increase coding efficiency, especially for depth atlases (Tables 6 and 8).

For texture atlases, the coding gain is smaller but noticeable. It should be noted, that both search methods perform better for atlas T1 than for T0.

The "hex" method is adapted for analyzing the natural videos (i.e., textures), while it performs worse for depth maps. In uneven multi-hexagon and star methods, more prediction directions are being analyzed, allowing to better adapt to different characteristics of depth video (large smooth areas, sharp edges).

me umh	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg G0	-1.53%	0.03%	-1.57%	0.03%
avg G1	-1.59%	0.04%	-1.50%	0.04%

Table 6. Encoding of depth atlases, --me umh (default: hex).

me umh	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg T0	-0.14%	0.01%	-0.13%	0.01%
avg T1	-0.46%	0.05%	-0.10%	0.00%

Table 7. Encoding of texture atlases, --me umh (default: hex).

me star	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg G0	-2.78%	0.06%	-2.91%	0.05%
avg G1	-2.44%	0.07%	-2.79%	0.06%

Table 8. Encoding of depth atlases, --me star (default: hex).

me star	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg T0	-0.19%	0.01%	-0.18%	0.01%
avg T1	-0.62%	0.06%	-0.14%	0.01%

Table 9. Encoding of texture atlases, --me star (default: hex).

RDO level

In experiments #14, #15, and #16, the different levels of the rate-distortion optimization were tested. In x265, the level of the RDO may vary from 1 to 5, where 5 is the full RDO analysis and 1 is the least exhaustive optimization. By default, RDO level is set to 3.

We have tested three levels of RDO: 1, 2, and 5 (4 was omitted as it produces exactly the same results as 3).

Setting the RDO level to 1 or 2 speeds up the computations, but it greatly decreases the coding efficiency (see Tables 20 and 21).

The results of using full RDO are presented in Tables 10 and 11. For encoding of texture atlases, full RDO not only increases the computational time but also

decreases the coding efficiency. The opposite results were obtained for depth atlases, where full RDO allows to decrease the bitrate by more than 14%.

The video encoders are adapted for natural video encoding, while, as mentioned earlier, depth maps have significantly different characteristics. The encoding tools, including partial rate-distortion optimization, are optimized for texture encoding. When the full RDO process is performed, these texture-aimed simplifications are disabled.

rd5	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg G0	-14.06%	0.74%	-14.15%	0.70%
avg G1	-14.09%	0.94%	-13.83%	0.75%

Table 10. Encoding of depth atlases, --rd 5 (default: 3).

rd5	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg T0	1.64%	0.22%	1.21%	0.22%
avg T1	0.72%	0.12%	1.25%	0.23%

Table 11. Encoding of texture atlases, --rd 5 (default: 3).

Rectangular motion partitioning

In experiments #18 and #19, the influence of the rectangular motion partitioning was tested. By default, x265 uses square partitioning only. In order to keep the short computational time, the --limit-modes parameter was used. This parameter limits number of modes analyzed for each CU, significantly decreasing the computational time while having a slight impact on coding efficiency.

In #18, only the symmetric motion partitions were permitted, thus the block can be split into two N×2N or 2N×N blocks.

The results gathered in Tables 12 and 13 indicate, that enabling the rectangular motion partitioning allows increasing coding efficiency, both for depth and texture atlases. However, the bitrate reduction achieved for depth atlases is significantly higher.

As mentioned earlier, depth video contains large smooth regions that often include only constant depth values. These regions are separated by sharp edges and many of them are horizontal or vertical. Enabling of the rectangular partitioning allows to better fit the CU grid to objects in the depth video thus significantly increase the coding efficiency. For textures, rectangular partitioning also allows to increase the efficiency, but the gain is much smaller because of video characteristics (fewer smooth areas, highly textured objects, etc.).

In #19, also asymmetric partitioning was permitted, so each block can be split also in 25%/75% or 75%/25% proportion (both vertically and horizontally. As presented in Tables 14 and 15, it allows to additionally increase the coding efficiency, but the difference is slight.

rect	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg G0	-21.87%	0.54%	-22.44%	0.51%
avg G1	-22.71%	0.70%	-22.06%	0.53%

Table 12. Encoding of depth atlases, --rect --limit-modes.

rect	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg T0	-0.70%	0.05%	-0.75%	0.05%
avg T1	-1.39%	-0.12%	-0.72%	0.05%

Table 13. Encoding of texture atlases, --rect --limit-modes.

rect amp	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg G0	-21.94%	0.54%	-22.50%	0.52%
avg G1	-22.71%	0.70%	-22.11%	0.53%

Table 14. Encoding of depth atlases, --rect --amp --limit-modes.

rect amp	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg T0	-0.76%	0.06%	-0.83%	0.06%
avg T1	-1.54%	0.02%	-0.79%	0.06%

Table 15. Encoding of texture atlases, --rect --amp --limit-modes.

Max number of L0 references

In experiment #20, the maximum number of L0 references was tested. HEVC specification allows up to 8 references, the default value in x265 is set to 3. In the experiment, 4 L0 references were tested.

As shown in Tables 16 and 17, the coding gain is slight, but noticeable for all the data types. In general, it decreases the bitrate of the atlases without having any impact on the quality.

ref 4	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg G0	-0.17%	0.00%	-0.17%	0.00%
avg G1	-0.15%	0.00%	-0.14%	0.00%

Table 16. Encoding of depth atlases, --ref 4 (default: 3).

ref 4	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg T0	-0.08%	0.00%	-0.08%	0.00%
avg T1	-0.13%	0.01%	-0.07%	0.00%

Table 17. Encoding of texture atlases, --ref 4 (default: 3).

Subpixel refinement level

In experiment #21, the number of subpixel refinements was changed. By default, it is set to 2. We have tested value 3, which increases the number of half-pel iterations and allows using the chroma residual in the motion estimation decisions.

As presented in Table 18, such a change allows reducing bitrate of depth by 0.3% without quality change. For texture atlases, increasing the subpixel refinement level slightly increases the bitrate.

subme 3	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg G0	-0.32%	0.01%	-0.31%	0.01%
avg G1	-0.28%	0.01%	-0.27%	0.01%

Table 18. Encoding of depth atlases, --subme 3 (default: 2).

subme 3	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg T0	0.05%	0.00%	0.05%	0.00%
avg T1	0.20%	0.03%	0.06%	0.00%

Table 19. Encoding of texture atlases, --subme 3 (default: 2).

All tests

Parameter	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
b-adapt 0	1.12%	-0.04%	1.47%	-0.04%
b-adapt 1	2.06%	0.06%	2.43%	0.03%
bframes 16	0.13%	-0.07%	-0.21%	-0.06%
b-adapt 0 bframes 16	5.74%	-0.32%	5.96%	-0.30%
b-adapt 1 bframes 16	2.00%	0.05%	2.35%	0.02%
bframes bias	1.73%	-0.19%	1.26%	-0.19%
lookahead-slices 1	-0.01%	0.00%	0.00%	0.00%
lookahead-slices 4	-0.01%	0.00%	-0.01%	0.00%
max-merge	0.00%	0.00%	0.00%	0.00%
me dia	0.77%	-0.02%	1.05%	-0.02%
me umh	-1.56%	0.04%	-1.54%	0.03%
me star	-2.59%	0.06%	-2.85%	0.06%
no-early skip	0.37%	0.13%	0.58%	0.17%
rd 1	53.04%	-1.40%	50.88%	-1.24%
rd 2	20.01%	-0.89%	21.96%	-0.86%
rd 5	-14.08%	0.84%	-14.00%	0.73%
rdoq-level 2	1.60%	0.85%	1.87%	0.59%
rect limit-modes	-22.32%	0.62%	-22.27%	0.52%
rect amp limit-modes	-22.36%	0.62%	-22.32%	0.53%
ref 4	-0.16%	0.00%	-0.16%	0.00%
subme 3	-0.30%	0.01%	-0.29%	0.01%

Table 20. Encoding of depth atlases.

Other tested parameters decrease the coding efficiency, or their impact is negligible. In Tables 20 and 21, the results for all tested parameters are presented. To preserve the readability of Table 20, values for G0 and G1 were averaged. For the same reason, we have averaged values for T0 and T1 in Table 21.

Parameter	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
b-adapt 0	0.84%	-0.58%	0.70%	-0.03%
b-adapt 1	5.80%	0.23%	6.49%	0.06%
bframes 16	-1.71%	0.02%	-2.01%	-0.07%
b-adapt 0 bframes 16	3.75%	-0.54%	2.12%	-0.29%
b-adapt 1 bframes 16	5.78%	-0.14%	6.48%	0.06%
bframes bias	-0.50%	-0.21%	0.03%	-0.23%
lookahead-slices 1	0.00%	0.00%	0.00%	0.00%
lookahead-slices 4	-0.02%	0.00%	0.00%	0.00%
max-merge	0.00%	0.00%	0.00%	0.00%
me dia	0.03%	0.00%	0.01%	0.00%
me umh	-0.26%	0.03%	-0.12%	0.00%
me star	-0.35%	0.04%	-0.16%	0.01%
no-early skip	0.74%	0.17%	0.66%	0.17%
rd 1	14.36%	-0.69%	8.44%	-0.36%
rd 2	1.89%	-0.26%	0.44%	-0.08%
rd 5	1.30%	0.17%	1.23%	0.22%
rdoq-level 2	3.60%	-0.14%	3.04%	-0.38%
rect limit-modes	-0.96%	-0.04%	-0.73%	0.05%
rect amp limit-modes	-1.05%	0.04%	-0.81%	0.06%
ref 4	-0.10%	0.00%	-0.08%	0.00%
subme 3	0.11%	0.02%	0.05%	0.00%

Table 21. Encoding of texture atlases.

Optimal combination of parameters

The analysis of the results of 21 experiments allowed to find the set of optimal parameters for encoding the immersive video.

In the last, 22nd experiment, the encoder was configured in order to provide the best encoding efficiency. For depth atlases, we have used: --me star --rd 5 --rect --amp --limit-modes --ref 4 --subme 3. For texture encoding: --bframes 16 --me star --rect --amp --limit-modes --ref 4.

The results are presented in tables 22 and 23.

optimal configuration	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg G0	-27.57%	1.01%	-28.07%	0.96%
avg G1	-30.27%	1.26%	-27.50%	1.00%

Table 22. Encoding of depth atlases, --me star --rd 5 --rect --amp --limit-modes --ref 4 --subme 3.

optimal configuration	A17		V17	
	dBitrate	dPSNR	dBitrate	dPSNR
avg T0	-3.41%	0.01%	-3.15%	-0.01%
avg T1	-2.74%	0.11%	-2.86%	0.00%

Table 23. Encoding of texture atlases, --me star --bframes 16 --rect --amp --limit-modes --ref 4.

4. CONCLUSIONS

The paper presents the research on the influence of HEVC configuration on immersive video coding. MPEG Immersive Video is agnostic to the internally used codec, therefore, the proposed experiments tested a set of coding tools in order to find the optimal configuration, i.e., adapted to the characteristics of immersive video.

The experimental results showed that the configuration of the internal codec (one of the available HEVC implementations was tested – x265) has a significant impact on the overall coding efficiency, confirming the importance of performing the proposed optimization. The proper configuration allowed to reduce the bitstream on average by 3% for textures and even by 30% for depth maps encoding.

Moreover, the noticed differences in encoding of textures and depth maps draw interesting general conclusions that can be utilized not only in the coding of immersive videos but can be used in future research on coding adapted to typical characteristics of depth maps.

5. ACKNOWLEDGMENTS

This work was supported by the Ministry of Education and Science of Republic of Poland.

6. REFERENCES

[Boi18] Boissonade, P., and Jung, J. [MPEG-I Visual] Proposition of new sequences for Windowed-6DoF experiments on compression, synthesis, and depth estimation. ISO/IEC JTC1/SC29/WG11 MPEG, M43318, Ljubljana, Slovenia, 2018.

[Boy21] Boyce, J., Doré, R., Dziembowski, A., Fleureau, J., Jung, J., Kroon, B., Salahieh, B., Malamal Vadakital, V.K., and Yu, L. MPEG Immersive Video Coding Standard. Proceedings of the IEEE, pp. 1-16, Early Access, 2021.

[Dom16] Domański, M., Dziembowski, A., Grzelka, A., Łuczak, A., Mieloch, D., Stankiewicz, O., and Wegner, K. Multiview test video sequences for free navigation exploration obtained using pairs of cameras. ISO/IEC JTC1/SC29/WG11 MPEG, M38247, Geneva, Switzerland, 2016.

[Dor18] Doré, R. Technicolor 3DoF+ test materials. ISO/IEC JTC1/SC29/WG11 MPEG, M42349, San Diego, CA, USA, 04.2018.

[Dor20a] Doré, R., Briand, G., and Thudor, F. [MPEG-I Visual] InterdigitalFan0 content proposal for MIV. ISO/IEC JTC1/SC29/WG11 MPEG, M54732, Online, 07.2020.

[Dor20b] Doré, R., Briand, G., and Thudor, F. [MPEG-I Visual] InterdigitalGroup content proposal. ISO/IEC JTC1/SC29/WG11 MPEG, M54731, Online, 07.2020.

[Doy17] Doyen, D., Langlois, T., Vandame, B., Babon, F., Boisson, G., Sabater, N., Gendrot, R., and Schubert, A. Light field content from 16-camera rig. ISO/IEC JTC1/SC29/WG11 MPEG, M40010, Geneva, Switzerland, 2017.

[Gra20] Graziosi, D., Nakagami, O., Kuma, S., Zaghetto, A., Suzuki, T., and Tabatabai, A. An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC). APSIPA Transactions on Signal and Information Processing, 9, E13, 2020.

[Ilo19] Ilola, L., Vadakital, V.K.M., Roimela, K., and Keraenen, J. New test content for immersive video – Nokia Chess. ISO/IEC JTC1/SC29/WG11 MPEG, M50787, Geneva, Switzerland, 10.2019.

[Ilo20] Ilola, L., and Vadakital, V. [MPEG-I Visual] [MIV] Improved NokiaChess sequence. ISO/IEC JTC1/SC29/WG11 MPEG, M54382, 2020.

[Kro18] Kroon, B. 3DoF+ test sequence ClassroomVideo. ISO/IEC JTC1/SC29/WG11 MPEG, M42415, San Diego, CA, USA, 04.2018.

[Mea20] Meardi, G., Ferrara, S., Ciccarelli, L., Cobianchi, G., Poularakis, S., Maurer, F., Battista, S., and Byagowi, A. MPEG-5 Part 2: Low Complexity Enhancement Video Coding (LCEVC): Overview and performance evaluation. Proc. of SPIE 11510, Applications of Digital Image Processing XLIII, 115101C, 08.2020.

[Mie20] Mieloch, D., Dziembowski, A., and Domański, M. [MPEG-I Visual] Natural outdoor test sequences. ISO/IEC JTC1/SC29/WG11 MPEG, M51598, Brussels, Belgium, 01.2020.

[MPEG17] High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Improved Encoder Description Update 9, ISO/IEC JTC1/SC29 /WG11 MPEG/N17047, July 2017, Torino, Italy.

[MPEG21] Common Test Conditions for MPEG Immersive Video. ISO/IEC JTC1/SC29/WG04 MPEG VC, N0051, Online, Jan. 2021.

[MPEG21a] Test Model 8 for MPEG Immersive Video. ISO/IEC JTC1/SC29/WG04 MPEG VC, N0050, Online, Jan. 2021.

[Sal18] Salahieh, B., et al, J. Kermit test sequence for Windowed 6DoF Activities. ISO/IEC JTC1/SC29/WG11 MPEG, M43748, Ljubljana, Slovenia, 2018.

[Sod11] Sodagar, I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. IEEE MultiMedia, vol. 18, no. 4, pp. 62-67, April 2011.

[Sta18] Stankiewicz, O., Domański, M., Dziembowski, A., Grzelka, A., Mieloch, D., and Samelak, J. A Free-viewpoint Television system for horizontal virtual navigation. IEEE Transactions on Multimedia, vol. 20, no. 8, pp. 2182-2195, 2018.

[Sze20] Szekiełda, J., Stankowski, J., Dziembowski, A., and Mieloch, D. On the computational time of MIV-related experiments. ISO/IEC JTC1/SC29 /WG11 MPEG, M54897, Online, 10.2020.

[X265] MulticoreWare Inc. x265 HEVC Encoder/ H.265 Video Codec. Available on http://x265.org

Efficient Immersive Video Compression using Screen Content Coding

Jarosław Samelak, Adrian Dziembowski, Dawid Mieloch, Marek Domański, Maciej Wawrzyniak
adrian.dziembowski@put.poznan.pl
Poznań University of Technology, Poznań, Poland
Institute of Multimedia Telecommunications, Polanka 3, 61-131

ABSTRACT

The paper deals with efficient compression of immersive video representations for the synthesis of video related to virtual viewports, i.e., to selected virtual viewer positions and selected virtual directions of watching. The goal is to obtain possibly high quality of virtual video obtained from compressed representations of immersive video acquired from multiple omnidirectional and planar (perspective) cameras, or from computer animation. In the paper, we describe a solution based on HEVC (High Efficiency Video Coding) compression and the recently proposed MPEG Test Model for Immersive Video. The idea is to use standard-compliant Screen Content Coding tools that were proposed for other applications and have never been used for immersive video compression. The experimental results with standard test video sequences are reported for the normalized experimental conditions defined by MPEG. In the paper, it is demonstrated that the proposed solution yields up to 20% of bitrate reduction for the constant quality of virtual video.

Keywords

Video compression, video codecs, virtual reality.

1 INTRODUCTION

The recent development of virtual reality applications raises rapidly growing research interests in immersive video [Isg14]. In particular, substantial efforts are made in virtual view synthesis [Ceu18], [Yua18], [Rah18], [Zhu19], virtual navigation and free-viewpoint television [Tan12], [Sta18], [Cha19]. Recently, image-based rendering of virtual views became widely applicable for head-mounted devices and other displays suitable for VR content. The content may be computer-generated or it may be acquired from multiple omnidirectional and perspective (planar) cameras. Such visual content constitutes an immersive video that may have various representations. Recently, great interest is attained by point clouds [Cui19], [Zha20], [Li20], [Sch19], but the representation that is most often used in research is multiview video plus depth (MVD) [Mue11]. Therefore, this paper is focused on multiview video plus depth representations of immersive video. For such representations, depth has to be estimated, and a lot of work has already been done for depth estimation in the above-mentioned applications, e.g. [Mie20]. Once the representation is estimated, the representation of immersive video needs to be compressed before transmission (cf. Fig. 1).

Obviously, the compression artifacts deteriorate the fidelity of view synthesis. Therefore, in the paper, we consider immersive video compression and the influence of the compression on the quality of the virtual video rendered from compressed data. Moreover, we propose an alternative approach to immersive video compression, and we demonstrate the advantages of this alternative approach. In particular, we demonstrate that our approach results in a reduced bitrate for the same quality of virtual views, i.e., for a constant bitrate, the proposed approach results in the improved quality of the synthesized virtual views as compared to the approaches from [Dom19], [Fle19], [Laf19], and [Wie19].

2 IMMERSIVE VIDEO COMPRESSION

A multiview representation of immersive video may consist of multiple perspective (planar, 2D) views with vastly overlapping fields of views, or it may consist of a few overlapping 360-degree videos. The compression of immersive video takes advantage of the inter-view

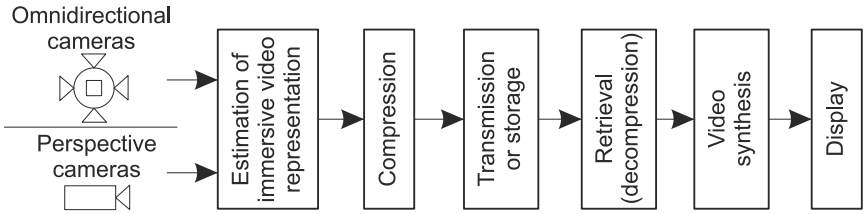


Figure 1: Data flow in immersive video systems.

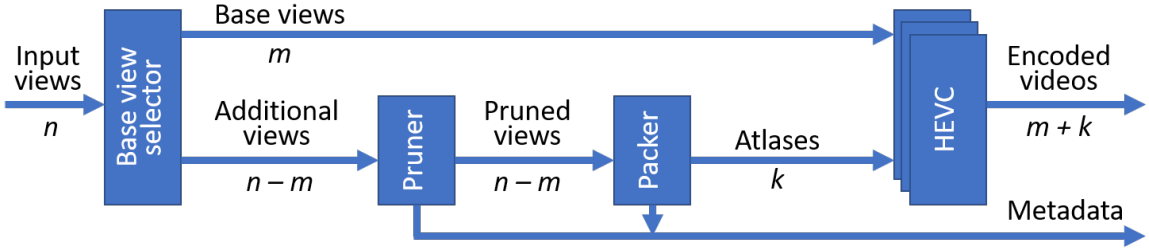


Figure 2: MPEG immersive video encoder (TMIV framework).

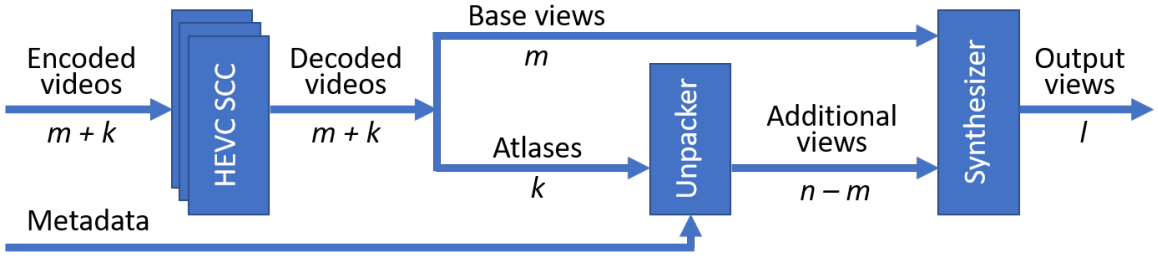


Figure 3: MPEG immersive video decoder (TMIV framework).

redundancy existing in the input multiview representation. Removal of this redundancy will result in decreasing the amount of data required to fully represent the whole three-dimensional scene.

One of the possible scenarios assumes the compression of MVD representation using a standard 3D-HEVC video encoder. Its coding techniques use inter-view prediction based on depth maps and statistical dependencies between views and corresponding depth maps. The use of this encoder reduces the required bitrate by up to 50% in comparison with simulcast HEVC [Tec16], which encodes each view and each depth map separately. Other works focus more on the reduction of pixel rate, i.e., the number of pixels that have to be sent during the transmission. An interesting technique described in [Gar19] proposes a decoder-side reconstruction of depth maps using views compressed using simulcast HEVC or MV-HEVC. This solution provides a 50% reduction of the pixel rate (because depth maps do not have to be sent) and up to a 35% reduction of the required bitrate while preserving similar quality of the video.

The state-of-the-art technology for immersive video compression is being developed by ISO/IEC MPEG group [ISO19e]. The MPEG Test Model for Immersive Video (TMIV) is already publicly available as a descriptive and software framework for research [ISO19d], and in the next months, the works on this future video standard are planned to enter one of the final stages of preparation.

The forthcoming standard is built using the technologies presented by proponents in response to the Call for Proposals for 3DoF+ video coding [ISO19a]. Some proposals followed nearly the same basic idea that several base views gathering most of the information of the scene should be encoded in their entirety, while supplementary information (e.g., disocclusions from other views, Fig. 4) can be transmitted in the form of a mosaic of much smaller patches, that all together are grouped into atlases [Dom19], [Fle19]. The main idea of TMIV follows a similar scheme – see Fig. 2 and Fig. 3 for the overview.

First of all, n input views with depth maps are split into two groups: m base views and n-m additional views. The pruner (cf. Fig. 4), basing on depth, identifies and extracts regions occluded in the base views. These occluded regions are left in additional views, while the rest of the regions are removed. It results in small patches left in the pruned additional views. The packer gathers patches from all additional views into k at-

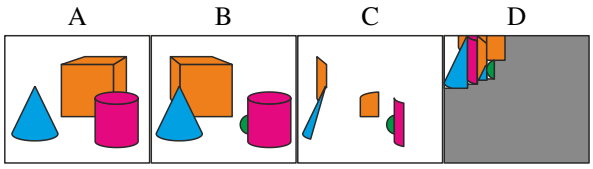


Figure 4: a) Base view, b) additional view, c) pruned additional view (preserved disocclusions), d) atlas.

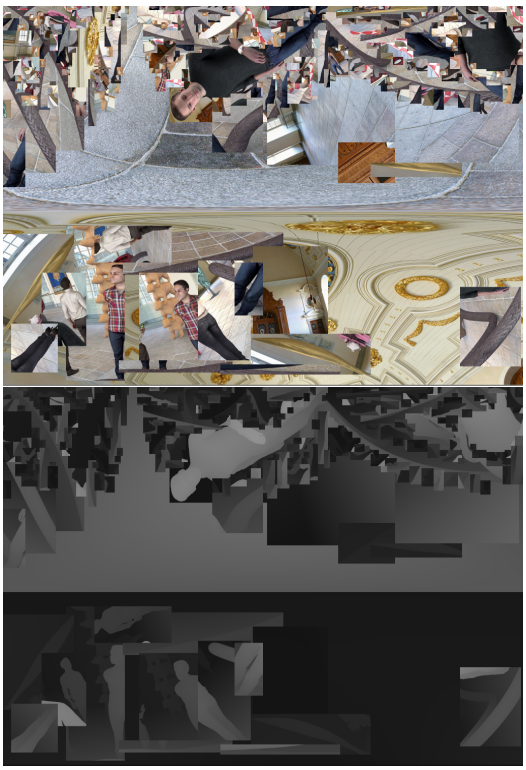


Figure 5: Example of an atlas with a corresponding depth map.

lases. In order to provide better encoding efficiency, the patches in atlases contain all information from their bounding box, as this decreases the number of sharp edges in the encoded atlas. A schematic example of pruning and packing is presented in Fig. 4. For example, an atlas for the TechnicolorMuseum [Dor18] test sequence is presented in Fig. 5. The number of atlases is usually much smaller than the number of additional views, ensuring the reduction of pixel rate, while still preserving the whole representation of the encoded three-dimensional scene. In the end, the base views and atlases are fed to simulcast HEVC encoders.

In the decoder, base views and patches from atlases, together with metadata that contain the initial positions of patches in input views, are used to synthesize l output views, which can be reconstructed input views, or any number of virtual views required by a user of the immersive video system (e.g., a stereopair for a virtual reality headset).

The development of an entirely new competitive compression technology would require huge costs and manpower. Therefore, in order to reduce the time needed to finish the works on the new coding technology and standard, the current approach is to build the immersive video coding technology on top of the general video compression technology [Dom19], [Fle19], [ISO19e].

The common feature of the above-mentioned coding technologies is the use of virtual view synthesis and the application of general video coding techniques like HEVC or even the application of 3D-HEVC that is the specialized coding technology for multiview plus depth video. In the following section, we propose the application of HEVC Screen Content Coding [Xu16b], the technique for computer-generated visual content, in order to increase the quality of virtual view synthesis performed on the compressed representation of the immersive video.

3 NEW APPROACH TO COMPRESSION OF PATCH ATLASES

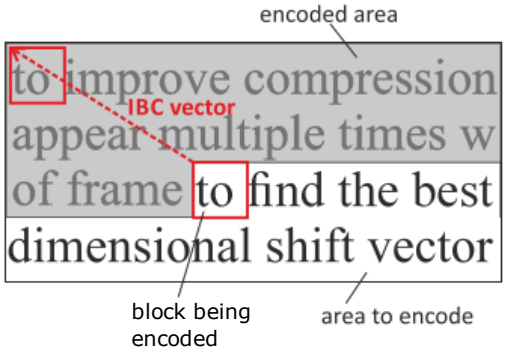


Figure 6: Operation of Intra Block Copy.

As mentioned before, for the efficient compression of patch atlases, the authors propose to use HEVC Screen Content Coding [Xu16b] instead of a standard video coding technology like HEVC [ISO15] or 3D-HEVC [Tec16]. Screen Content Coding is developed as an extension of HEVC, dedicated for the compression of computer-generated visual content, such as a remote keyboard, screen recordings or cloud gaming.

The basic tool used in HEVC-SCC is Intra Block Copy [Xu16a]. It is designed to improve the compression efficiency of fonts and other repetitive patterns that may appear multiple times within a single frame (cf. Fig. 6). The IBC tool searches the encoded part of the frame in order to find the best match for the unit being currently encoded. This search results in a two-dimensional shift vector with the components being integer multiples of the sampling periods (i.e., the horizontal and vertical sampling periods).

The idea to apply Intra Block Copy to the compression of camera-captured content was presented in [Sam17]

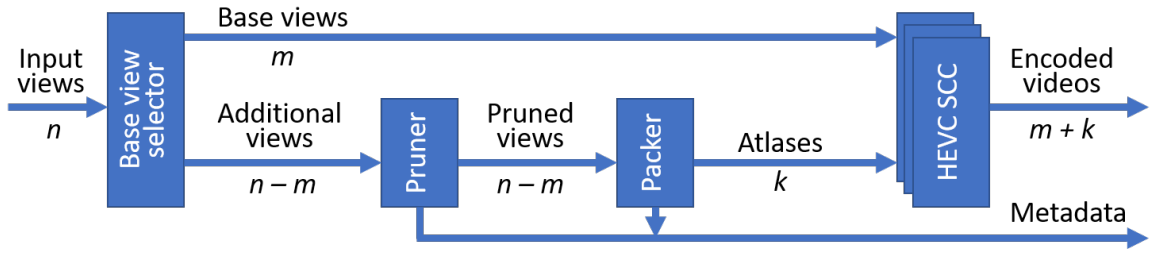


Figure 7: Proposed MPEG immersive video encoder with HEVC Screen Content Coding.

Sequence	Content type	Number of base views	Number of atlases
Classroom [Kro18]	O, CG	1	1
Museum [Dor18]	O, CG	2	2
Hijack [Dor18]	O, CG	1	2
Kitchen [Boi18]	P, CG	1	2
Painter [Doy17]	P, NC	1	4
Frog [Sal19]	P, NC	2	8
Fencing [Dom16]	P, NC	1	3

Table 1: Test sequences. O – omnidirectional, P – perspective, CG – computer generated, NC – natural content.

and [Sam19]. It was proven that IBC can be successfully used to exploit inter-view similarities in frame-compatible stereoscopic videos. The authors now propose to extend this idea onto the compression of patch atlases (Fig. 7). A single atlas often contains similar patches, located in distant parts of a frame. The IBC tool would be an ideal solution for efficient compression in such a case.

Other arguments in favor of using HEVC-SCC for the compression of patch atlases are additional SCC tools – Color Transform [Xu16b] and Palette Mode [Xu16c]. As presented in [Sam17], the influence of these tools on the compression efficiency of camera-captured content is negligible, however, they may provide a significant gain when applied to the compression of depth patch atlases. The results of using HEVC-SCC instead of HEVC are presented in the following section.

4 EXPERIMENTS AND RESULTS

4.1 Methodology of the experiments

The goal of the experiments is to demonstrate the usefulness and efficiency of the standard-compliant Screen Content Coding HEVC extension applied in immersive video coding. In order to present the advantages of such an approach, the recent MPEG Immersive Video encoder – TMIV [ISO19d] is used. The video data generated by TMIV is then encoded using HEVC-SCC. The results are compared to those obtained by the use of HEVC main profile.

The proposed approach is assessed using 7 miscellaneous test video sequences as described in Table 1. These sequences are commonly used in research and standardization activities on immersive video [ISO19b] because of their very diversified characteristics (natural and computer-generated content, omnidirectional and perspective cameras, different resolutions, etc.). For each sequence, 97 frames are used, which refers to 3 full groups of pictures (GOPs).

All common coding parameters (e.g. GOP size, Intra Period, max CU Width, Sample Addaptive Offset, etc.) are exactly the same for both encoders and the same as defined in MPEG recommendations for experiments on immersive video coding [ISO19b], [Yu15]. The same values of QP (Quantization Parameter) are set for both encoders: HEVC and HEVC-SCC. The ΔQP between depth and texture data is set to 10 in order to better preserve depth quality (e.g. when QP for texture was set to 22, QP for corresponding depth was set to 12; experiments were performed for 5 QP values – for texture: 22, 27, 32, 37, and 42), which is crucial for proper view synthesis.

In Section 4.2, the results of encoding of atlases are presented. For each sequence, the bitrate was calculated as a sum of bitrates for all atlases.

The quality (the average difference between atlases before and after encoding) was calculated as the average PSNR of all atlases. The texture and depth atlases are discussed separately.

In Section 4.3 the results of the virtual view synthesis are discussed. For each sequence, the bitrate is calculated as a sum of bitrates for all atlases, including both depth and texture.

The quality of synthesized views was measured using 5 objective quality metrics, which are commonly used in immersive video applications: Weighted-to-Spherically-Uniform PSNR (WS-PSNR) [Sun17], Multi-Scale SSIM (MS-SSIM) [Wan03], Visual Information Fidelity (VIF) [She06], Video Multimethod Assessment Fusion (VMAF) [Li16] and ISO/IEC MPEG's metric for immersive video: IVPSNR [ISO19c].

All used metrics are full-reference ones, therefore in order to estimate quality, the virtual views in positions

Sequence	Bitrate reduction Base view	Atlas	Quality improvement Base view	Atlas
Classroom	0.78%	1.74%	0.02 dB	0.04 dB
Museum	1.16%	4.00%	0.01 dB	-0.01 dB
Hijack	0.00%	6.55%	0.03 dB	0.22 dB
Kitchen	0.49%	7.44%	0.05 dB	0.11 dB
CG	0.61%	4.93%	0.03 dB	0.09 dB
Painter	-0.09%	1.53%	-0.01 dB	22.86 dB
Frog	0.44%	1.39%	0.00 dB	33.00 dB
Fencing	0.30%	0.69%	0.00 dB	0.02 dB
NC	0.22%	1.20%	0.00 dB	18.63 dB
Average	0.44%	3.33%	0.01 dB	8.03 dB

Table 2: Bitrate reduction and quality improvement for the use of HEVC Screen Content Coding tools instead of the plain HEVC for the base views and atlases. A positive number denotes bitrate reduction or quality index increase for the synthesized videos due to the usage of SCC.

of input views were synthesized using decoded video data. Then, the estimated quality was averaged over all views.

In order to calculate the difference between two encoding approaches, the Bjoentegaard Delta [Bjo01] metric was used.

4.2 Efficiency of immersive video coding using HEVC-SCC

In the proposed approach, all videos, i.e., base views, atlases, and corresponding depth maps, are being independently encoded using HEVC-SCC. Therefore, it was possible to split the encoding results depending on the data type.

In Figs. 8 and 9, the rate-distortion curves for views only (excluding depth) are presented. In general, the usage of HEVC-SCC allows to achieve better quality at the same bitrate when compared to the HEVC main profile.

At this point, it has to be mentioned why the quality of the TechnicolorPainter and IntelFrog sequences is astonishingly high. Actually, the PSNR value presented in Fig. 9 was averaged over all encoded base views and atlases. While there were no issues for base views, some of the atlases contain no patches within one or more group of pictures (e.g., within the third GOP of the IntelFrog sequence, where there are fewer occlusions than for the first two GOPs, 5 of 8 atlases are empty thus completely grey).

As discussed in Section 3, HEVC-SCC should perform better on atlases than on base views. Indeed, as the results presented in Table 2 show, the bitrate reduction

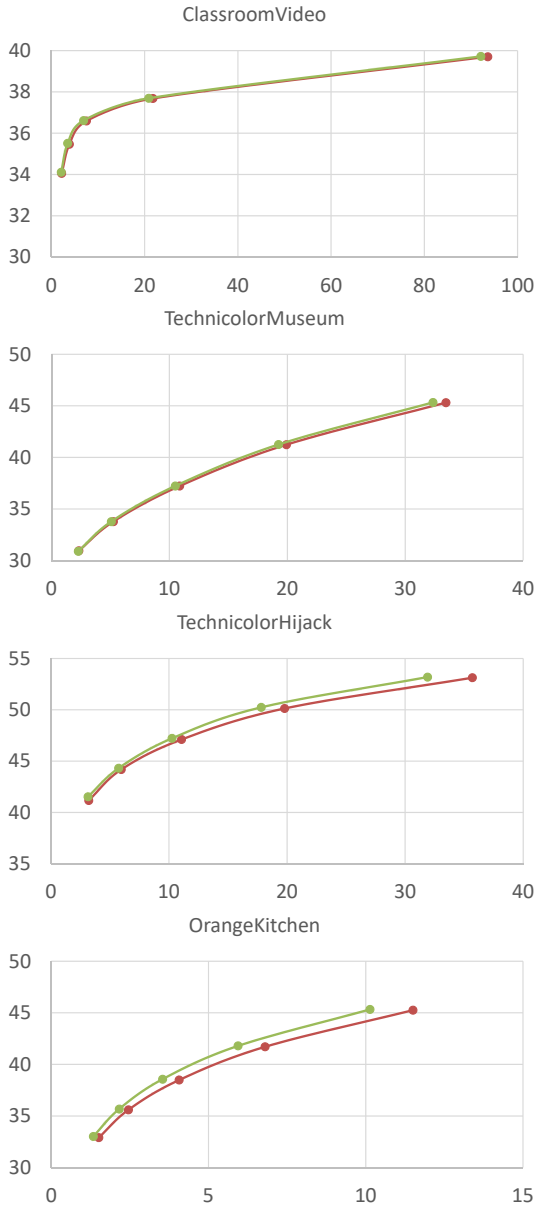


Figure 8: Rate-distortion curves for the immersive video codecs with the HEVC-SCC as compared to the plain HEVC: computer-generated sequences, input views encoding; red: HEVC, green: HEVC-SCC. Vertical axis: PSNR [dB], horizontal: bitrate [Mbps].

caused by using HEVC-SCC instead of HEVC is significantly higher for atlases than for base views. In general, also the quality improvement is bigger for atlases, however, the difference between HEVC and HEVC-SCC is really slight (except for the TechnicolorPainter and IntelFrog sequences, where HEVC-SCC performs much better for their almost empty atlases).

The second type of data being encoded is depth maps. The RD curves for depth are presented in Figs. 10 and 11. Compared to the encoding of input views, the en-

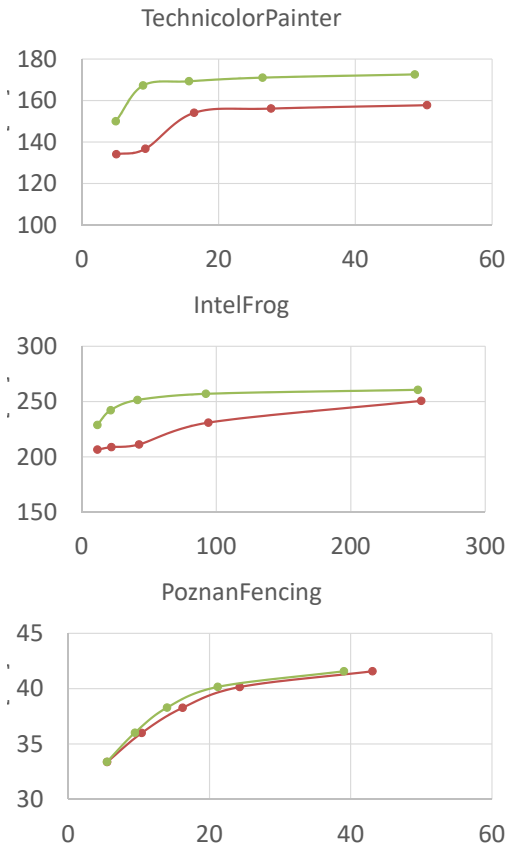


Figure 9: Rate-distortion curves for the immersive video codecs with the HEVC-SCC as compared to the plain HEVC: natural sequences, input views encoding; red: HEVC, green: HEVC-SCC. Vertical axis: PSNR [dB], horizontal: bitrate [Mbps].

coding gain in depth maps caused by the application of the SCC extension of HEVC is significantly higher. For all test sequences, HEVC-SCC allows for achieving a significantly better quality of depth maps, while preserving the same bitrates.

Such results are highly expected because of the characteristics of depth maps which contain mostly no texture, but large, smooth, semi-repeatable regions which can be efficiently encoded using SCC tools.

The efficiency of HEVC-SCC for base views and atlases is compared in Table 3. While for the input views encoding results were similar for natural and computer-generated sequences, the results for depth encoding are different for both sequence types. For computer-generated sequences, HEVC-SCC performs significantly better for atlases than for base views. However, for natural sequences, there is no significant difference between both types of data. The reason is the quality of depth maps, since for computer-generated sequences the depth is smooth within the objects' interior and sharp at their edges, while depth maps for natural content were algorithmically estimated based

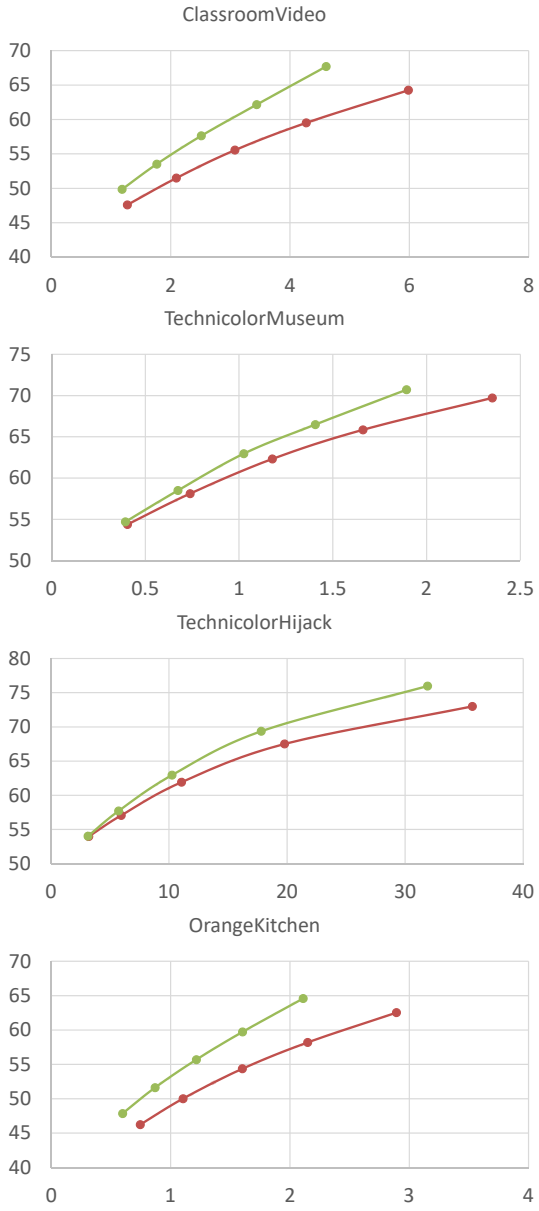


Figure 10: Rate-distortion curves for immersive video codecs with HEVC-SCC as compared to plain HEVC: computer-generated sequences, depth maps encoding; red: HEVC, green: HEVC-SCC. Vertical axis: PSNR [dB], horizontal: bitrate [Mbps].

on input views, therefore, they contain artifacts, such as blurred edges or grained objects. As a result, the atlases contain many small, different patches that negatively influence the HEVC-SCC encoding efficiency.

However, despite the problems described above, for depth data, HEVC-SCC performs much better than plain HEVC (even for natural sequences), helping reduce the bitrates and slightly increase the quality of decoded views.

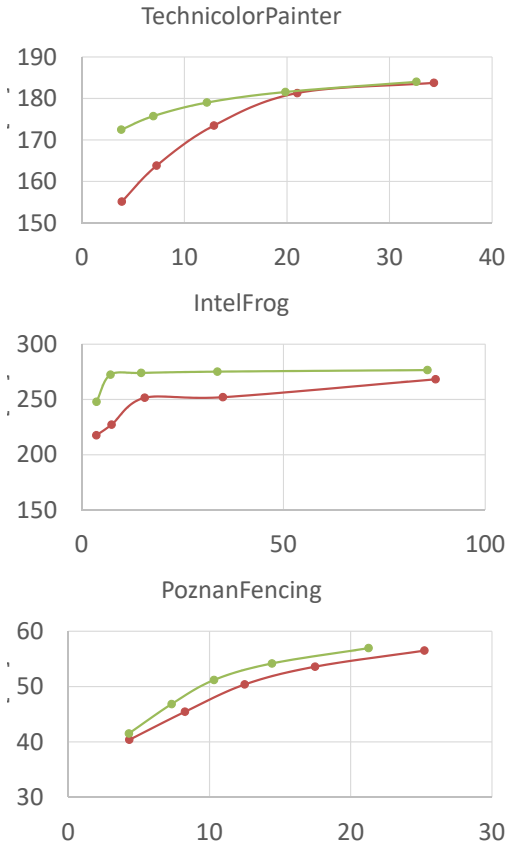


Figure 11: Rate-distortion curves for immersive video codecs with HEVC-SCC as compared to plain HEVC: natural sequences, depth maps encoding; red: HEVC, green: HEVC-SCC. Vertical axis: PSNR [dB], horizontal: bitrate [Mbps].

Sequence	Bitrate reduction Base view	Atlas	Quality improvement Base view	Atlas
Classroom	11.76%	18.38%	1.85 dB	3.14 dB
Museum	8.52%	13.12%	0.56 dB	0.62 dB
Hijack	7.89%	9.25%	1.09 dB	1.43 dB
Kitchen	15.34%	25.10%	1.28 dB	1.80 dB
CG	10.88%	16.47%	1.20 dB	1.75 dB
Painter	4.60%	4.27%	0.27 dB	8.77 dB
Frog	2.01%	3.16%	0.12 dB	32.32 dB
Fencing	14.23%	12.22%	0.90 dB	0.87 dB
NC	6.95%	6.55%	0.43 dB	13.99 dB
Average	9.19%	12.22%	0.87 dB	6.99 dB

Table 3: Bitrate reduction and quality improvement (compared to the HEVC main profile) for base views and atlases, depth data.

Sequence	WSPSNR	VIF	VMAF	SSIM	IVPSNR
Classroom	22.36%	10.87%	16.42%	10.28%	10.24%
Museum	7.82%	4.07%	8.77%	4.48%	4.78%
Hijack	19.83%	14.91%	20.06%	15.88%	9.20%
Kitchen	22.64%	16.35%	30.19%	16.30%	5.65%
CG	18.16%	11.55%	18.86%	11.74%	7.47%
Painter	3.37%	3.75%	2.92%	3.37%	3.33%
Frog	3.85%	2.70%	5.04%	4.14%	1.48%
Fencing	11.41%	11.18%	10.23%	10.31%	9.46%
NC	6.21%	5.88%	6.06%	5.94%	4.76%
Average	13.04%	9.12%	13.38%	9.25%	6.31%

Table 4: BD-rate reduction.

4.3 Rendered video quality from compressed data using the standard and proposed approaches

As presented in the previous section, HEVC-SCC allows for decreasing the total bitrate of immersive video data. However, the user of the immersive video system is not concerned about the quality of atlases or corresponding depth maps but pays attention to the final quality of the video he or she is watching. Therefore, in this section, the quality of synthesized virtual views is considered.

In Figs. 12 and 13 the RD-curves for synthesized virtual views are presented. On the horizontal axis, the total bitrate (base views + depth maps and atlases + depth maps) is presented, on the vertical one – the average value of WS-PSNR for luma component of synthesized video. As presented, the proposed approach allows for increasing the quality of synthesized views (compared to HEVC main profile) while preserving the total bitrate.

For each sequence, the average bitrate reduction (Bjoentegaard Delta – BD) between two curves was also estimated. The BD-rate measures the average bitrate change. The same calculations are performed also for 4 other, commonly-used quality metrics. All these values are gathered in Table 4.

As presented, HEVC-SCC performs better for computer-generated sequences. The encoding efficiency for natural sequences is lower, however, even for that type of content, HEVC-SCC works better than HEVC main profile.

In Fig. 14 fragments of virtual views synthesized using data compressed by two encoders are compared with fragments of input views. Note shifted and ragged edges generated by HEVC main (at the middle column).

In general, HEVC-SCC clearly outperforms plain HEVC for all the test sequences and all calculated quality metrics. Therefore, HEVC-SCC is a good choice for immersive video coding.

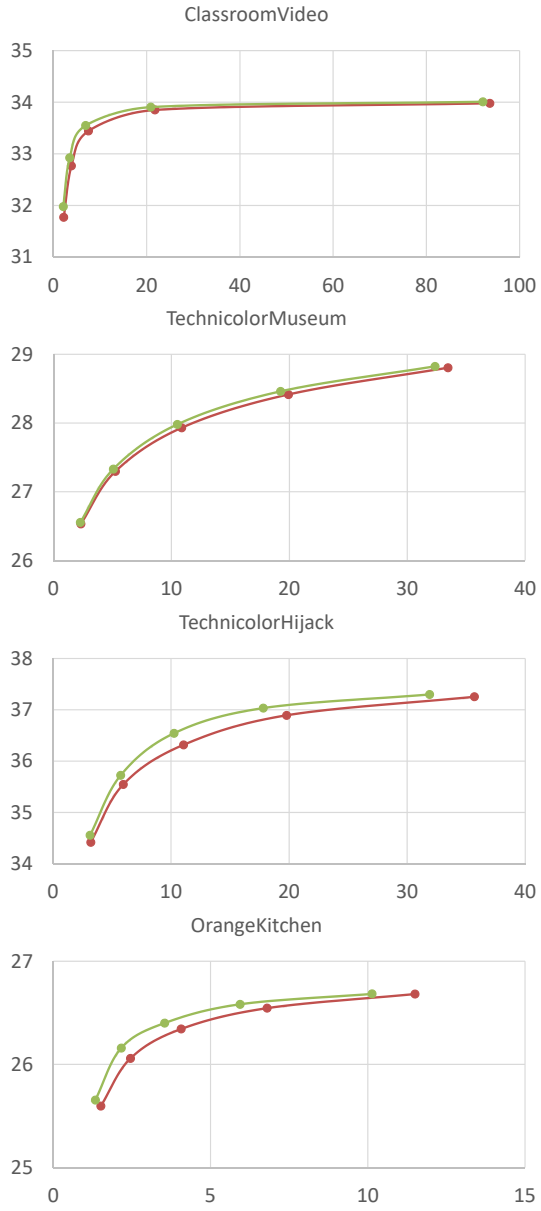


Figure 12: Rate-distortion curves for video synthesis from immersive video codecs with HEVC-SCC as compared to HEVC main profile: computer-generated sequences; red: HEVC, green: HEVC-SCC. Vertical axis: PSNR [dB], horizontal: bitrate [Mbps].

5 CONCLUSIONS

Immersive Video Coding is a new compression technology that is currently in the process of well-advanced standardization. The technology provides a solution for the generation of video sequences and parameters that represent immersive video. The video sequences may be then compressed using standard video coding techniques. In the process of development of this technology and its standardisation, HEVC coding was considered along with some experiments with VVC (Versatile

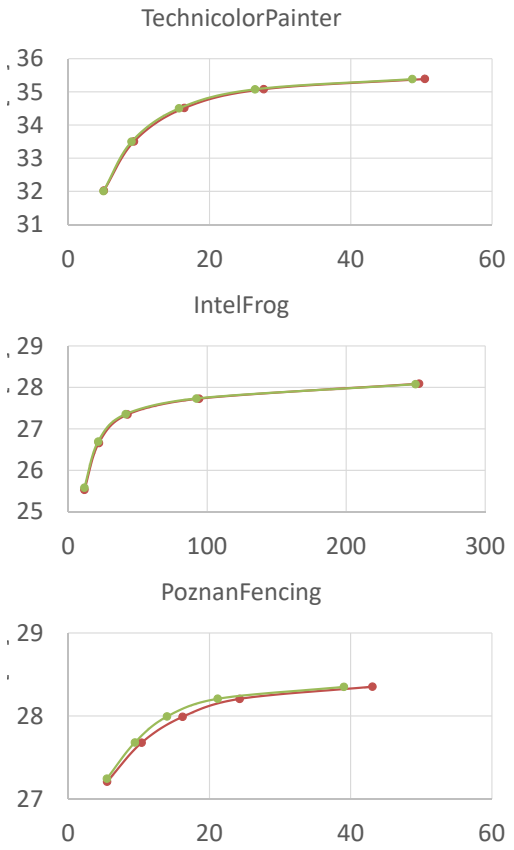


Figure 13: Rate-distortion curves for video synthesis from immersive video codecs with HEVC-SCC as compared to HEVC main profile: natural sequences; red: HEVC, green: HEVC-SCC. Vertical axis: PSNR [dB], horizontal: bitrate [Mbps].

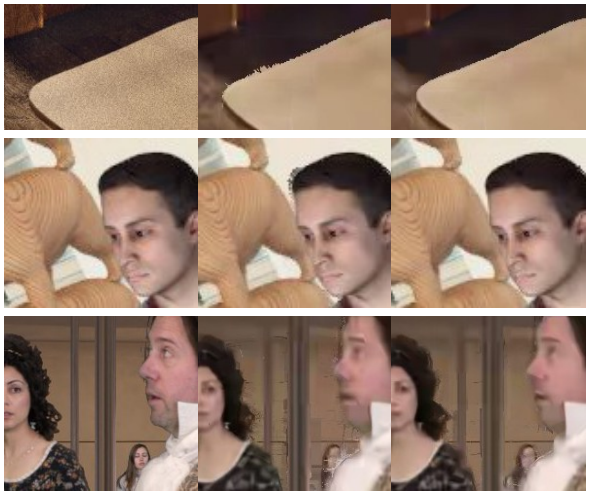


Figure 14: Fragments of: input views (left), views synthesized using data encoded using HEVC main profile (middle) and views synthesized using data encoded using HEVC-SCC (right). From top: ClassroomVideo, TechnicolorMuseum, TechnicolorHijack.

Video Coding) [Che20]. For Immersive Video Coding, both technologies proved to be useful and efficient according to the results obtained by MPEG. In this paper, we demonstrate that the overall rate-distortion coding efficiency may be even further improved by the use of the standard HEVC-SCC (HEVC Screen Content Coding). The proposed usage of HEVC-SCC requires no modification of the current draft for the standard on Immersive Video Coding [ISO20].

The novelty of the paper also consists in the application of the Screen Content Coding (SCC) technique for the compression of atlases that represent the immersive video. It is a new use of Screen Content Coding that was developed for completely other applications, i.e., with the aim to compress computer-generated images, like those transmitted to remote screens. This technique with the Intra Block Copy tool was never meant as a tool for the compression of immersive video content, in particular, natural immersive content acquired using cameras. The abovementioned application of Screen Content Coding was never described in the references. To our best knowledge, such an application is described for the first time in this paper.

In the paper, the application of Screen Content Coding to immersive video compression is experimentally tested in the framework of the Test Model for Immersive Video [ISO19d] that was recently developed by MPEG as a framework for the forthcoming international standard of immersive video compression [ISO20]. Currently, in the immersive video community, the research is executed using HEVC or 3D-HEVC codecs within the Test Model for Immersive Video. The idea of the paper is to replace HEVC or 3D-HEVC by another standard profile of the HEVC video codec, i.e., HEVC-SCC. It is worth to underline that the application of SCC (like HEVC-SCC) does not interfere with the general structure of the Test Model proposed for the future standard. For the standard test video sequences and the normalized experimental conditions used in the research on immersive video coding, the experimental data demonstrate that the application of HEVC-SCC is significantly more efficient than the traditional application of HEVC or 3D HEVC codecs for the compression of atlases representing the immersive video. This is clearly demonstrated for all MPEG test immersive video sequences available together with their reference data.

The quality improvement of the virtual views corresponds to the bitrate reduction of up to 20%. This quite a high value if we keep in mind that the whole HEVC technology has brought about 50% of the bitrate reduction. The experimental data (cf. Section 4.2) indicate that the main improvement yielded by the application of SCC is related to the higher fidelity of the decoded depth maps, and it is well-known that the fidelity of depth significantly influences the quality of the virtual views. For the approach proposed, the quality improvement of virtual views is also higher for computer-generated content than for natural content acquired from cameras.

6 ACKNOWLEDGMENTS

The research was supported by the Ministry of Education and Science of Republic of Poland.

7 REFERENCES

[Bjo01] G. Bjoentegaard. Calculation of average PSNR differences between RD-Curves. ITU-T VCEG Meeting, Austin, USA, 2001.

[Boi18] P. Boissonade and J. Jung. [MPEG-I Visual] Proposition of new sequences for Windowed-6DoF experiments on compression, synthesis, and depth estimation. ISO/IEC JTC1/SC29/WG11 MPEG/M43318, Ljubljana, Slovenia, 2018.

[Ceu18] B. Ceulemans et al. Robust Multiview Synthesis for Wide-Baseline Camera Arrays. IEEE Tr. on Multimedia, 2018.

[Cha19] J. Chakareski. UAV-IoT for next generation virtual reality. IEEE Tr. on Image Proc., 2019.

[Che20] J. Chen et al. The Joint Exploration Model (JEM) for Video Compression With Capability Beyond HEVC. IEEE Tr. on Circuits and Systems for Video Technology, 2020.

[Cui19] L. Cui et al. Point-Cloud Compression: Moving Picture Experts Group's New Standard in 2020. IEEE Cons. Electronics Magazine, 2019.

[Dom16] M. Domański et al. Multiview test video sequences for free navigation exploration obtained using pairs of cameras. ISO/IEC JTC1/SC29/WG11/M38247, Geneva, 2016.

[Dom19] M. Domański et al. Technical description of proposal for Call for Proposals on 3DoF+ Visual prepared by PUT and ETRI. ISO/IEC JTC1/SC29/WG11/M47407, Geneva, 2019.

[Dor18] R. Doré. Technicolor 3DoF+ test materials. ISO/IEC JTC1/SC29/WG11 MPEG/M42349, San Diego, USA, 2018.

[Doy17] D. Doyen et al. Light field content from 16-camera rig. ISO/IEC JTC1/SC29/WG11 MPEG, M40010, Geneva, Switzerland, 2017.

[Fle19] J. Fleureau et al. Technicolor-Intel Response to 3DoF+ CfP. ISO/IEC JTC1/SC29/WG11 MPEG/M47445, Geneva, Switzerland, 2019.

[Gar19] P. Garus et al. Bypassing Depth Maps Transmission For Immersive Video Coding. 2019 Picture Coding Symposium (PCS), 2019.

[Isg14] F. Isgro et al. Three-dimensional image processing in the future of immersive media. IEEE Tr. on Circuits and Systems for Video Tech., 2014.

[ISO15] ISO/IEC. High efficiency coding and media delivery in heterogeneous environment – Part 2: High efficiency video coding. ISO/IEC Int. Standard 23008-2, 2015.

[ISO19a] ISO/IEC MPEG. Call for Proposals on 3DoF+ Visual. ISO/IEC JTC1/SC29/WG11 MPEG/N18145, Marrakech, 2019.

[ISO19b] ISO/IEC MPEG. Common Test Conditions for Immersive Video. ISO/IEC JTC1/SC29/WG11/N18789, Geneva, 2019.

[ISO19c] ISO/IEC MPEG. Software manual of IV-PSNR for Immersive Video. ISO/IEC JTC1/SC29/WG11/N18709, Goeteborg, 2019.

[ISO19d] ISO/IEC MPEG. Test Model 3 for Immersive Video. ISO/IEC JTC1/SC29/WG11 MPEG/N18795, Geneva, Switzerland, 2019.

[ISO19e] ISO/IEC MPEG. Working Draft 3 of Immersive Video. ISO/IEC JTC1/SC29/WG11 MPEG/N18794, Geneva, Switzerland, 2019.

[ISO20] ISO/IEC MPEG. Text of ISO/IEC CD 23090-12 MPEG Immersive Video. ISO/IEC JTC1/SC29/WG11/N19482, Online, 2020.

[Kro18] B. Kroon. 3DoF+ test sequence ClassroomVideo. ISO/IEC JTC1/SC29/WG11 MPEG/M42415, San Diego, USA, 2018.

[Laf19] G. Lafruit et al. Understanding MPEG-I Coding Standardization in Immersive VR/AR Applications. SMPTE Motion Imaging Journal, 2019.

[Li16] Z. Li et al. Toward a practical perceptual video quality metric. Netflix Technology Blog, 2016.

[Li20] L. Li et al. Advanced 3D Motion Prediction for Video-Based Dynamic Point Cloud Compression. IEEE Tr. on Image Processing, 2020.

[Mie20] D. Mieloch et al. Depth Map Estimation for Free-Viewpoint Television and Virtual Navigation. IEEE Access, 2020.

[Mue11] K. Mueller et al. 3-D video representation using depth maps. Proc. of the IEEE, 2011.

[Rah18] D. Rahaman and M. Paul. Virtual view synthesis for free viewpoint video and multiview video compression using Gaussian mixture modelling. IEEE Tr. on Image Processing, 2018.

[Sal19] B. Salahieh et al. Kermit test sequence for Windowed 6DoF Activities. ISO/IEC JTC1/SC29/WG11/M43748, Ljubljana, 2019.

[Sam17] J. Samelak et al. Efficient frame-compatible stereoscopic video coding using HEVC Screen Content Coding. IWSSIP 2017, Poznan, 2017.

[Sam19] J. Samelak and M. Domaański. Unified Screen Content and Multiview Video Coding - Experimental results. ISO/IEC JTC1/SC29/WG11/M46332, Marrakech, 2019.

[Sch19] S. Schwarz et al. Emerging MPEG Standards for Point Cloud Compression. IEEE J. on Emerging and Sel. Topics in Circuits and Systems, 2019.

[She06] H. Sheikh and A. Bovik. Image information and visual quality. IEEE Tr. on Image Proc., 2006.

[Sta18] O. Stankiewicz et al. A free-viewpoint television system for horizontal virtual navigation. IEEE Tr. on Multimedia, 2018.

[Sun17] Y. Sun et al. Weighted-to-Spherically-Uniform Quality Evaluation for Omnidirectional Video. IEEE Signal Processing Letters, 2017.

[Tan12] M. Tanimoto et al. FTV for 3-D spatial communication. Proc. of the IEEE, 2012.

[Tec16] G. Tech et al. Overview of the Multiview and 3D Extensions of High Efficiency Video Coding. IEEE Tr. Circuits and Syst. for Vid. Tech., 2016.

[Wan03] Z. Wang et al. Multiscale structural similarity for image quality assessment. The Thrity-Seventh Asilomar Conference on Signals, Systems and Computers, 2003.

[Wie19] M. Wien et al. Standardization Status of Immersive Video Coding. IEEE J. on Emerging and Selected Topics in Circuits and Systems, 2019.

[Xu16a] X. Xu et al. Intra Block Copy in HEVC Screen Content Coding Extensions. IEEE J. on Emerging and Selected Topics in Circuits and Systems, 2016.

[Xu16b] J. Xu et al. Overview of the Emerging HEVC Screen Content Coding Extension. IEEE Tr. on Circuits and Systems for Video Technology, 2016.

[Xu16c] X. Xu et al. Palette Mode Coding in HEVC Screen Content Coding Extension. IEEE J. on Emerging and Sel. Top. in Cir. and Syst., 2016.

[Yu15] H. Yu et al. Common Test Conditions for Screen Content Coding. JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11: Doc. JCTVC-U1015r2, Warsaw, Poland, 2015.

[Yua18] Y. Yuan et al. Object shape approximation and contour adaptive depth image coding for virtual view synthesis. IEEE Tr. on Circuits and Systems for Video Technology, 2018.

[Zha20] J. Zhang et al. Point Cloud Normal Estimation by Fast Guided Least Squares Representation. IEEE Access, 2020.

[Zhu19] S. Zhu et al. An improved depth image based virtual view synthesis method for interactive 3D video. IEEE Access, 2019.

Advanced Visual Interaction with Public Bicycle Sharing Systems

Alexandra Cortez
ViRVIG, Universitat Politècnica de Catalunya
Carrer Llorens i Artigas, 4-6 08028, Barcelona
alexandra.cortez@upc.edu

Pere-Pau Vázquez
ViRVIG, Universitat Politècnica de Catalunya
Carrer Llorens i Artigas, 4-6 08028, Barcelona
pere.pau.vazquez@upc.edu

ABSTRACT

Nowadays, public bicycle sharing systems have become popular and widespread across the world. Their usefulness largely depends on their ability to synchronize with citizens' usage patterns and optimize re-balancing operations that must be carried out to reduce outages. Two crucial factors to tackle this problem are stations' characteristics (geography, location, etc) and the availability of bikes and drop-off slots. Based on the requirements and input from regular users and experts in policy-making, system operation, and urban planning, we have created a web-based visualization system that facilitates the analysis of docking stations' behavior. This system provides the first group with the availability prediction of both bike and free slots in docking stations to assist their planning. Besides, the system helps the second group understand patterns of usage and get deeper insights (e.g. need for resizing or complementary transportation systems) to facilitate decision-making and better fulfill the citizens' needs. In a final evaluation, both groups found it highly useful, effective, and better suited than other existent applications.

Keywords

Human-centered computing, Visualization systems and tools, Visual analytics.

1 INTRODUCTION

Bicycle usage is undergoing a rapidly growing popularity in urban areas thanks to its multiple environmental, economic, social, and health benefits. Bicycles are commonly used as a replacement to other transportation systems for short-to-medium distance trips and for the so-called last-mile (or first-mile) connections to other transportation modes. Most bicycle sharing systems require citizens (from now on called regular users) to borrow a bike by going to one docking station, and later return the bike to another station. This use is commonly complemented by web services or mobile applications where regular users can check availability or current occupation of the stations, as well as booking bicycles in advance. However, information available through official applications is scarce or not reliable enough and oftentimes the interfaces are poorly designed.

Moreover, applications for system monitors and domain experts in urban planning and policy-making (from now on called advanced users) are not suited for proper monitoring of the system as a whole, since they lack information that may be relevant to these users such as the predicted occupation of a docking station or usage patterns.

The objective of this study is to pilot a web-based visualization system that takes into consideration the requirements of regular and advanced users and provides additional and novel features not existing in other systems. For instance, the prediction of slots or bicycles of each type (mechanical or electrical), analysis of docking stations suffering outages, or a prioritized list of stations close to a point of interest.

2 PREVIOUS WORK

Public bicycle sharing systems (from now on BSS), have received a lot of attention lately, since they are a huge opportunity for medium and large cities to reduce pollution and increase the health of their citizens, among other benefits [Ric15].

The analysis of how the BSS are used has been addressed in various ways. One common approach is to study the effect of urban configuration (e.g. elevation of stations) on bicycle flows and the destination preferences [KPL, FR14, FIE15]. Other studies mainly focus on building mathematical models to analyze the characteristics of bike trips or to make trip predictions

[BAF*11, HMO18]. The effect of weather and other variables such as the calendar events has been also investigated [YZWB20]. These studies mostly illustrate their findings with charts and maps, but no visual tool has been developed to help users to drill down the data. Another area of interest is optimizing re-balancing operations to generate efficient vehicle routes, but we focus on a wider audience that includes citizens and advanced users.

With a different focus, several approaches have been developed to face the problem of BSS visualization with diverse goals in mind. In some cases, the visualizations are simply static depictions, while in others, authors provide fully interactive tools or propose the development of multiple channels to increase engagement. The individual customers' journeys have been analyzed by different authors [WBD14, BWB14] to gain insights into people using those services. In contrast, our approach focuses on the analysis of stations' behavior instead of people's. Other investigations develop multiple bicycle sharing systems [OMS18, OCB14, MDO*20], but they are restricted to the analysis of aggregated data. Hence, insights on a single system to the detail required by advanced users cannot be extracted from their work. Several studies propose interactive visualization systems. [OST*16] explores a BSS behavior with multiple views emphasizing the balance of both weekdays and weekends. [DTL20] facilitates the exploration of what they call patterns of usage which can be extracted using a combination of spatial, temporal and destination dimensions, and clustering. Therefore, even when we share with these studies their focus on the availability of bicycles and clustering, we cover other purposes, such as the prediction of station behavior, as well as understanding long outages or resizing needs.

3 ANALYSIS OF REQUIREMENTS

After the analysis of several BSS, we identified some limitations on how the information is provided (e.g. clarity on the number of available bikes), usability problems, or lack of details (prediction of future availability). Given that the reported problems and digital maturity largely vary among public BSS, we have concentrated on Barcelona's BSS named Bicing. We performed a number of semi-structured interviews with several regular and advanced users to get more details of their needs. We were especially concerned with issues that can be addressed using the publicly available data. As a result, we identified the following requirements that are typically provided by official applications:

R1: Clear availability of bicycles and free slots.

R2: Distinction between mechanical/electrical bikes.

R3: Closest stations to a certain position.

A surprising factor found is the importance of clearly distinguish between the availability of mechanical and electrical bikes. For instance, some regular users reported that if an electrical bike is not available they will choose an alternative form of transportation (electrical motorbike, underground,...). In addition, regular users also have some needs not fulfilled by the current systems:

RU1: Prediction of bicycles or slots availability in the next one or two hours.

RU2: Prioritized list of stations closest to a point.

RU3: Distance and slope to a certain station.

RU4: Usage ratio: number of bicycles that are picked up or dropped off in a certain time interval.

In fact, real-time information of docking stations' state is useful, but predictions are also necessary for planning trips in advance, especially short term. Furthermore, information regarding the three of four closest stations to the one of interest together with the distance and slope needed to reach them is also a must for all users. Additionally, for stations with high usage, one user also suggested having some flow ratio that helps to understand whether bicycles might be available even if a certain station is currently empty.

On the other hand, advanced users highlighted the need to address problems such as re-balancing the docking stations, the overall flows of stations along the day and the week, as well as the proximity of other means of transportation systems; adding to our list of requirements the following ones:

RP1: Daily docking station usage

RP2: Usage patterns for multiple stations.

RP3: Visual exploration of stations' behavior around a certain position.

RP4: Stations with a higher degree of outages (might require increase/decrease their capacity).

Fulfilling these needs may facilitate the understanding of how the system works as a whole and helps advanced users to better plan BSS optimization.

To address all the collected requirements, we have designed a multiple-view visualization system and several features intended to help both regular and advanced users with the tasks that are more likely going to be useful for them.

4 BSS EXPLORATORY ANALYSIS

Our application offers two different modes: ***User*** designed for regular users and ***Planning*** with features useful for advanced users; both are based on the requirements presented in section 3. Although currently, we do not restrict the views to access, the idea is to present as initial view, the one corresponding to the type of use expected. In both cases, the main screen consists of multiple coordinated views, as described next.

4.1 User mode

The user-mode has been tailored to solve regular users common tasks which include: *i)* Current status of docking stations, *ii)* prediction of future availability of bikes and free slots, *iii)* closest stations to a point of interest or position, and *iv)* information regarding the frequency of stations' usage. As in many other applications, we build the view around the map of the city, where we show the availability of bicycles, as shown in Figure 1.

Users can access the availability of free bicycles or slots in two different ways: *i)* Clicking on a docking station in the map, and *ii)* Selecting a point of interest in the left menu. The outcome of any of those interactions is a set of views that provides the following information to the user: current availability and prediction, prioritized list of closest stations, and usage ratio chart.

The availability information solves requirements **R1** and **R2**. Moreover, since this can be triggered using a nearby docking station position in the map through the points of interest, it also solves **R3**. The prediction widget is a stacked bar chart that encodes the number of available bikes and free slots for the following two hours, solving **RU1**. Given that the available bicycles may be scarce (and bike reservations are not visible through the official applications), we also provide a list of the closest stations. The list is prioritized based on two parameters: distance and altitude. This facilitates the user to make informed decisions (such as not going uphill if sweating could be an issue) and solves **RU2** and **RU3**. Finally, the usage ratio is presented as a line plot to help users decide whether bicycles might be available even if the docking station is currently empty (fulfilling requirement **RU4**). This is reported as a quite common situation typically around large transportation hubs. In order to solve the most common tasks, we have designed a set of interactions in the different views described next, based on the view that triggers them.

Control Panel. The user view can be configured by using different tools on the left panel. By default, we render the bicycles' availability on the map (color-coded), but the user can swap it to show free slots. Second, the user can change the day of the month and the hour and make bus, metro, and train stations visible by toggling them on. A widget that gives the altitude of the docking stations can also be used to filter out stations. For users that may not be completely familiar with the city, we also enable an option to show points of interest such as the main transportation hubs and some special highly touristic zones such as Sagrada Família or Park Güell. By selecting one of those points the closest stations are highlighted and their predicted availability is shown.

Map navigation. The map provides the usual navigation features such as drag and zoom. Hovering over one station on the map will highlight it, together with the closest ones by increasing their size. Moreover, the highlighted station displays a popup that indicates its ID and current availability data (see Figure 2). Similarly, the elevation chart, located in the control panel, emphasizes the altitude of the pointed station. Selection is carried out by clicking. Upon selection, the rightmost view is populated with detailed information of bicycle availability for the following two hours (30 minutes intervals) for the selected station and the closest ones and the usage ratio chart. Figure 3 illustrates this.

Prediction panel. As discussed in section 3, decisions can be made taking into account three factors: *i)* availability of mechanical bicycles, *ii)* availability of electrical bikes and *iii)* free slots. Therefore, this view contains color-coded information to depict the number of elements in each category for the selected point and the four closest stations that have been prioritized according to distance and slope. Hovering over each of those categories in any of the bars highlights it in the other bar charts to facilitate quicker inspection of the data, as shown in Figure 4. A line chart indicating the activity is also displayed. This helps the user to distinguish between stations that may be empty but without activity and others that might be currently empty but have a high amount of activity, thus resulting in a potential bike available in the following minutes.

Getting insights. This set of features secures that users get relevant information from the BSS to make more informed decisions. For example, one can get an idea of how difficult is to get a bicycle near a certain metro or bus stop by activating the transportation layers and changing the day and hour. By filtering stations using the elevation chart and checking the activity ratio and the predicted availability, we can also see how the stations with higher elevation get empty soon and then keep low activity throughout the day. More complex cases are analyzed in Section 6.

4.2 Planning mode

The planning view provides more information related to groups of stations, so the advanced users can analyze their behavior collectively, instead of exploring the stations one by one. To this end, we implement useful features such as clustering based on the pickups and drop-offs (**RP2**), heatmaps that show the status of all the stations along the day (**RP4**), sliders to explore the behavior of stations per time frame (**RP1**), among others. Figure 5 shows this interface. As discussed in section 3, advanced users have many more needs than a regular user, so it is necessary the development of a larger set of higher-level features. For instance, the detection of stations with long outages can help system operators to better plan re-balancing operations, determine which regions of the city may require a new docking station, or which stations need to be resized. Besides, urban planners can complement zones with a low

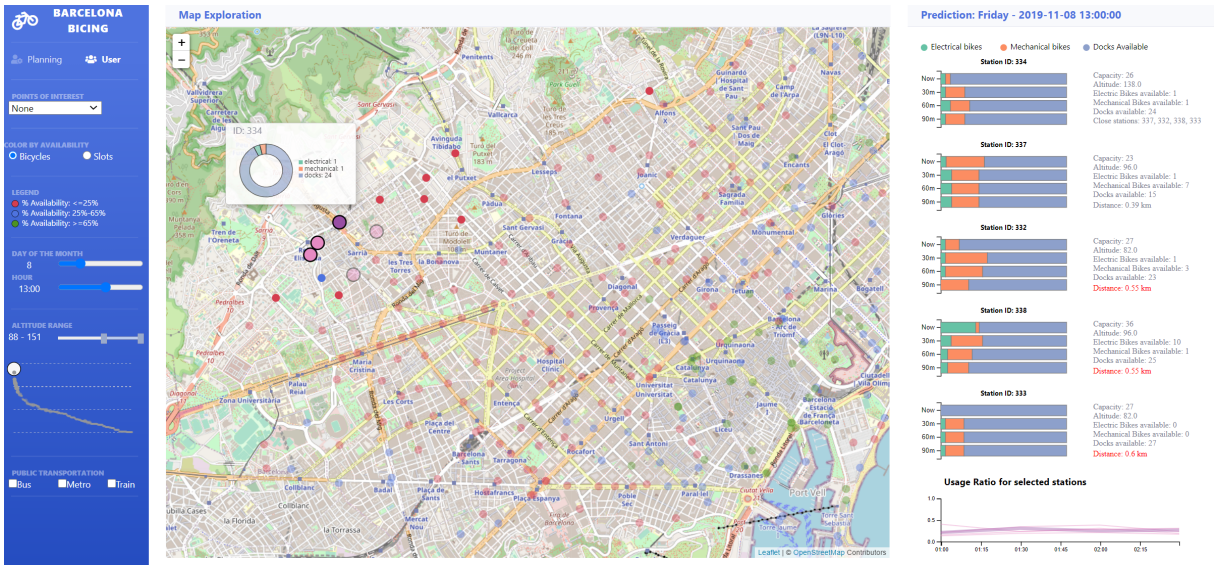


Figure 1: Interface for regular users. The user has selected the stations that are above a certain elevation on a Friday at 13:00. The map shows the availability of bicycles in stations (red colors indicate low availability). The right view displays the prediction for the next two hours. After selecting the one she is interested in, the rightmost view shows the closest stations prioritized by difficulty to access (altitude + distance) and the prediction shows that only one e-bike might arrive. Moreover, their activity is very low (bottom line chart).

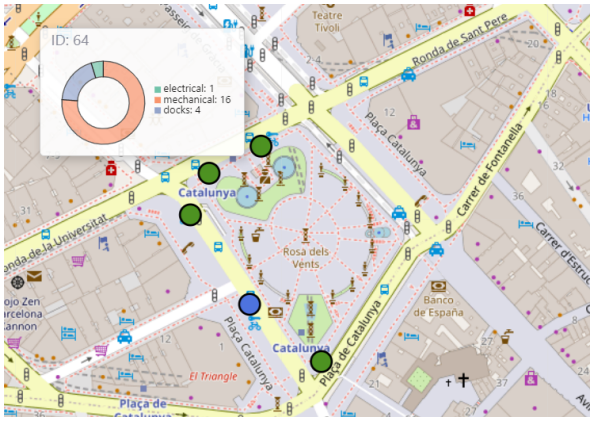


Figure 2: By hovering over the map, stations are highlighted and a donut chart glyph with the ID of the station and its availability is shown.

number of stations or large outage periods with other transportation means, such as bus stops. To solve their most common requirements, this view is divided into five components: Control panel, interactive map (**RP3**), status, clusters, and details view.

Clusters. After analyzing different strategies (e.g. pickups vs availability), both the pickups and drop-offs seem to produce an understandable classification of docking stations. We also separate the clustering for weekdays and weekends, since weekends tend to have a much smaller amount of activity and completely different patterns, as shown earlier [OMS18]. More details are given in section 5. The result for weekdays is a set of 5 clusters that can be described as:

Cluster 1: Peak of pickups in the 8-10 a.m. range and medium usage along the day. In general, low availability during the day.

Cluster 2: Peak of pickups in the 8-10 a.m. range and strong intensity of use during the afternoons, especially arrivals. Relatively low availability the central hours of the day.

Cluster 3: Low use and availability during the day.

Cluster 4: A high number of arrivals from 7 a.m. and intense use along the day, no clear availability pattern.

Cluster 5: Peak of drop-offs in the 7-9 a.m. range and medium activity with more departures than arrivals for the rest of the day, especially in the afternoon. They offer a healthy availability balance in the central hours of the day.

The identified behavior, relatively sharp peaks in clusters early in the morning but more distributed activity along the evening, and some usage peaks around 7-10 p.m., seems to align with the fact that working times in Barcelona start early in many jobs, but the end of the working journey is less defined, as found in other reports [ftIoLC*10].

Control panel. By default, the planning view displays weekdays data clustered and the cluster panel is showing the arrivals. Likewise, stations on the first cluster are selected and colored in the map panel using cluster information, and the stations with outages are emphasized. However, users can swap between different views with the tools on the left panel to show the information that suits their interest at most. As before, the

Figure 3: The prediction view provides the current status of the selected station and the closest ones, and the availability prediction for the following 2 hours in 30 minutes intervals. Finally, the bottom activity chart indicates whether the station is highly active or not.

day, hour, and transportation hubs can be changed, and stations can be filtered by their altitude, but there is no option to display the points of interest.

Map Navigation. As in user view, it is possible to drag and zoom. Using the control panel, it is easy to view the information for one cluster or all the stations and change the station's color by their cluster or by their availability. Hovering over one station will highlight it together with the closest stations, and will show a popup with the current availability. Station selection is carried out by clicking. This operation also emphasizes the pointed station in the elevation chart. The selected station and the closest ones are also highlighted in the status and cluster panel and the information panel is updated with the predicted availability for the next two hours at 30 min intervals and the usage ratio for the selected station.

Status panel. It is composed of three heatmaps that show arrivals, departures, and availability of all stations in the selected cluster along the day to facilitate the exploration per time frame and easily identify patterns.

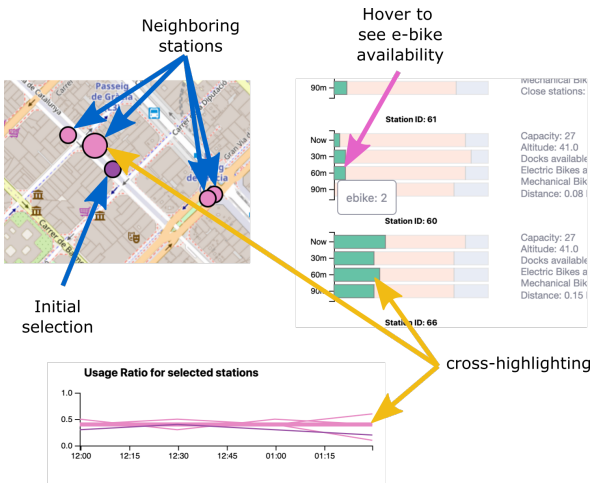


Figure 4: Exploring the status and prediction of a docking station and its neighborhood. After selecting a station on the map, we can hover over the prediction charts to see which ones will have electrical bikes, and the activity of the station is also highlighted, together with its location on the map.

Stations in a cluster are sorted according to the similarity between them, taking as an initial reference the medoid. Station and hour selection is done by hovering, this action also shows a popup with the percentage value of arrivals, departures, or availability and highlights the closest stations. The pointed station and the closest are emphasized in the elevation chart and the cluster and map panels. The control panel can be used to highlight the stations with extremely large outages.

Clusters behavior. It shows the information for all stations divided by clusters along the day and is useful to check different patterns between them. By default, the information of arrivals is displayed but it can be changed for departures or availability using the control panel. When a cluster is selected, the color of its corresponding plot changes to easily identify it. If users hover over a station, this and the closest ones are emphasized, even if they belong to a different cluster. This action also highlights the same stations in the map, status, and elevation chart.

Information panel. This panel is divided into three sections. The first one contains the predicted availability, for the selected station in the following two hours divided by electrical, mechanical bikes, and slots availability. The second part corresponds to a popup with information regarding its capacity, altitude, and current status. The last section shows the usage ratio chart.

Getting insights. The above-mentioned features may reveal interesting patterns that would be difficult to achieve with regular apps. For example, we can infer that rebalancing operations happen when we see sudden availability changes at night (or followed by very low activity ratios). Line charts can also reveal which dock-

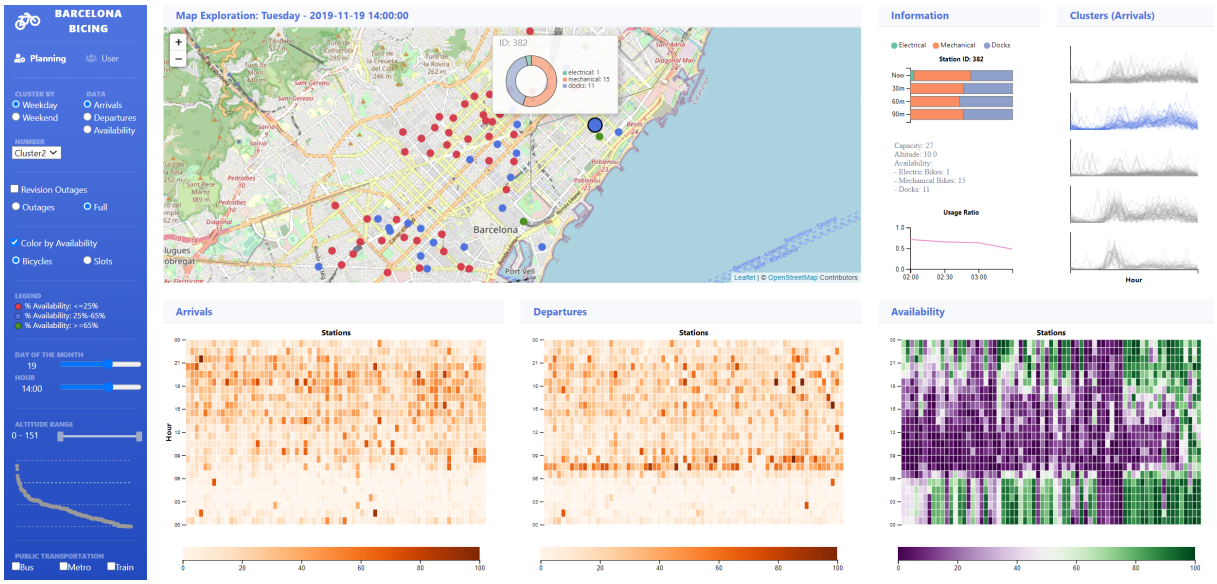


Figure 5: The planning interface. The map may encode the different clusters or the availability, together with information of potential outages. The heatmaps show different aspects of the usage: arrivals, departures, and availability. On the right, the view displays the behavior of each cluster, using the same usage state defined in the left panel. In this case, she has selected cluster 2, which groups several stations in the center of Barcelona. By hovering over them, we can see that they keep a high degree of activity along the central hours of the day, as may be inferred from the bottom heatmaps.

ing stations belong to mostly residential zones (e.g. high number of pickups in the morning and evening and very low activity in the middle of the day).

5 IMPLEMENTATION

Interface. The interface is built using HTML and it's composed of two different tabs: *PlanningHome* and *UserHome*. The code is built using Flask for managing HTTP calls and rendering templates. Views are implemented using D3, and Leaflet library for the maps. Data management is carried out in Python, using *pandas* and *numpy*. The forecasting and initial data processing, which includes cleansing, clustering, and data derivation algorithms are performed using R.

Data gathering and processing. The original samples are updated approximately every 5 minutes having more than 3.5 million records every month. Considering that 2020 data had an abnormal behavior due to the lockdown period and several mobility restrictions, we use data from 2019 to train and evaluate the models and deploy the use cases. Moreover, the Bicing provider company has changed during 2019 which means there is no continuity in the information during the first months since the change occurred gradually. Therefore we focus on the last months of 2019 from September to November.

The Barcelona BSS has 424 docking stations. We first clean the data to avoid including information of docking stations that could appear in the set but are still not used, a special field in the set denotes it, as well as others that were not working properly. The final result is a set of 409 stations. Also, garbled information is cleaned such as dates outside the analyzed range or negative data availability. From this data, we calculate additional information necessary to deploy the visual interface: day of the week (and month), the number of total docks available in each station, and we summarize data about the number of pickups and drop-offs (absolute and relative). Additionally, the Bicing usage data is combined with other sources that include information such as the GPS position of the docking stations as well as their altitude. We also add other public data such as the location of all metro, bus, and train stations and some information regarding the large transportation hubs in Barcelona city, such as Barcelona-Sants or Plaça Catalunya.

After the data is cleaned, we apply different algorithms to derive the data that is later used in our visualization. These includes: *i)* Clustering docking stations, *ii)*, closest stations calculation for all stations, *iii)* calculation of usage ratios, *iv)* calculation of outages, and *v)* prediction of usage per station.

Clustering. It can be addressed in different ways considering several aspects. First, data partitioning can be done for different time intervals (months, weeks, days, hours…). In our case, after some experimental evidence, we find that the behavior changes depending on whether the day is a weekday or a weekend [OMS18]. Second, we can use data of pickups, drop-offs, availability, or some ratio of usage. After testing different

combinations, we discover that data from pickups and drop-offs give us better results clustering the stations to discover their behavior. There are many clustering algorithms [RCC*19] that may be suitable for different tasks. Key to the clustering algorithm is the selection of the distance measure, and we have many options such as the Euclidean, correlation, temporal correlation, or dynamic time warping. Depending on the measure selected, our similarity measure will be more or less resistant to noise, and to time shifts [SGP16]. After testing some of the most widely used algorithms (PAM, K-means, DBSCAN, AGNES), we find that both Partitional Around Medoids (PAM or k-medoid) [KAU87] and K-means [M*67] outperform both internal and stability results and are suitable for our problem. PAM method has the objective of minimizing the average dissimilarity around the nearest medoid (always an element of the original dataset) which is recomputed in each iteration. K-means works in the same way, but groups the elements of a cluster around the nearest mean. Using the PAM method (which gives better results for the silhouette coefficient), we build two clusterization sets: one based on the activity score during weekdays and the other for the weekends. A similarity matrix is defined based on Euclidean distance, and the silhouette analysis is executed to test the degree of separation between the clusters.

Closest stations. When a docking station is empty and the user needs to take a bike, or when it is full and the user needs to drop off one, BSS do not provide information on the closest ones. To build this feature, we need to have some information on what are the closest stations. In our case, this is precalculated and attached to each element. Computation is pretty simple and has a quadratic cost. For each docking station, the four closest are found using the earth distance metric from *fossil* package in R and ranked using their distance to the original point and their altitude.

Usage ratios. Usage ratios are derived from the frequency of pickups and drop-offs. This ratio helps us to understand if a docking station status corresponds to a high dynamic behavior or a static behavior. It is calculated as follows:

$$use_{ijk} = \sum arrivals_{ijk} + \sum departures_{ijk}$$

Where i = StationID, j = day, k= hour

To better interpret this ratio, we re-scale the value to make all the elements lie between 0 and 1, having a common scale:

$$UsageRatio_{ijk} = \frac{(use_{ijk} - MinUse)}{MaxUse - MinUse}$$

Where i = StationID, j = day, k= hour

Outages ratios. Two different ratios are calculated: one used to identify stations that have a low number of bicycles available (10% or less) most of the day and the other to label those stations that have a high number of bikes available (75% or more) during working hours. These ratios are estimated with a daily frequency dividing the numbers of hours when a station i had an outage of type k by the number of total hours (24 hours for k=empty, and 11 hours for k=full). The generic formula used is presented as follows:

$$outageRatio_{ijk} = \frac{\sum outage_{ijk}}{Totalhours_k}$$

Where i = StationID, j = day, k= type of outage

Usage prediction. Usage forecasting is extremely important for both types of users. For advanced users, it is key for re-balancing stations and studying mobility patterns while for regular users is essential to plan future trips. Several studies have been conducted to predict bike-sharing trips using different methods and data [LHP18, XJL18, WK18]. These contribute valuable insights to understand the factors that affect bike-sharing demand. We use different features to train the models, which can be classified as station-related and weather features. Station-related features are the day, time slot, number of available bikes, and station's capacity. The weather features are composed of temperature, wind, humidity, atmospheric pressure, and weather classified in four categories: sunny, cloudy, light rain, and heavy rain. As mentioned before, the provider company was changed during 2019 causing data inconsistency during this transition. To tackle this, the models are trained using data from September and October 2019 and test with November data.

We test two forecast models: Random Forest [FW17] and Prophet [TL17]. As there are mechanical and electrical bikes available in each station, we run both models for each station and predict both mechanical and electrical bikes. Besides the predicted variable (availability) is transformed using the natural logarithm but results are not improved. Different measures are used to compare both models, but we keep RMSE since others (e.g. MAE, MAPE) have a very strong correlation with it and do not add valuable insights. Random Forest is the model which gives better results with lower RMSE levels for both scenarios especially for electrical bikes forecasting using non-transformed data. It is optimized in terms of the number of trees (500 trees). Additionally, we create a different error measure considering the scenarios that affect the users: no bicycles available or no slots to return bikes. The stations are divided into three categories: *i)* Available bikes less than 10% *ii)* Available slots less than 10% *iii)* Healthy status (bikes between 11-89%). Finally, we evaluate how many times our model predicts a different status. The

results are promising, for mechanical bikes only 33% of times the status is different while for electrical bikes the error is less than 6%. These results suggest the model applied has reasonably good prediction accuracy. However, it can be improved as systems mature and data is collected across a longer period.

6 USE CASES

To illustrate the use of the application, we design two different use cases that correspond to the two views designed: one for a regular user-oriented profile, and another for a planning-oriented profile.

6.1 Availability and free slots prediction

The first use case consists of a multi-query situation: the user is leaving from a meeting and she wants to know where is the closest docking station and whether it has electrical bikes because she is going to an elevated part of the city. Then, she wants to check the availability of free slots near her destination in one hour or later.

To solve the problem, she has to look for a docking station using the map. By hovering with the mouse, the visual marks indicating the closest stations scale up and are emphasized with an outline. Given that they are colored according to availability, it is easy to infer whether there are free bikes. If she is close to one of the most visited areas in the city, she can also use the left panel to select the closest stations to that point. By clicking on the station of interest, a list of the closest stations will appear on the right panel. Each station shows a horizontal bar chart where the first bar (top) is the current availability and the others are the prediction for 30 minutes slots. Electrical bikes are encoded in green, thus, it is easy to identify their availability in any of the neighboring stations. By hovering over the green bar, the concrete number can be read. Additionally, the usage ratio chart at the bottom is also highlighted for the currently selected station. Therefore, if the availability is low, she can have extra information on whether it is due to the lack of movement or the high throughput by checking it. Similarly, to determine the best place to leave the bicycle she can use the map. By hovering over a docking station near her destination, she will be able to see whether there are free slots. If there are no slots are available, by clicking on it, she can see a prediction of future available slots in the selected station as well as in neighboring docking stations. The list of docking stations specifies the slope and the distance to cover. This way, she can make more educated decisions on which stations are adequate for her needs.

6.2 Analyzing stations with severe outages

In this case, an urban planner wants to analyze docking stations that suffer outages. If that is the case, she will be interested in knowing where, and how other closest stations behave in the same time intervals. She will use a deductive method thanks to the characteristics provided by the application.

By selecting the planning view and the outages option, the stations that have outages or are full an important part of the day are highlighted on the availability heatmap. To increase clarity, she may set the color of the stations on the map to encode the availability instead of the cluster color, using the left control panel. By hovering over the availability heatmap the station under the mouse and the closest stations are emphasized to further investigate. For example, she sees a group of 3 stations that have outages, but one of them (number 3) behaves slightly differently along the day, as shown in Figure 6-left. She may hypothesize that it might be related to the presence of public transportation. Effectively, if she toggles metro stations on, there is a metro station, La Llacuna, which is closer to the docking stations that have severe outages (see Figure 6-right), while station number 3 is slightly farther. If the bus stops layer is activated, she sees that station 3 lies next to two bus stations. It turns out that those are precisely two stations belonging to two different (and orthogonal) lines that have highly frequent buses (the types H and V, which travel horizontally and vertically). Given that the same transportation ticket can be used (within one hour and a half) to take a bus, those passengers might not need to change to a bicycle because they can take a bus instead.

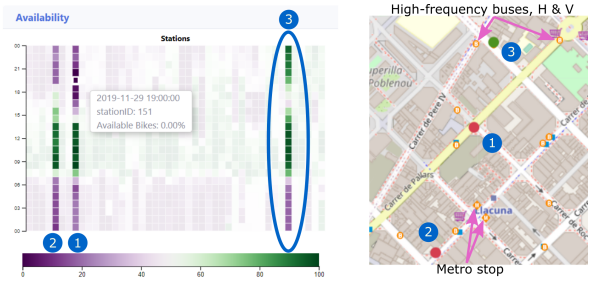


Figure 6: Inspecting the behavior of four neighboring docking stations. Station 3 does not suffer outages for so long in the afternoon. By activating the transportation layer, we find that 1 and 2 are next to a metro station. Moreover, we can also see that 3, is next to two bus stations that have highly frequent buses (H and V type) which travel in orthogonal directions, and can be used with the same transportation ticket.

6.3 Evaluation

We evaluated our application in two different ways: *i)* an informal user study with a few subscribers of the Barcelona Bicing system, and *ii)* interviews with domain experts in urban planning, system operation, and policy-making.

For the user study, we had an interactive remote session (due to the limitations in mobility and social contacts

during the pandemic) where we demonstrated the features of the application. This was followed by an interview where we discussed the possibilities and limitations of the system. Then, we asked them to complete a standard questionnaire offline. The questionnaire consisted of a series of questions that should be evaluated using a 1-7 Likert scale where 1 means totally disagree and 7 means totally agree. Questions were related to their experience with the Bicing system application and how the features of our application could benefit them. The average of obtained results are quite positive:

Question	Grade
It is useful to better plan usage	6.7
I would use it to check station status	6.7
I would use it to check future availability	5.7
Usage prediction would help me decide	7.0
Importance: stations altitude & closeness	6.0
Utility: check availability in relevant areas	6.0
Utility: prioritized list of closest stations	6.7
Information is relevant for a Bicing user	7.0
Ease of use	6.0
More useful than current website or app	6.3

Table 1: Results of users' interviews

We interviewed three domain experts, an architect with experience in urban planning, a politician with responsibilities in policy-making in the area of Barcelona, and a technician that supervises the bike-sharing system of a small city. We asked them to complete a questionnaire regarding potential features of the system. These advanced users agreed that being able to see the usage patterns is strongly useful to better understand citizens' behavior. Some features were highly rated such as the clustering based on stations' behavior and their representation using heatmaps (especially useful for the creation of mobility policies). They also considered the availability heatmap more relevant to monitor the system as a whole. Bikes' availability, altitude, and distance to the neighboring stations were quite interesting for re-balancing purposes. Detailed information on times of high usage of docking stations and availability prediction was found very useful for transportation planning and policy-making. Stations that are most of the time full or empty were considered extremely important for mobility planning. After the system was almost finished, we showed an interactive demo to a technician that supervises the BSS in his city, and he found that the usage patterns shown by the application, and no present in others, are highly useful to make decisions such as changing the placement of a docking station, improving the public transportation in certain locations, getting regular reports on the status and usage of the system, and monitoring the real usage (beyond to what the system operator company actually reports). All of them are aspects that they cannot analyze with their current means. As seen, both regular and advanced users were enthusiastic with the application, and have provided very important feedback for future improvements.

7 CONCLUSIONS

We implement a full web-based visualization system to explore and analyze public BSS. Barcelona Bicing system is used as our first scenario. Compared to current alternatives, we provide much-desired features for regular users such as availability prediction and nearest docking stations prioritized by distance and altitude. For advanced users, we also show station patterns along the day, emphasized stations with long outages, and cluster the stations based on their behavior.

However, the information we used is constrained to the one that is available through the Barcelona Open Data website, and unfortunately, there is only historical data available through this platform. The real potential of this application can be leveraged with real-time data that would require access to the original data sources and permanent computer resources. Due to pandemic and temporal limitations, the sample taken to evaluate the system has been reduced and a larger quantitative study would be needed to characterize it with greater precision. Additionally, the interaction with the system operator was challenging due to the change of operator and the pandemic.

There are several development lines we are considering. Regarding the prediction, we have already tested Random Forest and Prophet models but we plan to evaluate other algorithms' performance. The used data is limited in time due to the change in the operator and the effect of the 2020 lockdown. So, we plan to perform a deeper study on the most suitable prediction algorithm whenever the system is not interrupted again and recovers normal usage patterns. In addition, the current databases do not contain detailed information regarding the trips, that is, the concrete pickup and drop-off docking stations of each bike. This would open other possibilities to perform more advanced analysis. We have had contact with the Barcelona Open Data director to gain access to this information. Finally, another possibility would be to establish relationships between BSS and other transportation systems, for instance, information of travellers' arriving and leaving to Barcelona via train stations.

8 ACKNOWLEDGMENTS

Partially supported by project TIN2017-88515-C2-1-R(GEN3DLIVE), from the Spanish Ministerio de Economía y Competitividad, by 839 FEDER (EU) funds. The authors would like to thank Francisco Ridruejo and Ignacio Pascual for their invaluable contributions and feedback, as well as the users who participated in our user study.

9 REFERENCES

[BAF*11] Borgnat P., Abry P., Flandrin P., Robardet C., Rouquier J.-B., Fleury E., Shared bicycles in a city: A signal processing and data analysis perspective. Advances in Complex Systems 14, 03, 2011, 415–438.

[BWB14] Beecham R., Wood J., Bowerman A., Studying commuting behaviours using collaborative visual analytics. Computers, Environment and Urban Systems 47, 2014, 5–15.

[DTL20] Dai H., Tao Y., Lin H., Visual analytics of urban transportation from a bike-sharing and taxi perspective. Journal of Visualization 23, 6, 2020, 1053–1070.

[FIE15] Faghih-Imani A., Eluru N., Analysing bicycle-sharing system user destination choice preferences: Chicago's divvy system. Journal of transport geography 44, 2015, 53–64.

[FR14] Frade I., Ribeiro A., Bicycle sharing systems demand. Procedia-Social and Behavioral Sciences 111, 2014, 518–527.

[ftIoLC*10] Europen Foundation for the Improvement of Living Conditions, Comparative analysis of working time in the European union, 2010.

[FW17] Feng Y., Wang S., A forecast for bicycle rental demand based on random forests and multiple linear regression. In 16th IEEE/ACIS International Conference on Computer and Information Science, May 24-26, 2017, Zhu G., Yao S., Cui X., Xu S., (Eds.), IEEE Computer Society, pp. 101–105.

[HMO18] Holmgren J., Moltubakk G., O'Neill J., Regression-based evaluation of bicycle flow trend estimates. Procedia computer science 130, 2018, 518–525.

[KAU87] KAUFMANN L., Clustering by means of medoids. Proc. Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, 1987, 405–416.

[KPL] Kim I., Pelechrinis K., Lee A. J., The anatomy of the daily usage of bike sharing systems: Elevation, distance and seasonality, ACM SIGKDD workshop on Urban Computing, 2020.

[LHP18] LIN L., HE Z., PEETA S., Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. Transportation Research Part C: Emerging Technologies 97 (Dec 2018), 258–276.

[M*67] MacQueen J., et al., Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967, vol. 1, Oakland, CA, USA, pp. 281–297.

[MDO*20] Meddin R., DeMaio P., O'Brien O., Rabello R., Yu C., Gupta R., Seamon J., The meddin bike-sharing world map, 2020,

[OCB14] O'Brien O., Cheshire J., Batty M., Mining bicycle sharing data for generating insights into sustainable transport systems. Journal of Transport Geography 34, 2014, 262 – 273.

[OMS18] Oppermann M., Möller T., Sedlmair M., Bike sharing atlas: visual analysis of bike-sharing networks. Int. J. Transp 6, 2018, 1–14.

[OST*16] Oliveira G. N., Sotomayor J. L., Torchelsen R. P., Silva C. T., Comba J. L., Visual analysis of bike-sharing systems. Computers & Graphics 60, 2016, 119–129.

[RCC*19] Rodriguez M. Z., Comin C. H., Casanova D., Bruno O. M., Amancio D. R., Costa L. d. F., Rodrigues F. A., Clustering algorithms: A comparative approach. PloS one 14, 1, 2019, e0210236.

[Ric15] Ricci M., Bike sharing: A review of evidence on impacts and processes of implementation and operation. Research in Transportation Business & Management 15, 2015, 28–38.

[SGP16] Saas A., Guitart A., Periánez A., Discovering playing patterns: Time series clustering of free-to-play game data. In 2016 IEEE Conference on Computational Intelligence and Games (CIG), 2016, IEEE, pp. 1–8.

[TL17] Taylor S., Letham B., Forecasting at scale. The American Statistician 72, Sep, 2017.

[WK18] WANG B., KIM I., Short-term prediction for bike-sharing service using machine learning. Transportation Research Procedia 34(Jan. 2018), 171–178. International Symposium of Transport Simulation and International Workshop on Traffic Data Collection and its Standardization 2018, ISTS IWTDCS 2018;

[WBD14] Wood J., Beecham R., Dykes J., Moving beyond sequential design: Reflections on a rich multi-channel approach to data visualization. IEEE transactions on visualization and computer graphics 20, 12, 2014, 2171–2180.

[XJL18] XU C., JI J., LIU P., The station-free sharing bike demand forecasting with a deep learning approach and large-scale datasets. Transportation Research Part C: Emerging Technologies95 (2018), 47–60.

[YZWB20] Younes H., Zou Z., Wu J., Baiocchi G., Comparing the temporal determinants of dockless scooter-share and station-based bike-share in washington, dc. Transportation Research Part A: Policy and Practice 134, 2020, 308–320.

Unsupervised SIFT features-to-Image Translation using CycleGAN

Sławomir Maćkowiak, Patryk Brudz, Mikołaj Ciesielski, Maciej Wawrzyniak

Poznań University of Technology
Institute of Multimedia Telecommunications
ul. Polanka 3, 60965 Poznań, Poland

[slawomir.mackowiak, maciej.wawrzyniak]@put.poznan,pl
[patryk.brudz, mikolaj.j.ciesielski]@student.put.poznan.pl

ABSTRACT

The generation of video content from a small set of data representing the features of objects has very promising application prospects. This is particularly important in the context of the work of the MPEG Video Coding for Machine group, where various efforts are being undertaken related to efficient image coding for machines and humans. The representation of feature points well understood by machines in a video form, which is easy to understand by humans, is an important current challenge. This paper presents results on the ability to generate images from a set of SIFT feature points without descriptors using the generative adversarial network CycleGAN. The impact of the SIFT keypoint representation method on the learning quality of the network is presented. The results and a subjective evaluation of the generated images are presented.

Keywords

SIFT, features, keypoints, CycleGAN.

1. INTRODUCTION

Image or video compression is used in order to reduce the storage requirements of an image or video without substantially reducing the image quality so that the compressed image or video may be utilized by a human user. However, image and video data is nowadays not only looked at by human beings. Fuelled by the recent advances in machine learning along with the abundance of sensors, image and video data can successfully be analysed by machines, such as a self-driving vehicles, robots that autonomously move in an environment to complete a tasks, video surveillance in the context of smart cities (e.g. the traffic: monitoring, flow prediction, density detection and prediction). This led to the introduction of Video Coding for Machines (VCM) as described in document ISO/IEC JTC 1/SC 29/WG 2 N18 "Use cases and requirements for Video Coding for Machines" [Mpe20].

The current work of the MPEG VCM working group is concerned with the efficient transmission of both the stream of keypoints and descriptors intended for machines, classification algorithms as well as the stream of vision intended for humans who would have a view of the content that is described by a stream of features and feature points [Mpe20b].

It must be made clear, the technique presented here is not a video compression technique. The main goal is to reconstruct the image based on its features only. Features do not carry all the information about the image. Hence, we have taken the trouble to propose a method to reconstruct the video content represented by the features only. Such a reconstructed, synthetic, image could serve as a visual representation of content intended for humans and could, in some situations, replace a stream of vision transmitted in parallel. Moreover, an attempt has been made to generate video content based only on keypoints and not on the descriptors that accompany such keypoints in SIFT, SURF or MPEG-7 CDVS streams [Pas12, Dua15, Mpe17]. This approach is unique and there is a lack of such solutions in the literature.

The paper is organized as follows. In Section 2, we briefly review techniques of partial reconstruction of an image from its features only. In Section 3, briefly the SIFT technique is presented. In Section 4, the GAN networks, with special attention to CycleGAN networks are presented. In Section 5, we discuss our proposed reconstruction algorithm and the impact of different approaches to defining input feature maps on the reconstruction process. The extensive quality

assessment results are presented in Section 6, and a final summary is given in Section 7.

2. RECONSTRUCTION OF AN IMAGE FROM ITS FEATURES - REVIEW

There exist some analytical approaches of reconstruction of an image from its features [Vond13, Deso17, Deso18]. The reconstruction is built from the functions, which gradients match to the original descriptors. The results are mostly monochromatic and sometimes recover the object shapes well.

Other group of techniques are the techniques of partial reconstruction of an image. Work [Wein11] proposed to build the approximation of an image from small patches taken from external database. The descriptors from the original image are divided into subsets, corresponding to smaller areas. Then these subsets are matched to the features stored in database. The best matches point to the small patches, which should be placed in appropriate place in the reconstruction. The images obtained by this method looks like mosaic of patches. However most of the details of image like: the corners, edges are reconstructed, so the content can be recognized by the human.

Next group of techniques are methods that use the convolutional neural networks or, more precisely, using generative adversarial networks. At the output, the reconstruction looks natural, but the results strongly depend on the learning dataset used. In paper [Wu20c], the authors proposed an accurate generative model to reconstruct an image based on its SIFT features. The designed generative model consists of two networks, the first one tries to learn the structural information of the image by transforming from SIFT features to LBP (Local Binary Pattern) features, and the second one aims to reconstruct the point values with LBP support. The results are for the test sets very good. The authors conclude that it is much more difficult when only the SIFT descriptors are accessed and not their coordinates, then "modest success" of image reconstruction can be achieved for highly structured images (e.g. faces), but the technique fails for more general images. The image can be reconstructed with reasonably good quality from the SIFT coordinates alone. Another article showing the possibility of reconstructing a face image on the basis of its descriptors is [Wu19b]. The authors proposed a novel end-to-end face reconstruction model from local SIFT descriptors based on the Conditional Generative Adversarial Networks (cGAN). Their model works in a coarse to-fine manner. By resorting to the well designed multiscale feature maps generation algorithm and the conditional adversarial networks, their approach has substantially improved the reconstruction results compared with existing ones. The authors conclude that local descriptors contain a surprising amount of information about the original image. If the local descriptors (even part of them) are extracted, the image can be reconstructed with high probability.

It should be emphasized that the features, keypoints are determined on the monochrome image. Color information is discarded in the process of determining the features. Therefore, the image reconstruction is usually a monochromatic image. Color images can be obtained only in techniques based on the use of GANs and the quality of color images will depend on the size of the learning set.

3. SIFT

The Scale Invariant Feature Transform (SIFT) algorithm was published by David Lowe in 1999 [Low04]. It is a carefully designed procedure with empirically determined pair-measures for determining invariant and characteristic features.

A good definition of an image feature, is a point in an image showing detection stability under local and global perturbations in the image domain, including perspective transformations, changes in image scale, and illumination variations.

A SIFT type detector is divided into two phases, a keypoint detection phase and a keypoint description phase. In the detection phase, we determine the extremes in scale space for potential significant points in the image and their parameters. A SIFT keypoint is a circular image region with an orientation. It is described by a geometric frame of four parameters: the keypoint center coordinates *x* and *y*, its *scale/size* (the radius of the region), and its *orientation* (an angle expressed in degrees) (Fig.1). In the description phase, we assign to significant points their multidimensional descriptor. Descriptors contain only information about gradients - high frequency information in a small area around the keypoints.

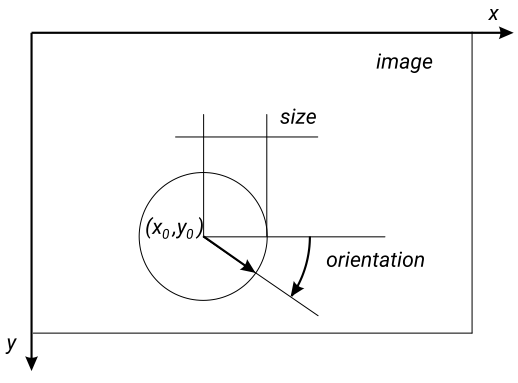


Fig.1. A SIFT keypoint.

We decided to use only the first phase of the algorithm and use only keypoints without descriptors in this proposal. Image reconstruction based on keypoint information without accompanying information about

the neighborhood of points is very difficult. The application of generative modeling using Generative Adversarial Networks (GANs) is very promising in this aspect. In particular, a cross-domain translation using GANs is very interested.

4. GENERATIVE ADVERSARIAL NETWORKS AND CycleGAN

Generative adversarial GANs [Goo14, Goo16] are particular machine learning architectures, developed by Ian Goodfellow in 2014. GANs are composed of two neural networks, one of them is called Generator, whose task is to modify the random input noise into a synthetic image, then this image is sent to a second neural network prepared to compare two images, the original input and the one prepared by the generator, this network is called Discriminator.

Supervision of learning is limited to keeping an eye on the quality and diversity in the training packet. The generator processes the noise in such a way as to "cheat" the Discriminator, whose task is to detect the original image. Then the network that lost the competition is modified. In this way the full process of machine learning in adversarial networks is carried out, it is important that these networks receive a large and diverse training set (input images) to avoid overfitting the network.

Image-to-image translation involves the creation of a new synthetic image through the process of learning the mapping between the input image and the output image using a properly prepared training data set [Gat16, Iso17]. This process usually requires a very large dataset, which can be difficult or expensive to prepare, and sometimes impossible to prepare. Cycle Generative Adversarial Network (CycleGAN) is a technique involving automatic learning of image-to-image translation models without labels or example pairs [Zhu20]. The models are trained in an unsupervised manner using two image databases that can be uncorrelated. The CycleGAN network architecture is based on two GANs having their own generator and discriminator. The task of the generators is to transform an image from their dataset into an image that will match the dataset of the other network. The job of the discriminators, on the other hand, is to compare the image generated by the generator from its own network and compare it to the data set of the other network. The cycle in CycleGAN is that the images generated by the generator of the first network are provided to the generator of the second network and similarly the images of the generator of the second network are provided to the generator of the first network. CycleGAN can be used in, among other things: transferring painting styles, generating images from images, changing individual objects to other objects. In this case, we want to use CycleGAN to translate the data of a domain representing SIFT keypoints into an image.

5. PROPOSED METH|OD

First of all, as a basic step, it was necessary to focus on the representation of keypoints that could be implemented as feature maps in the learning process of the CycleGAN network. We already know that we only want to use information about keypoints and not keypoint descriptors. So we need to develop an efficient representation of these points.

In the first approach, the CycleGan network was learned with the position of keypoints only. Reconstructing the image as a generative image did not give satisfactory results. The neural network returned highly distorted images after about 14 hours of learning (Fig. 2). The reason for this is that too little information was transmitted through the feature map representing the keypoints.

Note that the color representation of the reconstructed images is obtained as an additional effect of applying the GAN on the training set and results from the learning process on a limited set of objects. The goal is to obtain good representations in the shape and details of the objects. The color representation of the objects will not be evaluated.

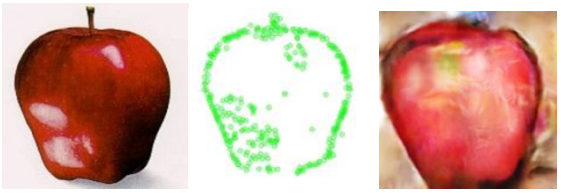


Fig. 2. Example of generated image [Epoch 1482].

Note that the SIFT keypoints are a collection of data. This makes it impossible to directly feed the keypoints into a CycleGAN network for training. In the first stage, we propose to rearrange the SIFT keypoints of an image as a set of feature maps, which can accommodate the input of the coarse image reconstruction component. Several approaches were tested. The best solution was to use three parameters to describe the keypoint. Proposed framework is presented on Fig. 3. For each keypoint with position x,y we will use from the SIFT algorithm the strength of the technique's response to the presence of a corner, and the dominant orientation based on the distribution of quantize gradients of the point directions (SIFT additionally performs Gaussian filtering to reduce the influence of gradients from the boundary of the region of interest). So we have *response*, *orientation*, and *size* parameters.

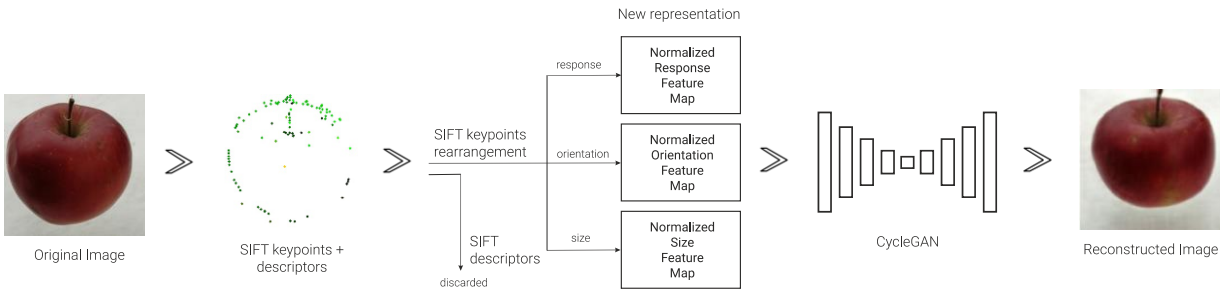


Fig. 3. The proposed framework.

This three parameters were selected to simplify the training process of CycleGAN network. Each set of keypoints along with the parameters were converted into an image form thus forming a set of training and test images. So the three feature maps representing response, orientation and size respectively were represented as three components forming a color image. Hence, to match the CycleGAN network requirements, the *response* values, originally represented by float value between 0 and 1, are multiplied by 255 and represented by integers in the range <0,255>. This will be the blue component of the image. The *orientation* parameter representing the angle from 0 to 360 degrees is first normalized to 1 and then represented by an integer value in the range <0,255>. The size parameter is normalized to integer values in the range <0,255>.

Attribute-based color parameterization appeared to indicate the largest values in the gradient orientation attribute (green). After visualizing the descriptors and comparing them with previous parameterization attempts, another attempt was made to reconstruct the image. For this attempt, a fully composited set of images was prepared aiming to maximize the quality of the results.

Details of the experiments

We used the following implementations and parameters in the experiments: the SIFT features were extracted using the SIFT feature detector/extractor from OpenCV version 4.3.0. and Python.

The first step was to determine such parameters of SIFT keypoints in order to train the network to represent the shape of the object. It was ensured in the SIFT algorithm that all possible feature points would be determined. The number of layers in an octave was equal to 3. The threshold to eliminate feature points with poor contrast was set to 0.03. The larger the parameter, the fewer feature points will be determined. The threshold for eliminating feature points on edges was set based on experience and the need to recreate the outline of the object. We left the sigma parameter at the default value, i.e. 1.6, and the edge parameter at the smallest possible value. Of course, it was possible to choose these parameters so that we could get more keypoints and "more accurate" outline, but we resigned from that, because a larger number of characteristic points did not significantly affect the learning of the network. A larger number of keypoints, however, increased network learning time.

The CycleGAN network implementation [Lin00] was used in this study, with the following parameters: unpaired datasets with 128x128 [px] resolution, three feature maps as input, number of filters in the first layer of G and D, 32 and 64 respectively, cycle-consistency loss equal to 10, identity loss equal to 1, Adam optimizer algorithm used. Learning rate parameter equals to 0.0002 (also referred to as the learning rate or step size, the proportion that weights are updated). The exponential decay rate for the 1st moment estimates is 0.5 [Kin14, Sas19].

The research were conducted on a computing unit with the following characteristics: processor: Intel Core i5 9300H 2.4Ghz, graphics card: Nvidia GeForce GTX 1650 (mobile) 4GB GDDR5, RAM: 16GB DDR4. Software: system: Microsoft Windows 10 Education N 10.0.18363 version 1909, environment: PyCharm Community 2020.2.3, Nvidia Cuda 10.1, Nvidia cuDNN 7.6.4.38, Keras 2.4.3, TensorFlow 2.3.2, OpenCV 4.5.1.

Training the adversarial network

Earlier attempts at the learning process were conducted on pre-made ImageNet datasets. The time required to learn the network was very long, this was due to the wide variety of content contained within it. With respect to the generative images obtained during testing, created from the keypoint parameterization, the decision was made to create a custom controlled dataset.

Preparation of the dataset assumed appropriate composition: uniform background, strongly outlined object edges, diverse perspective, resolution: 128x128 [px]. The independent dataset contained 2000 images

of each objects, of which 7.5% were reused to extract keypoints and transform their parameters into maps of the features represented by the image. This resulted in two independent training sets, which, starting from the assumptions of the type of neural network used, were uncorrelated with each other. Three unique objects were represented, with increasing levels of complexity from the perspective of the neural network used, i.e. banana, apple and bauble (Fig.4).

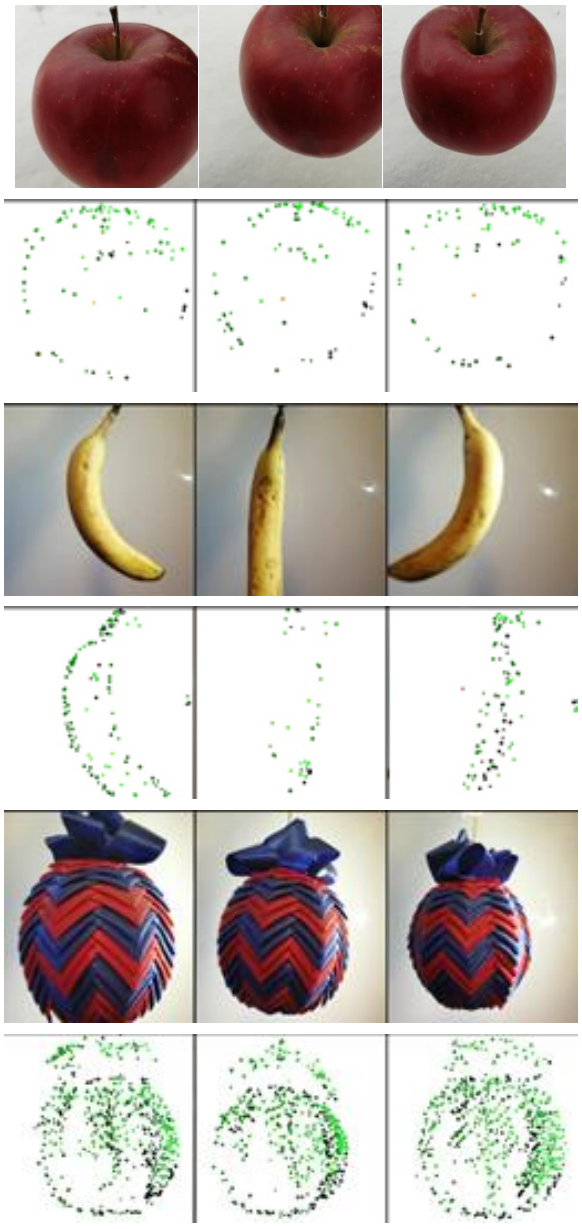


Fig. 4. Example images from natural object image collections and sets of extracted keypoints for these images. Due to the requirement of unpaired sets, keypoints represented images of other images of example objects.

The object of least complexity is represented by the apple. Due to its symmetrical structure, simple texture, and nearly uniform color, it is an ideal test object, subjectively the simplest from a neural network perspective. The object - a banana presents a medium degree of complexity. Asymmetrical structure, irregular color, and varied shape depending on perspective. The most challenging, and subjectively most complex object from a neural network perspective, turned out to be the bauble. The complicated pattern, heterogeneous texture, different colors and lack of symmetry in the particular settings of the bauble, it is an ideal object for testing the limits of the reconstructive capabilities of the neural network.

A loss plot was used as a measure of learning progress. The data in the graph indicate the differences between the expected value representing the image and the result obtained by the discriminator and generator, respectively (Fig. 5).

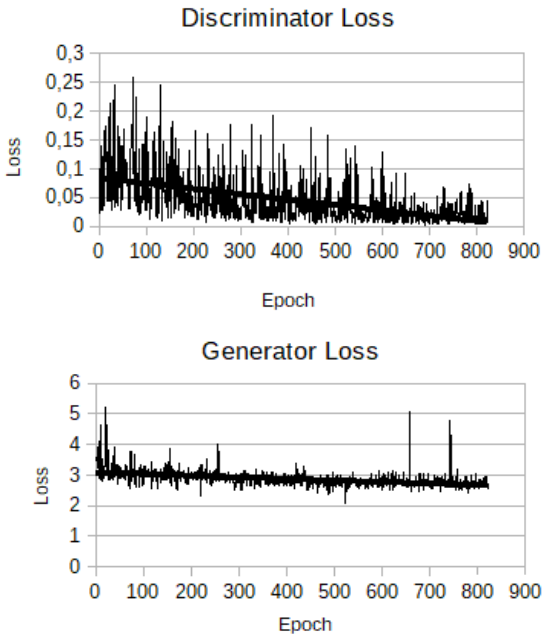


Fig. 5. Discriminator loss graph (top), Generator loss graph (bottom).

The discriminator losses are the mean squared errors between the output of the discriminator, given an image, and the target value, 0 or 1, depending on whether it should classify that image as fake or real. The Generator loss will include cycle consistency loss. This loss is a measure of how good a reconstructed image is when compared to an original image. Training was discontinued based on subjective performance assessment (Fig. 6). The results of the network were promising after 1000 epochs. When the learning process assuming 2000 epochs came to an

end, the results obtained were much more distorted than in the first phases of learning, numerous artifacts appeared and the network, every few epochs, seemed to return to the initial state of the generator and discriminator. Closer analysis of the resulting generative images indicated that an overfitting process was taking place.

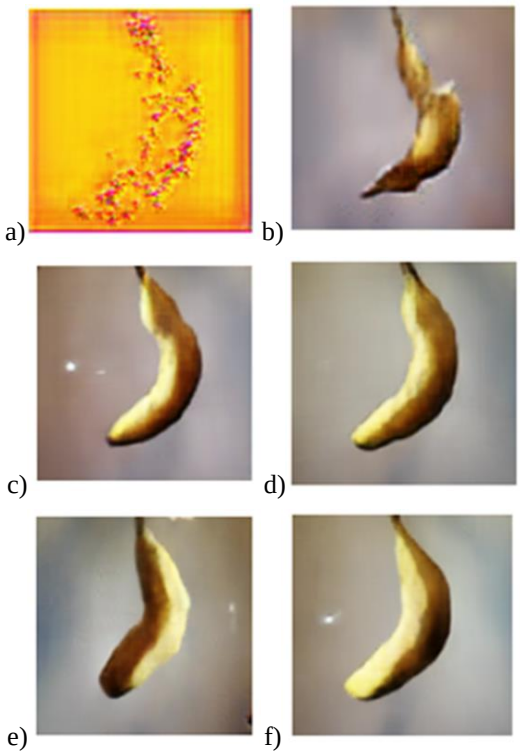


Fig. 6. Examples of consecutive generations of learning a) 0 epoch b) 100 epoch c) 300 epoch d) 800 epoch e) 1200 epoch f) 2000 epoch.

The problem of overfitting

Of the many problems that can occur when training a neural network, one of the most common is the problem of overfitting. This problem, manifests itself in different ways depending on the type of network. In the case of networks designed to classify objects, it manifests itself in the classification of specific types of objects. In case of networks designed to learn features of an image and transfer them to another one, e.g. in case of CycleGAN network, the problem manifests itself in distortions resulting from focusing the learning process on insignificant or extremely isolated features of images. In the case of the prepared image database, the problem consisted in light reflections in the background of objects, which the learning process tried to follow, at the same time moving away from the main goal which was to represent the object well. In order to avoid this error, the object dataset was rebuilt.

The generative images (Fig. 7) are an example of the overfitting problem. Neural network learning was successful in the right direction until about the 1000th epoch, when both the generator and discriminator loss values reached a minimum. After this point, both of the adversarial networks lost their ability to evaluate the image, and the output images were characterized by numerous distortions.

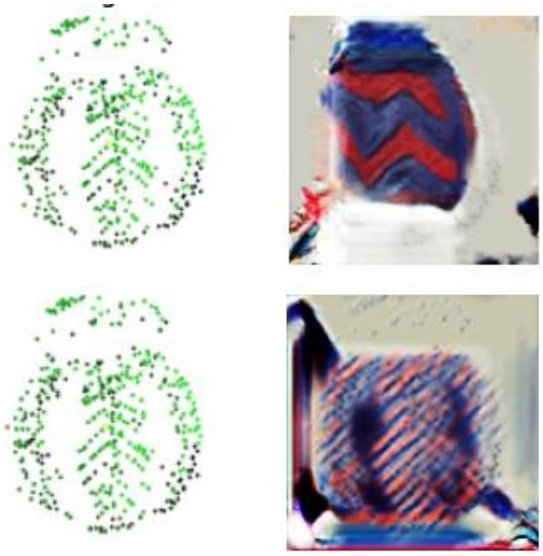


Fig. 7. The effect of network overfitting using the bauble object as an example.

6. QUALITY ASSESSMENT OF THE RESULTS

In order to perform reliable tests on the quality of individual generative images obtained from the GAN network learning process, it was decided to perform a subjective evaluation study in two groups of independent subjects in the form of an environmental questionnaire.

Unfortunately, there is no universal objective testing method for all types of adversarial neural network. We proposed to use the MPEG VCM group methodology for image quality assessment. For this purpose, generative image classification was evaluated using Detectron2 networks.

Subjective evaluation study

The assumptions of the survey were both questions about the degree of reality rendering of implicitly presented original and generative images, and questions explicitly indicating the origin of the image.

The survey culminated with a question testing whether the viewer could identify the real image from the generative images, along with the degree of confidence in the answer.

Two groups of independent people were selected for the subjective evaluation. The first, closed group consisted of 45 people from the community who were partially familiar with the research topic or who were in contact with generative images. The second group contained 44 random people who were not experts in the technique. The individual questions in both groups are as follows:

Question 1: Which image more closely looks like a real apple (scale from 1 to 10, where 1- definitely left, 10- definitely right)?

Fig. 8. The left image represented a real object, the right image is generated based on SIFT keypoints.

Question 2: To what extent does the following picture represent reality (scale from 1 to 10 where 1 is completely unreal, 10 is completely real)?

Fig. 9. From left, generative images created from the SIFT keypoints banana, apple, bauble, and the image representing the real object apple.

Question 3: The image shows 6 photos, one of them is a real photo. Identify which one? With how much certainty would you state your answer (scale of certainty from 1 to 10, where 1-totally uncertain, 10-totally certain)?

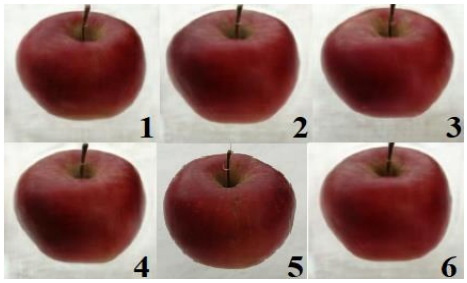


Fig. 10. One of the images above shows a natural image. Image number 5 is a picture of a real apple.

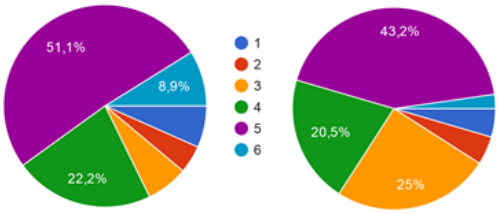


Fig. 11. Distribution of responses expressed in question 3 [%].

Table 1. Distribution of expressed responses in the question 1 [%].

Group 1 (45 person) mean value = 2.62, standard deviation = 1.9 In case we have a scale of answers, they have implicitly assigned weights (1, 2, 3... in order). The average in this case is calculated from the indexes of these responses, e.g. 1*x number of responses + 2*x responses + 3*x responses / number of responses given = mean value.									
37.8	15.6	26.7	6.7	4.4	4.4	0	2.2	2.2	0
Group 2 (44 person) mean value = 3.32, standard deviation = 2.4									
31.8	9.1	20.5	18.2	4.5	2.3	4.5	4.5	2.3	2.3

Table 2. Distribution of expressed responses in the question 2[%].

Group 1 mean value = 7.2, standard deviation = 1.9,
Group 2 mean value = 7.66, standard deviation = 2.03.

Group 1 (45 person) (1st object - banana)									
4.4	6.7	24.4	22.2	13.3	6.7	13.3	6.7	0	2.2
Group 2 (44 person)									
2.3	2.3	15.9	20.5	9.1	20.5	11.4	13.6	2.3	2.3
Group 1 (45 person) (2nd object - apple)									
0	0	2.2	11.1	6.7	13.3	17.8	15.6	26.7	6.7
Group 2 (44 person)									
0	2.3	4.5	0	9.1	6.8	15.9	20.5	22.7	18.2
Group 1 (45 person) (3rd object - bauble)									
8.9	6.7	17.8	17.8	11.1	13.3	6.7	11.1	4.4	2.2
Group 2 (44 person)									
11.4	9.1	4.5	2.3	18.2	6.8	20.5	11.4	6.8	9.1
Group 1 (45 person) (4th object – real apple)									
0	4.4	4.4	0	4.4	8.9	17.8	13.3	24.4	22.2
Group 2 (44 person)									
0	2.3	4.5	4.5	4.5	2.3	11.4	18.2	18.2	34.1

Table 3. Certainty of the question 3 answers [%].

Group 1 (45 person) mean value = 7.71, standard deviation = 2.19									
0	0	8.9	24.4	11.1	8.7	13.3	22.2	8.9	4.4
Group 2 (44 person) mean value = 8.02, standard deviation = 2.23									
6.8	13.6	11.4	6.8	6.8	15.9	13.6	13.6	6.8	4.5

After a preliminary analysis of the survey results, a divergence of responses can be observed in all questions. The distribution of the answers of question 1 (Table 1) indicates that about 47% of the respondents of group one and about 61% of group two are not sure or almost sure (Indication 1 or 2) that the

image on the right is real (Fig. 8). Such a result can be considered satisfactory, due to the fact that the generative image created for the purpose of this paper proved to be authentic enough to introduce doubt. As expected, the image ratings of individual objects are proportional to the subjective complexity of these objects (Table 2). The evaluation of the generated image representing a banana (Fig. 9) indicates that in group one, almost 30% of the respondents indicated that the image was closer to the real one (ratings of at least 6), while in group two, 50% of the respondents indicated such a rating. The generated image showing apple (Fig. 9) is closer to the real one for 80% of the respondents of group one and 84% of group two. As expected, the subjectively unsatisfactory generated images depicting bauble were rated as closer to reality for 38% of the first group and 55% of the second group respondents. Surprisingly, despite expectations, the image depicting a banana object rather than a bauble was rated worst among respondents. A large majority of the respondents, even after seeing the actual image, responded uncertainly or doubted completely the authenticity of the image they saw. This is indicated by 53%, in the first group, and 47%, in the second group, of ratings that can be classified as unsure of authenticity (Ratings less than 9). The culmination of the survey was to find the real image among the group of generative images – question 3 (Fig. 10), and to indicate the certainty of the answer (Fig. 11). The image was correctly indicated by 51.1% of the respondents in group one and 43.2% in group two. This result is much higher than expected and at the same time very satisfactory, moreover the correct answer with confidence not less than 9 was declared by 11% of respondents in both groups. This result together with the above indicates that subjectively, the generative images generated in the study based on SIFT keypoints are close enough to reality that the results can be considered successful.

Additionally, respondents were asked to identify any distortions, artifacts, and any unreal anomalies perceived in the images. In the banana object group, respondents most frequently identified an unnatural shape, an overly sharp bend in the banana, an unnatural texture, and an unreal shadow. In the group of objects representing an apple, many respondents indicated that the edges were unnaturally blurred. In the case of this object, there were also many opinions about there being nothing unreal in the image. The distortions mentioned above can be eliminated by increasing the training sets and longer network learning time.

Evaluation of classification quality using Detectron2 networks

The objective results have been obtained using COCO Evaluation Framework val. 2017 dataset [Lin14]. The methodology of experiment follows the recommendations described in Evaluation Framework for VCM document [Vcm20]. The evaluation have been done using the neural networks for object classification the Detectron2 R-CNN X101-FPN from Facebook Research Detectron2 project [Wu19].

The quality of the learning process was also verified by evaluating image classification using Detectron2 network (Fig.12).

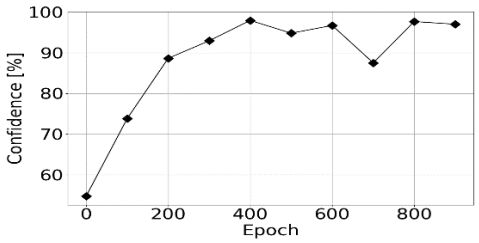


Fig. 12. Confidence of generative object image recognition using Detectron2 network as a function of CycleGAN network learning process.

The Detectron2 network along with the COCO test object set recognizes from the prepared test set real apple type objects with a confidence of 97.87% with a standard deviation of 1.27%, banana type objects with a confidence of 98.99% and a standard deviation of 1.41%. The network classified objects from generative images as apples with a confidence of 97.19% with a standard deviation of 0.4%. For banana objects, this result was 98.7% with a deviation equal to 1.12% (Fig. 13). So the result is only slightly worse than for images representing real objects. Unfortunately the COCO database does not contain any bauble type objects.

Fig. 13. An example of a result from the Detectron2 network

7. CONCLUSIONS

This paper presents results of image reconstruction based on a set of SIFT keypoints using the learned CycleGAN network. Both objective results (obtained through the process of classifying generative images and comparing the results to those of real images using the Detectron2 network) and subjective results (in the form of questionnaires and a series of questions in two

target groups) confirm that cross-domain translation between SIFT keypoints and images is possible. Moreover, the results are satisfactory despite the lack of use (by assumption) of descriptors describing the local neighborhood of the keypoints. This is a good starting point for further research on how to represent keypoints in feature maps of the learning set of a generative network.

8. ACKNOWLEDGMENT

This work was supported by the Ministry of Education and Science of Republic of Poland.

9. REFERENCES

[Deso17] A. Desolneux, A. Leclaire, "Stochastic Image Reconstruction from Local Histograms of Gradient Orientation" in Proceedings of the International Conference on Scale Space and Variational Methods in Computer Vision, 2017.

[Deso18] A. Desolneux, A. Leclaire, "Stochastic Image Models from SIFT-like descriptors", SIAM Journal on Imaging Sciences, 2018.

[Dua15] L. Duan, T. Huang and W. Gao, "Overview of the MPEG CDVS Standard," 2015 Data Compression Conference, Snowbird, UT, USA, 2015, pp. 323-332.

[Gat16] Gatys L. A., Ecker A. S., and Bethge M., "Image style transfer using convolutional neural networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2414–2423.

[Goo14] Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., Generative Adversarial Networks, arXiv: 1406.2661, 2014.

[Goo16] Goodfellow I. J, Bengio Y., Courville A., "Deep Learning" , MIT Press, 2016.

[Iso17] Isola P., Zhu J.-Y., Zhou T., and Efros A., "Image-to-image translation with conditional adversarial networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[Kin14] Kingma DP, Ba J, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014.

[Lin00] Linder-Norén E., CycleGAN https://github.com/eriklindernoren/Keras-GAN

[Lin14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft COCO: Common objects in context," In ECCV, 2014.

[Low04] Lowe D. G., Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, 60(2), 2004, pp91-110.

[Mpe17] ISO/IEC 15938-15, "Information Technology - Multimedia Content Description Interface - Part 15: Compact Descriptors for Video Analysis", 01.12.2017.

[Mpe20] "Use cases and requirements for Video Coding for Machines," Doc. ISO/IEC JTC1/SC29/WG11 N19506, June 2020.

[Mpe20b] "Draft Evaluation Framework for Video Coding for Machines," Doc. ISO/IEC JTC1/SC29/WG11 N19507, June 2020.

[Pas12] S. Paschalakis et al., "Information technology - multimedia content descriptor interface – part 13: Compact descriptors for visual search," in ISO/IEC DIS 15938-13.

[Sas19] Sashank J. Reddi, Satyen Kale, Sanjiv Kumar, On the Convergence of Adam and Beyond, arXiv:1904.09237, 2019.

[Wein11] P. Weinzaepfel, H. Jegou, and P. Perez, "Reconstructing an image from its local descriptors", Computer Vision and Pattern Recognition, IEEE, June 2011.

[Wu19] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo and R. Girshick. Detectron2, Faster R-CNN X101-FPN, 2019.

https://github.com/facebookresearch/detectron2/blob/master/configs/COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml

[Wu19b] H. Wu, J. Zhou and Y. Li, "Image Reconstruction from Local Descriptors Using Conditional Adversarial Networks," 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Lanzhou, China, 2019, pp. 1773-1779.

[Wu20c] Wu H. and Zhou J., Privacy Leakage of SIFT Features via Deep Generative Model based Image Reconstruction, arXiv: 2009.01030, 2020.

[Vcm20] ISO/IEC JTC 1/SC 29/WG 2, "Evaluation Framework for Video Coding for Machines", Doc. N19, October 2020.

[Vond13] C. Vondrick, A. Khosla, T. Malisiewicz, A. Torralba, „HOGgles: Visualizing Oject Detection Fetures", in Proc. IEEE Int. Conf. Computer Visoin, 2013, p. 1-8.

[Zhu20] Zhu J.-Y., Park T., Isola P., Efros A., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, arXiv: 1703.10593, 2020.

Exploration of U-Net in Automated Solar Coronal Loop Segmentation

Shadi Moradi
Computer Science Dept.
Bowling Green State Univ.
Bowling Green, OH 43403
U.S.A.
smoradi@bgsu.edu

Jong Kwan Lee
Computer Science Dept.
Bowling Green State Univ.
Bowling Green, OH 43403
U.S.A.
leej@bgsu.edu

Qing Tian
Computer Science Dept.
Bowling Green State Univ.
Bowling Green, OH 43403
U.S.A.
qtian@bgsu.edu

ABSTRACT

This paper presents a deep convolutional neural network (CNN) based method that automatically segments arc-like structures of coronal loops from the intensity images of Sun's corona. The method explores multiple U-Net architecture variants which enable segmentation of coronal loop structures of active regions from NASA's Solar Dynamic Observatory (SDO) imagery. The effectiveness of the method is evaluated through experiments on both synthetic and real images, and the results show that the method segments the coronal loop structures accurately.

Keywords

Convolutional Neural Network, U-Net, Segmentation, Solar Physics Application

1 INTRODUCTION

Feature segmentation has been studied in many scientific and application domains (e.g., [**bps05**, **cng09**, **slh13**, **bbb18**, **lel18**, **zll19**, **zmo20**]) as it supports new findings for the feature of interests or other further important information discovery. For example, Cao et al. [**cng09**] introduced a new randomized Hough transform method that enabled segmentation of aurora structures which can help other geophysical studies. A medical application (i.e., segmentation of brain tumor) of random forest and level set methods was recently presented [**lel18**]. In this paper, we consider the automated feature segmentation in the solar physics application.

The study of solar activities is of great importance for understanding their effects on Earth. For example, these activities can impact terrestrial communication or geospace weather. The solar scientists gain insights of these solar activities by examining the curvilinear features that appear in the outermost layer of the Sun's atmosphere. These curvilinear features are called the *coronal loops*. These are the traces of the solar magnetic field that drives the Sun's activities [**lng06**]. The solar scientists study the coronal loops from solar imagery to analyze scientifically important phenomena, such as the controversial coronal heating [**ssc02**, **scb03**], coronal mass ejections [**pjh12**], or solar storms [**gbm90**]. Figure 1 shows an example of a solar image with a close up on coronal loops. In the figure, the white arching structures are the coronal loops. Automated segmentation of these coronal loops can greatly help the solar scientists enable an efficient and consistent analysis of a large number of coronal images. Nevertheless, automated coronal loop segmentation is very challenging since the coronal loops have irregular shapes, widths, and orientations, as well as varying intensities without clear outer boundary. In addition, the loops overlap and there are image noises.

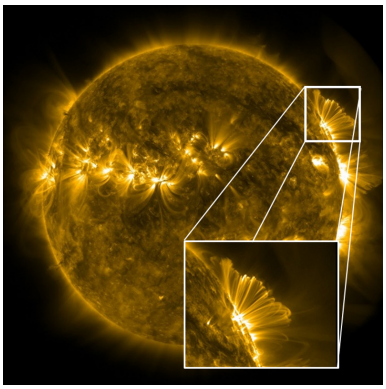


Figure 1: An example of solar image with a close up on a coronal loop region. (We have added the close up part of the image from NASA's SDO solar image.)

In this paper, we present a deep convolutional neural network-based method for the coronal loop segmentation. The method employs a U-Net architecture [**rfb15**]

by considering different depths and widths in the arcitecture for its accurate segmentation of coronal loops. To our knowledge, our method is the first attempt to utilize deep learning for coronal loop segmentation.

This paper is organized as follows. In Section 2, the related work is discussed. Section 3 describes our new method of coronal loop segmentation. The experimental results of applying the new method to synthetic data and real coronal images are presented in Section 4. Section 5 concludes this paper and discusses future work.

2 RELATED WORK

Processing of satellite and observatory-collected imagery (e.g., [**and99**, **kng04**]), including coronal loop segmentations, has been investigated by many studies. In this section, the related work in convolutional neural networks and the prior efforts on automated coronal loop segmentation are discussed.

2.1 Convolutional Neural Networks for Image Segmentation

In the deep learning era, convolutional neural networks (CNNs) have achieved state-of-the-art performance on a wide variety of challenging tasks, such as classification (e.g., [**ksh12**, **kaz15**, **slj15**]), object detection (e.g., [**gdd14**, **est14**]), and object segmentation (e.g., [**zjr15**, **bhc15**]. Among them, image segmentation is usually considered to be the most difficult one. Instead of classifying one image-level label or a limited number of bounding boxes, image segmentation requires a model to predict one label for each image pixel. In other words, image segmentation is the process of partitioning an image into multiple segments (sets of pixels). In most cases, a segmentation model directly takes an entire image as its input and outputs its segmentation result in the form of pixel mask(s). Specifically, segmentation tasks can be further categorized into semantic segmentation and instance segmentation. Semantic segmentation involves predicting image pixels to defined categories. That is to say, objects of the same class are treated as a single entity. On the other hand, instance segmentation predicts each pixel to belong to distinct individual objects/instances. In this paper, we focus on semantic segmentation– we aim to determine whether each pixel is either coronal loop pixel or not.

Various deep CNN architectures have been proposed for image segmentation, such as SegNet [**bhc15**] (which is a deep encorder-decoder architecture), Piecewise CRF-CNN [**lsv16**] (which exploits piecewise training based on fully convolutional neural networks (FCNN) [**lsd15**]), efficient neural network (ENet) [**pck16**] (which is a compact encoder-decorder architecture), and MaskRCNN [**hgd17**] (which extends Faster R-CNN by adding a branch for predicting an object masks in parallel). These segmentation networks are used in many fields such as medical image analysis [**kk16**], real-time segmentation of dash cam video using ENet [**pck16**], and satellite image segmentation using SegNet [**bhc15**]. One of the most widely used CNNs for segmentation is the U-Net [**rfb15**]. It is usually used for medical image segmentation, such as retina blood vessels detection in retina fundus images [**ru18**] and brain tumor detection in MRI images [**dyl17**].

2.2 Coronal Loop Segmentation

The methods by Lee et al. (i.e., Oriented Connectivity-based Method (OCM) and Daynamic Aperture-based Method (DAM)) [**lng06a**, **lng06b**] were the very first two coronal loop segmentation methods. OCM [**lng06a**] utilized external estimates of the magnetic fields' local orientation to enable a constructive edge linking-based coronal loop segmentation. DAM [**lng06b**] exploited the Gaussian-like shape of loop's cross-sectional intensity profiles. In particular, DAM segments the loops via a search through the image for regions whose intensity profiles are well fitted by a Ruled Gaussian Surface (RGS). The method joins (i.e., to construct loops) fitted RGSs only if they have similar orientation, which is determined by applying principal component analysis on the RGSs.

Aschwanden [**asc10**] presented the oriented-directivity loop tracing method [**asc10**], which exploited directional information for guiding the tracing of the loops, similar to the OCM, but makes use only of the local directivity and the curvature radius constraints in corona loop tracing. The curvature radius constraint enabled loop tracing by providing estimates of loop direction range based on previously-traced loop segment.

There are other studies that included coronal loop segmentation. Durak et al. [**dns09**, **dns10**] developed a solar loop mining system for coronal loop segmentation, which included a block-by-block loop segmentation to retrieve images with coronal loops from a large number of solar image datasets. McAteer et al.'s 2D Wavelet-Transform Modulus Maxima Method [**mka10**] exploited the derivative of a 2D Wavelet-based smoothing function as an edge detector in segmenting coronal loop structures.

Other traditional computer vision-based method for coronal loop segmentation included the Snake-based approach [**let11**]. The approach used a greedy minimization-based active contour models (snakes) for coronal loop segmentation. Specifically, the Gaussian-like shapes of the coronal loop's cross-sectional intensity profile were used to refine the snake's position.

Recently, Zhiming [**zxz19**] proposed a new coronal loop segementation method. The method exploited a clustering algorithm using approximated local directionality based on a match and image enhancing filters.

3 DEEP CORONAL LOOP SEGMENTATION

In this section, we introduce the new deep learning-based coronal loop segmentation method.

Our method exploits variants of the U-Net [**rfb15**] to segment coronal loop structures. The U-Net was developed by Ronneberger et al. for biomedical image segmentation. There are some similarities between solar images and biomedical images. For example, both kinds of images usually contain complicated fine-grained patterns of the imaged features (e.g., brain vessels and solar loops). Specifically, we adopt U-Net like architectures due to their following advantages for segmentation:

- To segment delicate object patterns, the U-Net makes use of skip-architectures which combine deeper high-level features from decoding layers with relatively low-level features from encoding layers. At the same time, the location and context information from the down-sampling and up-sampling paths are better combined, which plays a key role in predicting a good segmentation map.
- Since there are no dense layers, images of different sizes can be used as input (unlike fully connected layers, convolutional kernels are independent of input image's size).
- The U-Net architecture can lead to precise segmentations with small numbers of training images [**rfb15**].

In our method, variants of U-Net architecture were exploited. Specifically, different depths and widths in the architecture layers were considered. This inspiration was from [**tl19**] where the authors experimented with model scaling on MobileNets and ResNets for image classification tasks. They identified that carefully balancing network depth, width, and resolution can result in improvement of classification accuracy. However, in our method, the original resolution was kept in order not to lose any image information nor to inject extra (and usually unreliable) information through interpolation.

To be more specific, our method utilized varying architectures of U-Net by using deeper or shallower variants, which included more or fewer layers, respectively. In addition, the width of the U-Net was varied by changing the number of filters in each convolutional layer. The variation in network width and depth allowed the network to better extract coronal loops while maintaining high generalization ability (i.e., without suffering from overfitting). Here, we note that the input image size used in this study was 512×512.

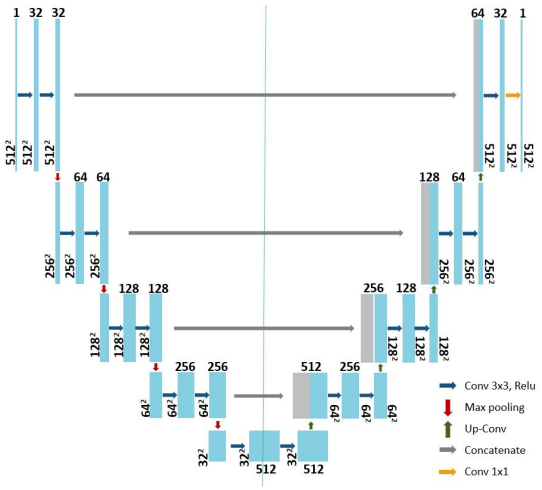


Figure 2: U-Net A: variation in width showing a **narrower** network. The dimensions of each layer and the operations are shown.

Our first U-Net architecture is shown in Figure 2. We denote this network the *U-Net A*. The down-sampling path has five convolutional blocks. Every block has two convolutional layers (stride of 1) before non-linear rectifier activation. Over the first five blocks, while the number of 3×3 filters (also the number of 2D feature maps) is increased from 32 to 512. For the down-sampling, max pooling with a stride of 2×2 is applied to the end of every block except for the last one, so the size of feature maps decreases from 512×512 to 32×32. In the up-sampling path, every block starts with a deconvolutional layer with filter size of 3×3 and stride of 2×2, which doubles the size of feature maps in both dimensions but reduces the number of feature maps by two. Over the upsampling path, the size of feature maps increases from 32×32 to 512×512. It is worth noting that at the start of every up-sampling block, the feature maps from the last block are concatenated with the feature maps produced by the corresponding encoding block to form this block's input. This operation is shown as gray "concatenate" arrows in Figure 2. When necessary, we used zero padding to keep the dimensions of the 2D feature maps to be concatenated the same. This is different from the original U-Net architecture. Each of the following convolutional layers reduces the number of feature maps by two. Finally, a 1×1 convolutional layer is used to reduce the number of feature maps to one, which corresponds to a binary map, one for foreground and another for the background.

Our second U-Net architecture is shown in Figure 3. We denote this network the *U-Net B*. The down-sampling path has four convolutional blocks. Every block has two convolutional layers (filter size: 3×3, stride: 1) and a non-linear rectification. The down-sampling path increases the number of feature maps from 1 to 256. As in the down-sampling path

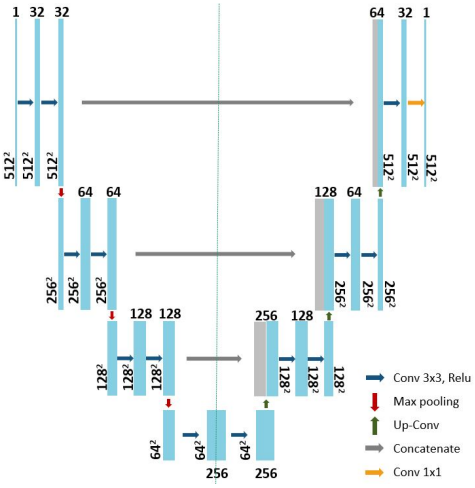


Figure 3: U-Net B: variation in width showing a **shallower and narrower** network. The dimensions of each layer and the operations are shown.

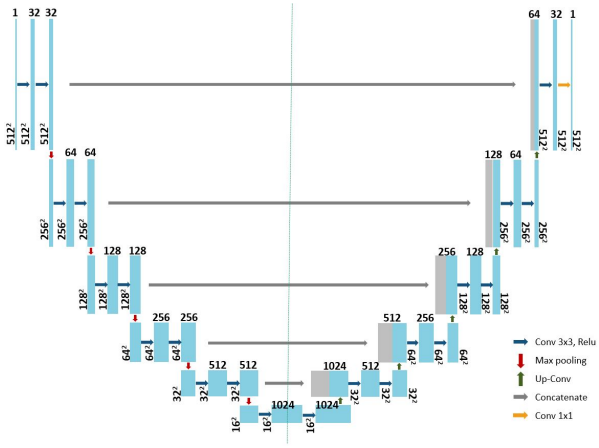


Figure 4: U-Net C: variation in width showing a **deeper** network. The dimensions of each layer and the operations are shown.

of the previous architecture, max pooling with a stride of 2×2 is applied to the end of every block except for the last one. As a result, the size of feature maps decreases from 512×512 to 64×64. In the up-sampling path, every block starts with a deconvolutional layer with a filter size of 3×3 and a stride of 2×2. The upsampled features are then concatenated with the feature maps generated by the corresponding encoder block. The concatenation process and later convolutions are similar to the previous architecture. Each up-sampling block doubles the dimension while halves the number of feature maps. From the first decovolution/upsampling block to the end, the dimension of feature maps increases from 64×64 to 512×512.

Our third network is shown in Figure 4. We denote this network the *U-Net C*. The down-sampling path has six convolutional blocks. Every block has two convolutional layers with a filter size of 3×3 and a stride of 1, followed by non-linear rectification. The down-

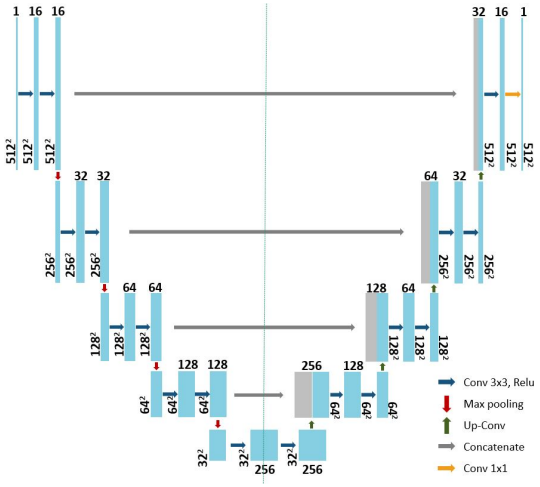


Figure 5: U-Net D: variation in width showing a **narrower** network. The dimensions of each layer and the operations are shown.

sampling path increases the number of feature maps from 1 to 1024. Similar to the previous architecture, max pooling with stride 2×2 is applied at the end of every down sampling block except for the last one. Through the down-sampling path, the dimension of feature maps decreases from 512×512 to 16×16. In the reverse up-sampling path, similar to previous architectures, the dimension of feature maps increases from 16×16 back to 512×512.

Finally, our forth network architecture is shown in Figure 5. We denote this network the *U-Net D*. The down-sampling path has five convolutional blocks. Every block has two convolutional layers (with a filter size of 3×3, stride of 1) and non-linear rectifier activation. This can be considered as a skinnier version of U-Net A. The convolutional layers in the first encoder block have only 16 filters. The number of filters in each convolutional layer doubles every time we advance to the next block. As a result, the up-sampling path increases the number of feature maps to 256.

4 RESULTS

In this section, the results of our method are presented. The experimental setup is discussed first followed by the experimental results on synthetic data and real images with their analysis. Our experiments were tested on a Windows OS PC with 2.6 GHz, Intel Core i7 CPU and 16 GB RAM. We have test over 3,000 synthetic images and over 150 real images in our experiments. Specifically, we have used each 150 images as a set with 90% of the images for training/validation, and 10% of testing.

4.1 Experimental Setup

4.1.1 Training Details

The input images and their corresponding segmentation ground truth masks were used to train the network. During the training process, the binary cross-entropy was used as the cost function of the network since there are only two classes (i.e., coronal loop and non-coronal loop). The adaptive moment estimator (Adam) [**kib14**] was used to minimize the cost function. In general, Adam utilizes the first and second moments of gradients for updating and correcting moving average of the current gradients. The parameters of our Adam optimizer were set as: learning rate = 0.002 and the maximum number of epochs = 500. We used Keras callbacks to implement learning rate decay and to reduce it to 0.0002. Early stopping was implemented when the validation loss did not improve for 5 consecutive epochs. Only best weights were saved. We used a batch size of 4 for training. (In our experiments, the training took about 2 hours and about 5 hours for synthetic and real images, respectively. Here, we note that we did not utilize any GPUs in this work.)

4.1.2 Evaluation Metrics

The evaluation has been done using accuracy and recall. Accuracy is the ratio of predictions our model predicted correctly. Recall refers to the ratio of total relevant results correctly classified by our method.

4.1.3 Thresholding

After training, a network produces a value between 0 to 1 for each pixel. 0 represents non-loop pixel and 1 represents for coronal loop pixel. Upon examination of accuracy, recall, and precision on our synthetic data, the sum of the standard deviation and the mean of all predicted values was used as the threshold to decide whether to classify a pixel as non-loop or coronal loop pixel. (Figure 6 shows the effect of thresholding on various segmentation metrics.) For real solar images, since there exists a large variation in the background and non-loop intensity, we adopted adaptive local thresholding [**srs11**] (also known as dynamic thresholding). It determines thresholds in regions with a characteristic size "block size" surrounding each pixel (i.e., local neighborhoods). Each threshold value is the weighted mean of the local neighborhood minus an offset value. We used a block size of 21.

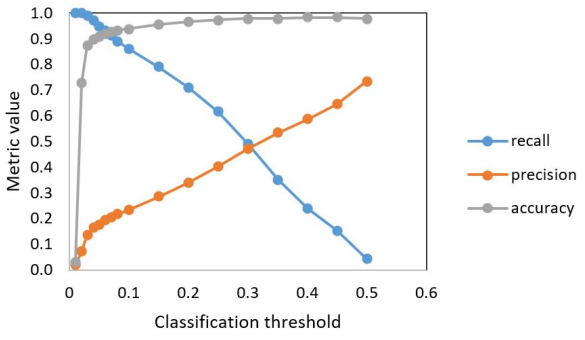


Figure 6: Effect of threshold on various segmentation metrics.

4.2 Synthetic Data Tests

For synthetic data tests, we have simulated the coronal images using various combinations of image parameters. In our data, randomly generated parabola curve sets with (1) varying number of curves, (2) varying curve widths, (3) varying combinnation of image noises (i.e., Gaussian noise with different stanadard deviations and salt & peper noise) were considered.

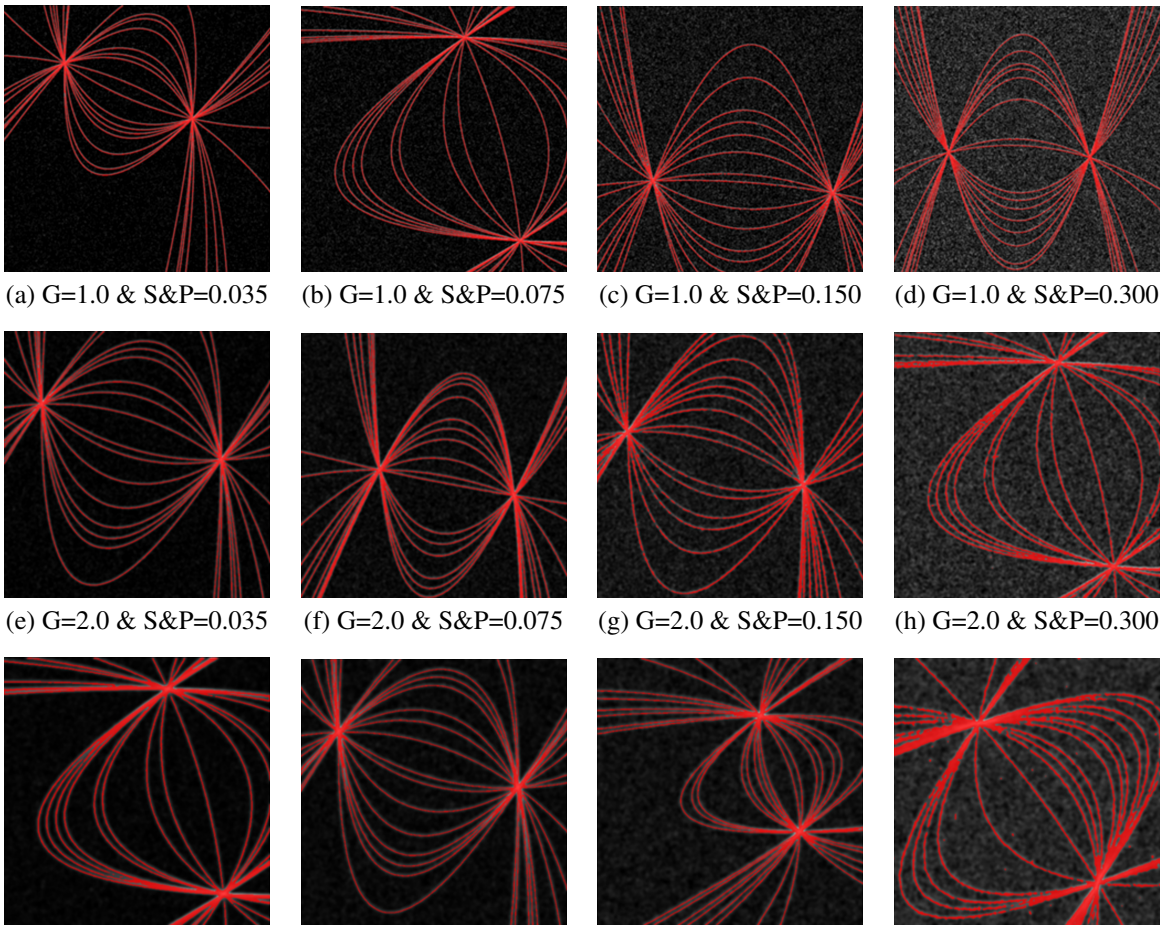
Figure 7 shows an example set of synthetic data segmentation results. In the figure, segmentation results with higher noise levels are shown from left column to right and from top row to bottom (e.g., subfigure (a) has the least amount of noise and subfigure (l) has the highest amount of noise) and the U-Net segmentation results are shown as the red overlays. As shown in the figure, while some mislabel of loops pixels were shown when the noise levels were high, our method segmented the loops very accurately.

Figure 8 summarizes the performance metric evaluation for image parameters. Figure 8 (a) shows the accuracy and recall when different number of loops/curves were used. While slightly lower performance was shown when more number of loops were considered, the U-Net still achieved very high accuracy and recall. Figure 8 (b) and (c) show the performance metric surfaces for noise levels. As shown in the figures, the U-Net achieved very high accuracy and recall. (Here, we note that the false positives were about 0.014 on average with its standard deviation of 0.002.)

Table 1 lists the performance metrics on synthetic data for different numbers of layers and filters in the network. As shown in the table, the U-Net achieved very high accuracy and recall, and acceptable precision. The performance was the highest when six layers and 32 filters were used in the U-Net.

Table 1: Accuracy (Acc.) & Recall (Re.c) & Precision (Pre.) on Synthetic Data: U-Net B on First Row, U-Net D on Second Row, U-Net A on Third Row, U-Net C on Fifth Row

Data	Layers	Filters	Acc.	Rec.	Pre.
Syn. Data	4	32	0.97	0.85	0.81
	5	16	0.98	0.91	0.76
	5	32	0.97	0.82	0.80
	5	64	0.97	0.80	0.79
	6	**32**	**0.98**	**0.96**	**0.81**



(a) G=1.0 & S&P=0.035 (b) G=1.0 & S&P=0.075 (c) G=1.0 & S&P=0.150 (d) G=1.0 & S&P=0.300

(e) G=2.0 & S&P=0.035 (f) G=2.0 & S&P=0.075 (g) G=2.0 & S&P=0.150 (h) G=2.0 & S&P=0.300

(i) G=3.0 & S&P=0.035 (j) G=3.0 & S&P=0.075 (k) G=3.0 & S&P=0.150 (l) G=3.0 & S&P=0.300

Figure 7: Results on synthetic data with varying noise levels: Gaussian (G) noise and Salt & Pepper (S&P) noise

4.3 Real Image Tests

For real image tests, we have used the coronal loop images from the NASA's SDO satellite. Specifically, we have cropped the Active Regions (i.e., where most coronal loops appear) of size 512×512 from the SDO's full disk solar image of size 4096×4096. For the training of network, we have used the manually-traced coronal loops as the ground truth masks. (The use of the manually-traced masks is typical when there is no true ground truth in real images.) We have also found that using unsharpmasked images, instead of raw images, produced the better performance results.

Figure 9 shows the test results of two real coronal images. Subfigures (a) and (e) are the raw images, (b) and (f) are the unsharpmasked images, (c) and (g) are the U-Net segmented loops, and (d) and (h) are the images with segmented result overlays in red, respectively. As shown in the figure, the U-Net produced very accurate segmentation results.

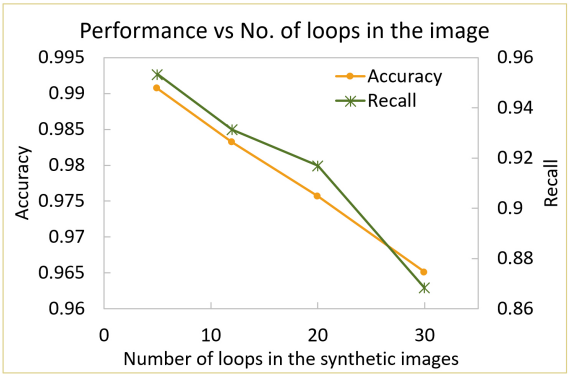
Table 2 lists the performance metrics on real coronal image for different numbers of layers and filters in the network. As shown in the table, the U-Net achieved very high accuracy and reasonable recall. The performance was the highest when six layers and 32 filters were used in the U-Net.

Table 2: Accuracy & Recall on Real Images: U-Net B on First Row, U-Net D on Second Row, U-Net A on Third Row, U-Net C on Fifth Row

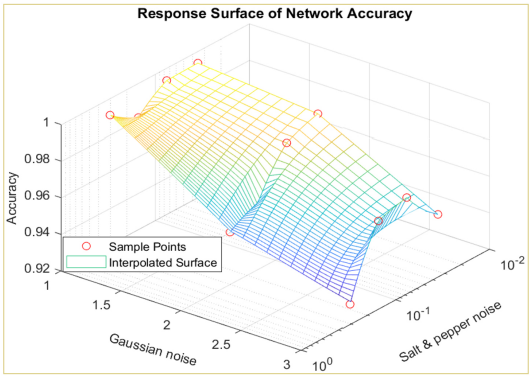
Data	Layers	Filters	Accuracy	Recall
Real Images	4	32	0.86	0.53
	5	16	0.87	0.51
	5	32	0.86	0.53
	5	64	0.87	0.51
	6	**32**	**0.90**	**0.55**

5 CONCLUSION

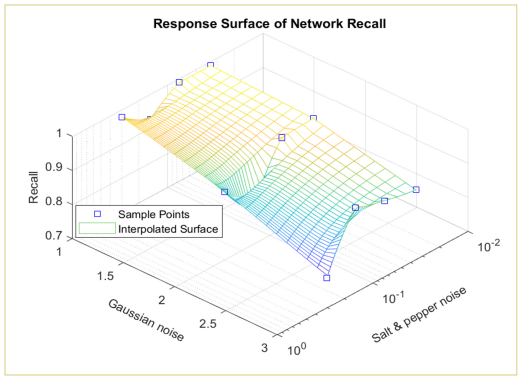
In this paper, the U-Net-based deep convolutional neural network for solar coronal loop segmentation was explored. Our method is the first attempt to utilize a deep learning model in coronal loop segmentation.



(a) Performance On # of Loops/Curves

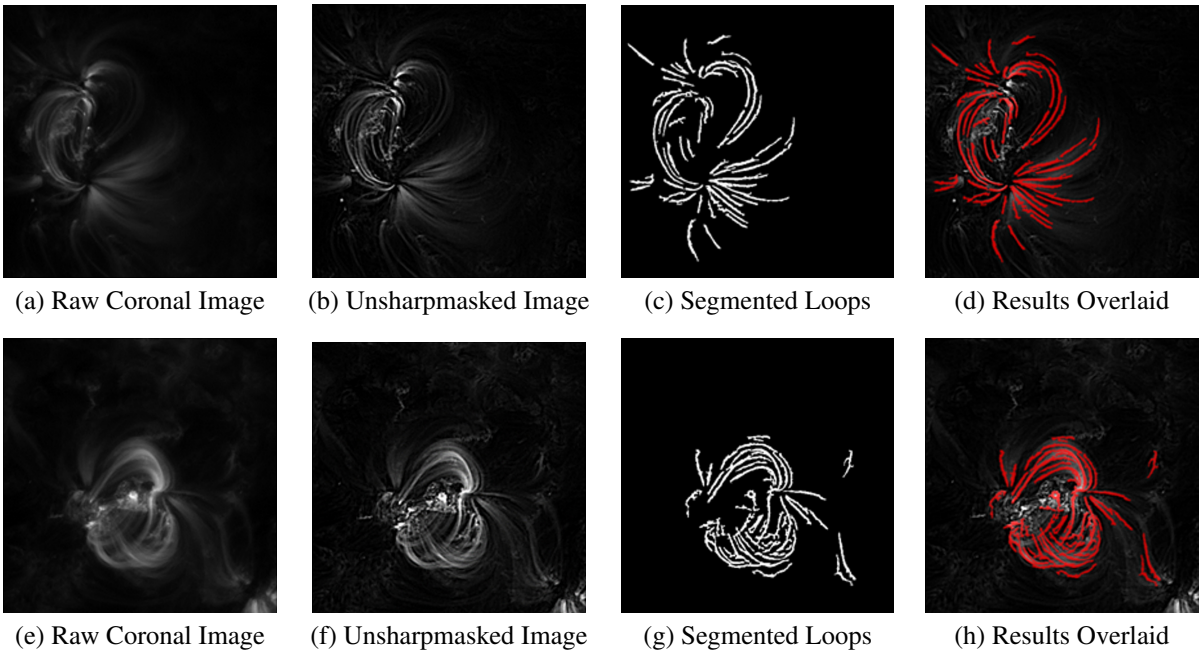


(b) Accuracy on Noise



(c) Recall on Noise

Figure 8: Analysis on Synthetic Data Tests



(a) Raw Coronal Image (b) Unsharpmasked Image (c) Segmented Loops (d) Results Overlaid

(e) Raw Coronal Image (f) Unsharpmasked Image (g) Segmented Loops (h) Results Overlaid

Figure 9: Results on real coronal image: (a, e) raw images, (d, f) unsharpmasked images, (c, g) segmented loops, (d, h) segmented results overlaid in red on image

The method explored the variants of the U-net architecture to enable accurate segmentation of coronal loops. Through evaluation of the method, we have shown that the U-Net can provide consistent and accurate segmentation results of the loops. On synthetic data, the method achieved about 0.98 accuracy and 0.96 recall. On real coronal images, the method achieved about 0.90 accuracy and 0.55 recall.

In the future, we plan to investigate ways to further improve the performance, and perform effectiveness comparisons with other coronal loop segmentations. In addition, the method will be considered for high performance computing (e.g., GPU processing). We also hope to explore applications of our U-Net to other scientifically-interesting features that exhibit similar image characteristics.

6 ACKNOWLEDGMENTS

This research was supported in part by the David and Amy Fulton Endowed Professorship in Computer Science at Bowling Green State University. We also thank the reviewers for their valuable comments which improved our paper.

7 REFERENCES

[BPS05] Bleser G., Pastamov Y., and Stricker D., Real-time 3D Camera Tracking for Industrial Augmented Reality Applications, in Proc., 13th Int'l Conf. in Central Europe on Comp. Graphics, Vis. and Computer Vision (WSCG 2005), Jan. 31–Feb. 4, 2005, pp. 47–54.

[CNG09] Cao, C., Newman, T.S., and Germany, G.A., New Shape-based Auroral Oval Segmentation Driven by LLS-RHT, Pattern Recognition, Vol. 42 (5), 2009, pp. 607–618.

[SLH13] Smith, A.H., Lee, J.K., Hu, H., and Mandell, E.S., Hough Transform-based Technique for Automated Carbon Nanocone Segmentation, in Proc., 21st Int'l Conf. in Central Europe on Comp. Graphics, Vis., and Computer Vision (WSCG 2013), June 2013, pp. 19–28.

[BBB18] Brahimi, S., Ben Aoun, N., Ben Amar, C., Benoit, A., and Lambert, P., Multiscale Fully Convolutional DenseNet for Semantic Segmentation, Journal of WSCG, Vol. 26 (2), 2018, pp. 104–111.

[LeL18] Lefkovits, L., and Lefkovits, S., Two-phase MRI brain tumor segmentation using Random Forests and Level Set Methods, in Proc., 26th Int'l Conf. in Central Europe on Comp. Graphics, Vis., and Computer Vision (WSCG 2018), May–June 2018, pp. 152–159.

[ZLL19] Zhao, Y., Lin, L., Liu, S., and Zhuhua, H., Constrained-Focal-Loss Based Deep Learning for Segmentation of Spores, IEEE Access, Vol. 7, Nov. 2019, pp. 165029–165038.

[ZMO20] Zhang, Y., Ma, Y., Omrani, A., Yadav, R., Fjeld, M., and Fratarcangeli, M., Automated Microwave Tomography (MWT) Image Segmentation: State-of-the-Art Implementation and Evaluation, in Proc., 28th Int'l Conf. in Central Europe on Comp. Graphics, Vis., and Computer Vision (WSCG 2020), May 2020, pp. 126–136.

[LNG06] Lee, J.K., Newman, T.S., and Gary, G.A., Oriented Connectivity-based Method for Segmenting Solar Loops, Pattern Recognition, Vol. 39 (2), 2006, pp. 246–259.

[SSC02] Schmelz, J.T., Scopes, R.T., and Cirtain, J.W., Determining Coronal Heating of Plasma Loops through Differential Emission Measure Analysis, Advances in Space Research, Vol. 30 (3), 2002, pp. 507–516.

[SCB03] Schmelz, J.T., Cirtain, J.W., Beene, J.E., Blevins, H.T., Ellis, D., Medlin, D.A., Nasraoui, K., and Nevels, C., Coronal Loops: Isothermal OR Multithermal?, Advances in Space Research, Vol. 32 (6), 2003, pp. 1109–1115.

[PJH12] Peng-Xin, G., Jing-Lan, X., and Hong-Fei., L., Periodicity in the Most Violent Solar Eruptions: Recent Observations of Coronal Mass Ejections and Flares Revisited, Research in Astro. and Astrophysics, Vol. 12 (3), 2012, pp. 322–330.

[GBM90] Gosling, J.T., Bame, S.J., McComas, D.J., and Phillips, J.L., Coronal Mass Ejections and Large Geomagnetic Storms, Geophysical Research Letters, Vol. 17 (7), 1990, pp. 901–904.

[RFB15] Ronneberger, O., Fischer, P., and Brox, T., U-Net: Convolutional Networks for Biomedical Image Segmentation, Medical Image Computing and Computer-Assisted Intervention (MICCAI), LNCS, Vol. 9351, 2015, pp. 234–241.

[AND99] Aschwanden, M.J., Newmark, J., Delaboudiniere, J.-P., Neupert, W.M., Klimchuk, J.A., Gary, G.A., Portier-Fozzani, F., and Zucker, A., Three-dimensional Stereoscopic Analysis of Solar Active Region Loops. I. SOHO/EIT Observations at Temperatures of (1.0-1.5) x 106 K., The Astrophysical Journal, Vol. 515 (2), 1999, pp. 842–867.

[KNG04] Kandimalla, R.M., Newman, T.S., and Gallagher, D.L., Snake-based Technique for Plasmapause Tracking, in Proc., 17th Int'l Conf. on Pattern Recognition (ICPR 2004), Aug. 2004.

[LNG06a] Lee, J.K., Newman, T.S., and Gary, G.A., Oriented Connectivity-based Method for Segmenting Solar Loops, Pattern Recognition, Vol. 39 (2), 2006, pp. 246–259.

[LNG06b] Lee, J.K., Newman, T.S., and Gary, G.A., Dynamic Aperture-based Solar Loop Segmenta-

tion, in Proc., 2006 IEEE Southwest Symp. on Image Analy. & Interpret., Mar. 2006, pp. 91–94.

[Dur09] Durak, N., Nasraoui, O., and Schmelz J., Coronal Loop Detection from Solar Images, Pattern Recog., Vol. 42 (11), 2009, pp. 2481–2491.

[Dur10] Durak, N., Nasraoui, O., and Schmelz J., Automated Coronal-Loop Detection based on Cuntour Extraction and Contour Classification from the SOHO/EIT Images, Solar Physics, Vol. 264, 2010, pp. 383–402.

[Asc10] Aschwanden, M.J., A Code for Automated Tracing of Coronal Loops Approaching Visual Perception, Solar Physics, Vol. 262, 2010, pp. 399–423.

[LeT11] Lee, J.K., and Tang, W.K., Snake-based Technique for Automated Coronal Loop Segmentation, in Proc., 19th Int'l Conf. in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG 2011), 2011, pp. 33-40.

[ZXZ19] Zhiming, S., Xiaoli, Y., Zhongquan, Q., and Hong-Bo, L., Automatic Detection and Extraction Algorithm of Coronal Loops Based on Match Filter and Oriented Directivity, Monthly Notices of the Royal Astronomical Society, Vol. 490 (4), 2019, pp. 5567–5584.

[KSH12] Krizhevsky, A., Sutskever, I., and Hinton, G.E., Imagenet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems, Vol. 25 (2), 2012.

[KaZ15] Karen, S., and Zisserman, A., Very Deep Convolutional Networks for Large-scale Image Recognition, in Proc., 3rd Int'l Conf. on Learning Representations (ICLR 2015), San Diego, May 7–9, 2015, pp. 1–14.

[SLJ15] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanho, V., and Rabinovich, A., Going Deeper with Convolutions, in Proc., IEEE Conf. on Comp. Vision and Pattern Recog. (CVPR 2015), June 2015, pp. 1–9.

[GDD14] Girshick, R., Donahu, J., Darrell, T., and Malik, J., Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, in Proc., IEEE Conf. on Comp. Vision and Pattern Recog. (CVPR 2014), June 2014, pp. 580–587.

[EST14] Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D., Scalable Object Detection using Deep Neural Networks, in Proc., IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR 2014), June 2014, pp. 2155–2162.

[ZJR15] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P.H.S., Conditional Random Fields as Recurrent Neural Networks, in Proc., IEEE Int'l Conf. on Comp. Vision (ICCV 2015), 2015, pp. 1529–1537.

[BHC15] Badrinarayanan, V., Handa, A., and Cipolla, R., SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-wise Labelling, Comp. Research Repo., 2015.

[TL19] Tan, M. and Le, Q., Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks, in Proc., Int'l Conf. on Machine Learning (ICML 2019), Long Beach, June 9–15, 2019, pp. 6105–6114.

[HGD17] He, K., Gkioxari, G., DollÃ¡r, P., and Girshick, R., Mask R-CNN, in Proc., IEEE Int'l Conf. on Computer Vision (ICCV 2017), Venice, Oct. 22–29, 2017, pp. 2961–2969.

[LSV16] Lin, G., Shen, C., Van Den Hengel, A., and Reid, I., Efficient Piecewise Training of Deep Structured Models for Semantic Segmentation, in Proc., IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2016), Las Vegas, June 26 – July 1, 2016, pp. 3194–3203.

[LSD15] Long, J., Shelhamer, E., and Darrell, T., Fully Convolutional Networks for Semantic Segmentation, in Proc., IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2015), Boston, June 7–12, 2015, pp. 3431–3440.

[PCK16] Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E., Enet: A Deep Neural Network Architecture for Real-time Semantic Segmentation, arXiv preprint arXiv:1606.02147.

[KK16] Kalinovsky, A. and Kovalev, V., Lung Image Segmentation using Deep Learning Methods and Convolutional Neural Networks, in Proc., Int'l Conf. on Pattern Recognition and Info. Processing, 2016, pp. 21–24.

[RU18] Retina Blood Vessel Segmentation with a Convolutional Neural Network (U-Net), available on https://github.com/orobix/retina-unet (licensed under the MIT License)

[DYL17] Dong, H., Yang, G., Liu, F., Mo, Y., and Guo, Y, Automatic Brain Tumor Detection and Segmentation using U-Net Based Fully Convolutional Networks. in Proc., The Annual Conf. on Medical Image Understanding and Analysis, Edinburgh, July 11–13, 2017, pp. 506–517.

[KiB14] Kingma, D.P., and Ba, J., Adam: A Method for Stochastic Optimization, arXiv preprint arXiv:1412.6980.

[SRS11] Singh, T.R., Roy, S., Singh, O.I., Sinam, T., and Singh, K.M., A New Local Adaptive Thresholding Technique in Binarization, International Journal of Computer Science Issues, Vol. 8 (6), 2011, pp. 271–277.

Particle-Based Fluid Surface Rendering with Neural Networks

Viktória Burkus
Budapest University of Technology and Economics
Budapest, Hungary
burkus@iit.bme.hu

Attila Kárpáti
Budapest University of Technology and Economics
Budapest, Hungary
karpati@iit.bme.hu

László Szécsi
Budapest University of Technology and Economics
Budapest, Hungary
szecsi@iit.bme.hu

ABSTRACT

Surface reconstruction for particle-based fluid simulation is a computational challenge on par with the simulation itself. In real-time applications, splatting-style rendering approaches based on forward rendering of particle impostors are prevalent, but they suffer from noticeable artifacts.
In this paper, we present a technique that combines forward rendering simulated features with deep-learning image manipulation to improve the rendering quality of splatting-style approaches to be perceptually similar to ray tracing solutions, circumventing the cost, complexity, and limitations of exact fluid surface rendering by replacing it with the flat cost of a neural network pass. Our solution is based on the idea of training generative deep neural networks with image pairs consisting of cheap particle impostor renders and ground truth high quality ray-traced images.

Keywords

Computer Graphics, Metaballs, Generative Neural Network, Surface Reconstruction

1 INTRODUCTION

Realistic and resource-efficient visualization of fluid surfaces reconstructed with particle-based simulations is an ongoing challenge in modern computer graphics. The thousands of particles required to create plausible fluid behavior significantly complicate the execution of algorithms such as recursive ray tracing, which, due to their complexity, can often only be used to display simulation results in real-time using accelerating methods.

Representing particles with metaballs is a widely used and popular method for displaying the results of particle-based simulations. The metaball construct, introduced by Blinn [Bli82] and Nishimura [Nis85], describes an implicit surface formed by interacting objects. Each metaball has a radial density function (or radial basis function, RBF) and the metaballs together represent the isosurface of the density field. Due to the temporal change in the simulation, it is necessary to evaluate the isosurface in each frame, which significantly limits its usability in real-time applications. Since their inception, several publications have been produced to speed up the visualization of metaballs.

Grid-based solutions usually give faster results when the grid resolution is low, but artifacts form on the liquid surface. The disadvantage of solutions based on *on-surface tracer particles* is the complexity of maintaining a uniform coverage and the difficulty of avoiding gaps that form on the fluid surface. The problem to be solved in *ray tracing-based methods* is to speed up the evaluation of the density function required for the computation of the intersection between a ray and the surface. All of these methods are useful when a larger number of metaballs needs to be visualized, but realistic shading or global illumination requires additional resources in a different order of magnitude.

Neural networks trained with input exemplar–expected output image pairs have achieved significant success in the field of image manipulation. This paper contributes to the field of neural rendering by proposing a hybrid rendering solution for particle-based fluid simulation, combining stochastic rendering and image-to-image neural networks. This can be used in any context where fast fluid surface rendering is needed, but certain lighting effects require the environment to be fixed at training time.

Metaballs with complex shading are achieved by introducing image enhancement using a neural network, trained with the method presented in our paper. The question we investigate is how to generate the input images and the expected target images needed to train the

network to obtain a satisfactory result. We also analyze emerging artifacts to motivate further research.

The paper is organized as follows. Section 2 presents previous work on metaball rendering (not using neural networks), and the state of the art on neural rendering in general. Section 3 describes the proposed method. The results are summarized in Section 4, and finally, the conclusion and further research directions are described in Section 6 and Section 5.

2 PREVIOUS WORK

The first particle-based implicit surface was presented by J.F. Blinn [Bli82], which involved displaying electron density maps of molecular structures.

Blinn used a linear combination of Gaussian *radial basis functions* (RBFs) and a constant term for constructing the implicit equation of the surface. As Gaussian RBFs do not have finite support, all objects must be considered in order to determine the function value at any point, which is a necessary operation in any method extracting an isosurface. Therefore, its application involves considerable computational effort. Replacing the Gaussian density function with a finite-support function can reduce the computational effort considerably, since to evaluate the total density function at a point it is sufficient to consider the contribution of the metaballs within the finite environment of this spatial point. For our experiments in this paper, we used the sixth-degree polynomial RBF proposed by Wyvill [WMW86].

2.1 Metaball rendering techniques

The previous work on the representation of the surface of metaballs can be divided into several groups. However, the aim of all the publications mentioned below is to accelerate the evaluation of the density function. In this section, we survey these methods to highlight the fact that existing methods are either expensive or specialized with respect to the light transport phenomena they support.

2.1.1 Grid-based solutions

A widely used method is the *marching cubes* algorithm presented by Lorensen et al. [LC87]. The algorithm generates a triangular mesh model approximating an isosurface of volumetric density function. Such a model can be ray-traced efficiently using conventional methods. However, the quality of the generated surface depends on the resolution of the grid used. If a low-resolution grid is used, the algorithm is able to generate the surface quickly, but the resolution of the surface is low. When a high-resolution grid is used, the algorithm generates a detailed surface, but at increased computational and memory costs. In particular, the density function has to be evaluated at a high number of grid nodes.

Krone et al. [KSES12] present a GPU-accelerated version of the marching cubes approach. Even though they work with Gaussian density kernels, they use a cutoff radius to decrease complexity. Sorting into a 3D auxiliary grid is used for proximity searches.

The approach of Iwasaki et al. [IDYN06] makes use of a grid, but it is constructed using a virtual camera, and rendering metaballs, clustered into layers, into the grid. The method is strongly oriented towards configurations where most of the fluid forms a roughly horizontal surface in a container, with very few layers of splashes above. The isofurface is rendered using a splatting-like method named *surfels*. Real-time performance is achieved.

All grid based approaches are ill-suited for scenarios where the fluid cannot be assumed to reside within a finite container.

2.1.2 Screen-space visualization

Wladimir J. van der Laan et al. [vdLGS09] presented a screen-space metaball visualization method, which relies on rendering the metaballs as solid spheres, and applying image-space filtering to smooth the surface. The method provides real-time performance with a configurable speed-quality preference, but it is only efficient if the spheres provide a close approximation of the surface indeed, i.e. if only a few particles affect a surface point. Larger reconstruction kernels encompassing higher number of particles cannot be reproduced, and thus the fluid retains a somewhat viscous appearance even with the best filtering, especially when the balls appear large in image space.

Müller et al. [MGE07] also used metaballs for the visualization of a molecular dynamics simulation. They propose two ideas. The *vicinity texture* is a precomputed lookup structure for metaball neighbors, which is suitable for static geometries, but requires high texture bandwidth still. The *walking depth plane* approach finds isosurface depth in pixels using iterative correction of an initial guess. This, however, requires the metaballs to be rendered as billboards up to 100 times, making the approach scale poorly both with respect to the number of particles and to the target resolution.

Xu et al. [XW16] makes use of hardware multisampling anti-aliasing and alpha-to-coverage to perform four-layered depth peeling in a single pass. Using multiple such passes, a polynomial approximation of the density function using the first few metaballs can be obtained and solved explicitly. The approach works fast and provides accurate results when only a few metaballs contribute to individual surface points. Despite the ingenuity and performance delivered, the acceleration still only works for primary rays.

Van Kooten et al. [KBT07] reconstruct the surface using on-surface tracer particles uniformly distributed on the fluid surface. Repulsive forces, velocity constraints, and particle densities are used to determine the position of the particles, which are travelling on the surface following its motion. The expensive part of the method is computing the forces, especially maintaining the data structure to find near tracer particles. Where maintaining uniform coverage fails, small gaps may appear on the surface.

Kanamori et al. [KSN08] present a method based on ray tracing. Using depth peeling to obtain boundaries of effective spheres, polynomials for the density along ray segments are obtained and solved using the Bezier clipping root finding method. The method can produce high quality images, but the root finding is somewhat expensive.

Szécsi et al. [SI12] present a similar scheme based on building per-pixel fragment lists. They render particles as billboards, storing an entry, midpoint, and exit fragment in a per-pixel list. By sorting these fragments, and finding a cubic density approximation on ray segments between them, they are able to perform ray-surface intersection in a less expensive fashion. The drawback is the high overhead of gathering the per-pixel list.

2.1.3 Ray-casting and ray-marching methods

All screen-space methods fail to obtain correct reflections and refractions efficiently, as they organize their data structure around processing primary rays. An environment mapping approximation, neglecting self-reflections, multiple refractions, and media participation, is acceptable in some, but not all configurations.

Accurate visualization requires recursive ray tracing, where the evaluation of the density function at arbitrary points must be very efficient. Recently, Winchenbach and Kolb [WK20] proposed a GPU-friendly data structure that offers exceptional performance. However, the approach is still far from real-time.

2.2 Deep learning

The problem we investigate in this paper is a neural rendering problem, in the sense that an image should be produced from an RBF volume representation. As we use conventional rendering combined with a neural network, ours is not a pure neural rendering approach.

Pure text-to-image efforts are reviewed Frolov et al. [FHR+21], and today they are capable of synthesizing images sampled from a space spanned by training photo databases, but not from abstract geometrical scene descriptions.

Tewari et al. [TFT+20] summarize and classify the state-of-the-art neural rendering approaches that combine classical computer graphics pipelines and neural networks. Notably absent for the results are solutions that would render deterministic specular effects like reflections or refractions.

Thomas and Forbes [TF17] managed to achieve fast global illumination with deep neural networks, featuring diffuse interreflections with detailed occlusion, but without reflective or refractive materials.

Feygina et al. [FIM18] introduced CycleWGAN networks for enhancing global illumination rendering in post-processing for computer games, showing that the deep learning approach can be viable in real time.

Constantin and Bigand [CCB20] handle scenes with reflections and refractions, but they only use the neural network to detect noise and thus direct sampling in a path tracing algorithm.

Bui et al. [BLMD18] present a convolutional neural network-based approach for rendering high-resolution images from a point cloud.

Horvath [Hor19] uses a conditional GAN to generate metaball images from a simplified set of input training data. For the input, four-channel (rgb, depth) images of circles are rendered, where every circle represents a metaball. For output, it generates images of metaballs consisting of a color buffer, a depth buffer, and a normal buffer. No shadows, reflections, or refractions are produced. Our approach in this paper is going to be similar, but our objective is rendering images with perceptively acceptable reflections and reflections.

As we assume a volume representation based on RBFs, it is of interest to mention RBFNNs, or Radial Basis Function Neural Networks [BL88], and the more recent deep-RBF neural networks [ZHS18]. These contain a layer that outputs positions and weights for RBFs that reconstruct the function whose samples were used as training data. It has been used successfully in the function approximation and classification domains, but they have not been applied to rendering so far. In our work, the RBF representation of the fluid is considered an input, not an output—it is created by conventional physical simulation, not by a neural network. The output, in our case, is a rendered image. Therefore, our solution may be used in conjunction with deep-RBF networks producing 3D density representations.

Overall, high quality metaball rendering methods capable of rendering reflections, refractions, and media participation, are not real-time for a reasonable number of particles. Fast methods rely on some kind of screen space accumulation, and offer poor substitutes for reflection and refraction rendering. Note that our work is based on none of the above techniques. Instead, we aim to stylize an inexpensive render of particles into one featuring reflections and refractions using trained image-to-image transfer with a deep neural network. In part, this is similar to the van der Laan

approach [vdLGS09] of image space filtering, but using learned kernels instead of constructed ones. As above methods using depth listing or depth peeling, we also strive to gather information about multiple layers of metaballs, but we exploit the idea of stochastic rendering. In this, our solution is also different from Horvath's work [Hor19], and we also extend the approach to rendering reflections and refractions.

Our solution is based on Pix2Pix [IZZE17] and CycleGAN [ZPIE17]. Pix2Pix uses conditional adversarial networks for image-to-image translation. Training is done with input and expected output picture pairs. CycleGAN does not require the images to be paired. The resulting generator network can be used for images that are similar to the input training data, producing an output that is similar to the set specified as expected.

3 OUR PROPOSAL IN DETAIL

Our goal is to accelerate particle-based fluid visualization in applications where perceptual correctness is important. First, we render metaballs using an inexpensive stochastic forward rendering solution. Then, we use neural network processing to either reconstruct a surface in image space, or render a scene with the reconstructed fluid surface featuring reflections and refractions. The neural networks are image-to-image GAN realizations (those of the Pix2Pix and CycleGAN network in particular) trained on image pairs produced by our stochastic renderer and our reference ray tracer.

3.1 Reference ray-casting and ray-tracing

In order to train the neural networks, we need expected output images for given sets of metaball particles. We created two types of such reference images: false-colored surface reconstructions encoding the surface normals (Figure 1, left), and recursively ray-traced fluid surface rendering with a surrounding environment (Figure 1, right). In both cases, we used ray-marching to find surface points intersected by primary or secondary rays. For primary rays, we used a screen-space A-buffer for accelerating the evaluation of the density field given by the metaball particles, but we used brute force for secondary rays. The rendering time of the reference images is still negligible compared to the time requirements of training the neural networks. In the ray-traced reference images, radiance along rays not hitting any surface was taken from an environment texture. Thus, the environment itself was encoded in the neural network, meaning that for different environments, different networks have to be trained — even if an existing network of another environment provides a good starting point for the training. This way, every pixel in the reference image contains the contribution of all possible combinations of reflections and refractions along the light paths.

Figure 1: Reconstructed surface target reference and ray traced target reference.

3.2 Stochastic rendering

We need a method for the fast rendering of the fluid particles that does not lose information about the particles occluded by other metaballs. This is particularly important for those particles that contribute to the primary surface, but surfaces further back may also be important, especially if refractions are also to be reconstructed. Horvath [Hor19] renders particles as circles, which accomplishes these goals, but offers a starting point for the neural network that is far from the expected output. We propose to use stochastic rendering, where we randomly discard fragments of billboards representing metaballs, false-colored using the camera-space sphere normals, and linear depth values. As shown in Figure 2, the stochastic renders are perceptually already quite close to the ray-cast reference, but with noise that has to be filtered. Part of the purpose of the neural network is to perform this filtering in a learned way to produce a smooth surface.

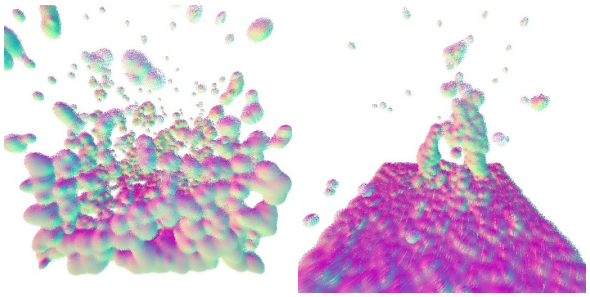


Figure 2: Images created using stochastic rendering. Images like these are inputs to both the training process, and to the image enhancement network used in real-time rendering.

3.3 Training datasets

To train the neural network, we render training sets consisting of image pairs. The *input* image is created using stochastic rendering, and the *target* image is rendered using either ray-casting or recursive ray tracing. Input images encode normals in camera space in their red and green channels, while camera-space linear depth is stored in the blue channel. To achieve stochastic rendering, we need to discard random fragments. For this

purpose, we used Cerisano's version [Cer15] of the *gold random* GPU-friendly fast random number generator. While this very simple code does not have high-quality PRNG properties, it is visually convincing, and serves our purpose of revealing occluded billboards.

In order to ensure realistic metaball configurations, we implemented SPH fluid simulation. In addition to physical forces, the simulation is controlled by animated triangle mesh models. In each simulation time step, particles are rendered with both the stochastic and the reference method to output images. The distance of the camera from the simulation space and the position of the camera changes randomly in each frame. Since camera depth is encoded in the blue channel, the farther the camera is from the metaball, the less blue there is in the color. Figure 3 shows an example image pair of the stochastically render input and the ray traced target output.

Figure 3: Image pair from the training set.

3.4 Training

We used a Tensorflow implementation of Pix2Pix [IZZE17] and a PyTorch implementation of CycleGAN [ZPIE17] to train the generator network with input and target image pairs. We do not give results separately for the two networks, as results were visually very similar, with CycleGAN performing faster. On the network obtained as the result, we ran the consecutive sequence of images saved from the application. The animation includes rotation and zooming around the simulation space.

3.5 Real-time rendering

The trained neural network can be used in a real-time application to render fluid surfaces. The input of the network is the stochastically rendered particle billboard cloud, and the output is presented by texturing it on a screen-filling quad.

If we use the surface reconstruction network, shading has to be performed in every pixel of the reconstructed surface. First, we decode the normal vector from the texture, then transforming it back to the world. Figure 4 shows the network output image and its shaded version using local illumination and environment mapping for approximated reflections and refractions.

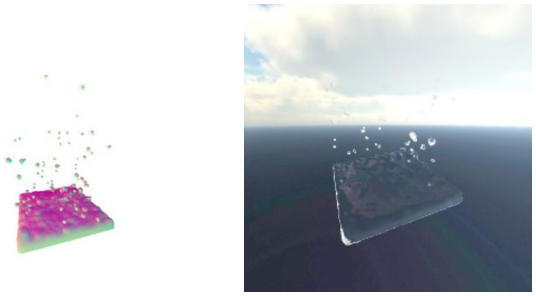


Figure 4: Surface reconstructed by the network and its shading with local illumination and environment mapping.

If we use the network trained with recursively ray-traced images, then further shading is not necessary.

4 RESULTS

We used a Tensorflow port of the Pix2Pix algorithm to train a deep neural network. We performed three trainings. First, we trained the network with 2078 image pairs, both of the pair in 512×512 resolution, showing incomplete and complete metaballs. The second one trained with input set contains metaballs shaded with surface reconstruction network and metaballs with ray-casting. The third trained with incomplete metaballs and metaballs with ray-casting. They performed around 30-50 epochs encompassing 72000-120000 training steps, taking approximately $8-15$ hours using an Nvidia Geforce RTX 2080 Super GPU. The same results could be achieved using the PyTorch implementation of CycleGAN in just 5 hours.

For testing we generated two test sets. One of them contains images of our fluid in consecutive simulation steps with changing camera setting. The other one contains zoom-in zoom-out pictures. Figure 5 and Figure 6 show the result of the first neural network. Therefore the environment not encoded in the neural network, but additional render pass is needed to shade the surface. Figure 7 shows the result of the second neural network. The environment is encoded and the network focus more on that part rather then the actual surface. The Figure 7 shows the result of the third neural network.

The Pix2Pix implementation proved to be inferior to the CycleGAN implementation both in terms of training time and network evaluation time. CycleGAN can be evaluated in $40ms$, which would be fast enough for actual real-time performance for full resolution targets.

5 LIMITATIONS AND FUTURE WORK

We plan to apply Wasserstein loss for the CycleGAN network, which has been proven to be useful in similar neural rendering applications [FIM18].

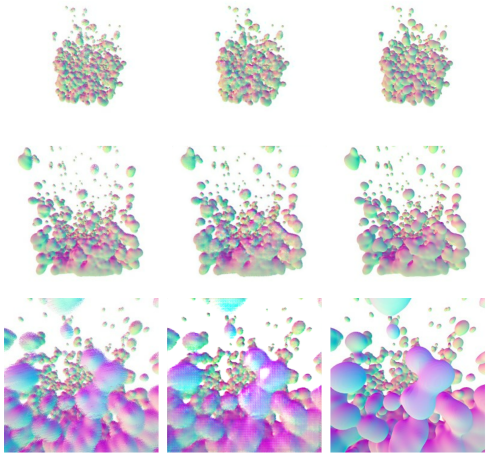


Figure 5: Stochastically rendered inputs (left), reconstructed surface images (middle), and target images (right).

Training the ray-tracing network with a fixed environment means that the method cannot be used for dynamic settings, including moving solid bodies mixed in the fluid. Such a static environment is acceptable in applications like computer games, where the game levels and rooms are designed in advance, and pre-training a fluid-lighting network for every relevant location would be reasonable. However, a solution using an environment map or scene capture on-the-fly would certainly extend the applicability of our approach. The most promising avenue would be training a network to render fluids and their lighting interactions with solid geometry and the environment in a single run.

Currently, the ray-tracing network does not only learn to shade the fluid surface, but also to create the background. This may diminish its ability to correctly render reflections and refractions. Modifying the training process to exclude backgrounds, and rendering the background in a forward manner would be desirable.

6 CONCLUSIONS

We have described a solution for enhancing fast fluid rendering with neural networks train in an adversarial manner. Our results indicate that the approach is viable, but further research is needed to extend the solution to more dynamic use cases and further improve performance.

7 ACKNOWLEDGEMENTS

This work has been supported by OTKA K-124124. The research presented in this paper, carried out by BME, was supported by the *Ministry of Innovation*, and the *National Research, Development and Innovation Office*, within the framework of the *Artificial Intelligence National Laboratory Programme*.

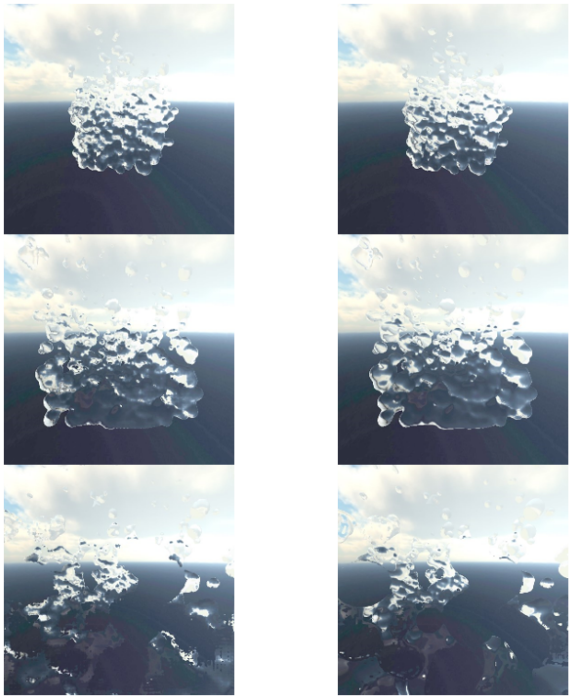


Figure 6: Final results of surface reconstruction with environment mapping (left) compared to the environment mapped reference surface (right)

8 REFERENCES

[BL88] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[Bli82] James F Blinn. A generalization of algebraic surface drawing. *ACM transactions on graphics (TOG)*, 1(3):235–256, 1982.

[BLMD18] Giang Bui, Truc Le, Brittany Morago, and Ye Duan. Point-based rendering enhancement via deep learning. *The Visual Computer*, 34(6):829–841, 2018.

[CCB20] Ibtissam Constantin, Joseph Constantin, and André Bigand. On the use of deep active semi-supervised learning for fast rendering in global illumination. *Journal of Imaging*, 6(9):91, 2020.

[Cer15] Dominic Cerisano. Gold noise uniform random static. `http://https://www.shadertoy.com/view/ltB3zD`, 2015. Online; accessed 2021-03-15.

[FHR+21] Stanislav Frolov, Tobias Hinz, Federico Raue, Jörn Hees, and Andreas Dengel. Adversarial text-to-image synthesis: A re-

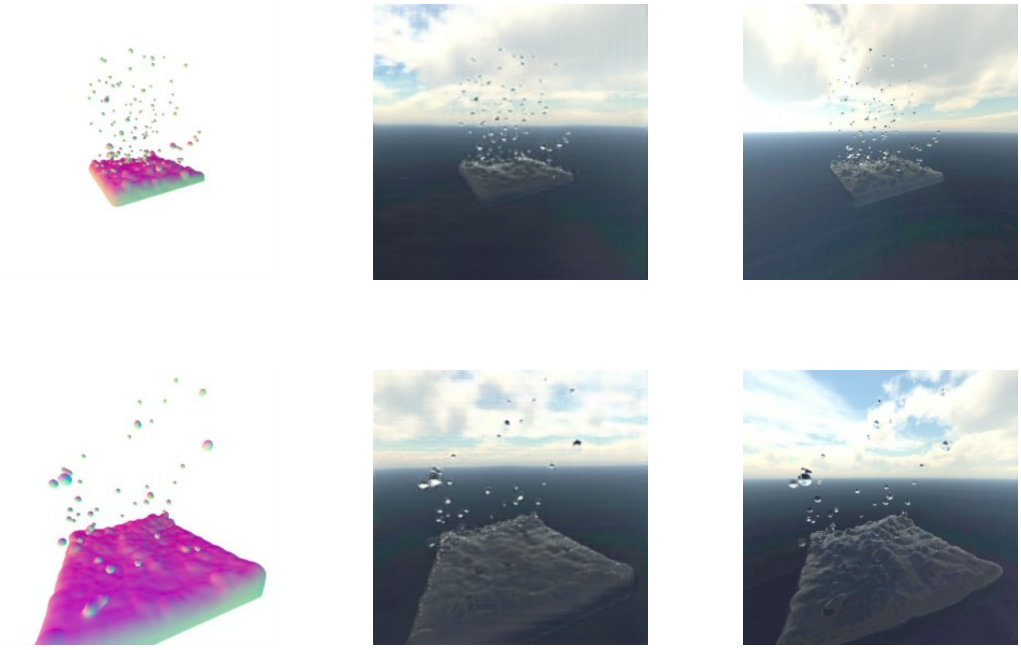


Figure 7: Surface reconstructed by the network (left), output images of the ray-tracing network (middle), and ray-traced reference (right).

view. *arXiv preprint arXiv:2101.09983*, 2021.

[FIM18] Anastasia Feygina, Dmitry I Ignatov, and Ilya Makarov. Realistic post-processing of rendered 3d scenes. In *ACM SIGGRAPH 2018 Posters*, pages 1–2. ACM New York, NY, USA, 2018.

[Hor19] Robert Horvath. *Image-space metaballs using deep learning*. PhD thesis, Wien, 2019.

[IDYN06] Kei Iwasaki, Yoshinori Dobashi, Fujichi Yoshimoto, and Tomoyuki Nishita. Real-time rendering of point based water surfaces. In *Computer Graphics International Conference*, pages 102–114. Springer, 2006.

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[KBT07] K KOOTEN, G BERGEN, and A TELEA. Point-based visualization of metaballs on a GPU. *GPU Gems*, 3, 2007.

[KSES12] Michael Krone, John E Stone, Thomas Ertl, and Klaus Schulten. Fast visualization of gaussian density surfaces for molecular dynamics and particle system trajectories. In *EuroVis (Short Papers)*, pages 067–071, 2012.

[KSN08] Yoshihiro Kanamori, Zoltan Szego, and Tomoyuki Nishita. GPU-based fast ray casting for a large number of metaballs. In *Computer Graphics Forum*, volume 27, pages 351–360. Wiley Online Library, 2008.

[LC87] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[MGE07] Christoph Müller, Sebastian Grottel, and Thomas Ertl. Image-space GPU metaballs for time-dependent particle data sets. In *VMV*, pages 31–40, 2007.

[Nis85] Hitoshi Nishimura. Object modeling by distribution function and a method of image generation. *Trans Inst Electron Commun Eng Japan*, 68:718, 1985.

[SI12] László Szécsi and Dávid Illés. Real-time metaball ray casting with fragment lists. In *Eurographics (Short Papers)*, pages 93–96, 2012.

[TF17] Manu Mathew Thomas and Angus G Forbes. Deep illumination: Approximat-

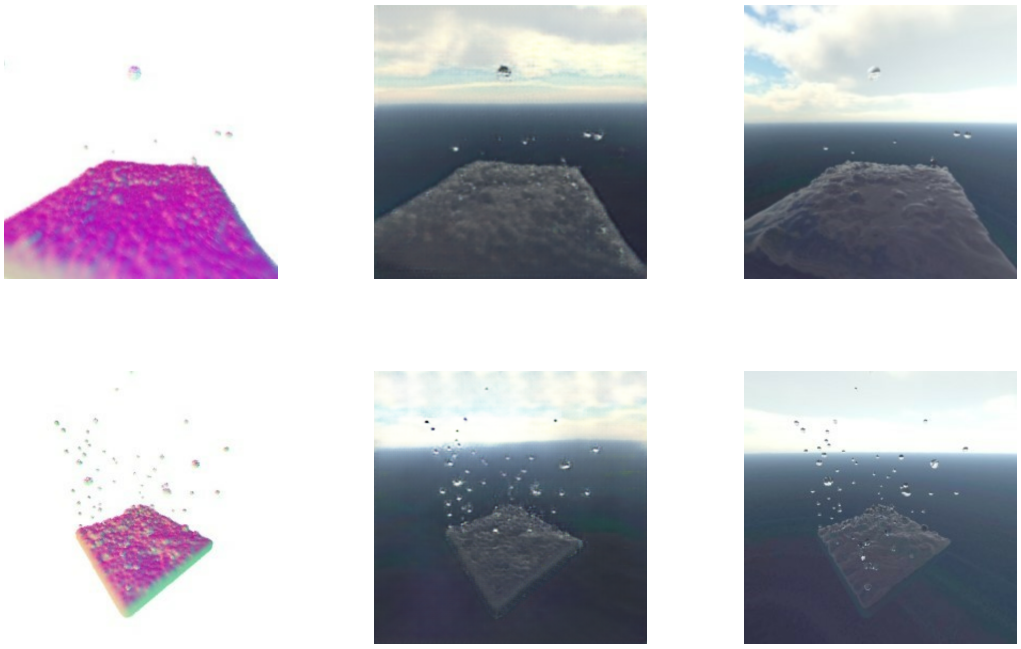


Figure 8: Stochastically rendered inputs (left), output images of the ray-tracing network (middle), and ray-traced reference (right).

ing dynamic global illumination with generative adversarial network. *arXiv preprint arXiv:1710.09834*, 2017.

[TFT+20] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, volume 39, pages 701–727. Wiley Online Library, 2020.

[vdLGS09] Wladimir J van der Laan, Simon Green, and Miguel Sainz. Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 91–98, 2009.

[WK20] R Winchenbach and A Kolb. Multi-level memory structures for simulating and rendering smoothed particle hydrodynamics. In *Computer Graphics Forum*, volume 39, pages 527–541. Wiley Online Library, 2020.

[WMW86] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The visual computer*, 2(4):227–234, 1986.

[XW16] Tian-Chen Xu and En-Hua Wu. View-space meta-ball approximation by depth-independent accumulative fields. In *SIGGRAPH ASIA 2016 Technical Briefs*, pages 1–4. ACM New York, NY, USA, 2016.

[ZHS18] Pourya Habib Zadeh, Reshad Hosseini, and Suvrit Sra. Deep-rbf networks revisited: Robust classification with rejection. *arXiv preprint arXiv:1812.03190*, 2018.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Signal Extraction for Classification of Noisy Images Compressed using Autoencoders

Dorsaf Sebai
Cristal Laboratory
ENSI
Manouba, Tunisia
dorsaf.sebai@ensi-uma.tn

Nour Missaoui
INSAT
Centre Urbain Nord
Tunis, Tunisia

Asma Zouaghi
INSAT
Centre Urbain Nord
Tunis, Tunisia

ABSTRACT

The world is experiencing an increasing boom in computer vision. This is more and more used in many domains such as robotics, medicine, industry, security systems, etc. In this context, Deep Neural Networks (DNNs) have great capabilities and are widely used. Convolutional Neural Networks (CNNs) present a particular class of DNNs that is most commonly leveraged to analyzing visual imagery. However, CNN performances completely depend on two main issues. The first issue is related to the quality of the images generated by capture cameras. All images captured by remote sensors and modern imaging systems are practically noisy, which can prevent the image from being correctly classified and identified by a CNN. The second issue is the throughput available for the transmission of the large amount of data between capture sensors and units processing CNNs. A seamless transmission can be ensured by compression techniques that help reducing the size of data, while affording the required quality for computer vision algorithms. Since lossy compression of noise-free and noisy images differ from each other, this work firstly raises the question of CNNs resilience to noisy images compression using the particular autoencoders. We secondly propose a method that aims to improve this resilience so that CNNs can achieve better classification performances. The compressed noisy images are passed, as a test set, along a model that is learnt from a noise dataset. The subtraction of the so captured noise from the noisy images is then performed to extract the useful signal to classify. This will be first work, where we learn the autoencoder from the noise sample, and not the noisy sample, while denoising. Obtained results prove the efficiency of the proposed method.

Keywords

Autoencoder, noise, classification, noise dataset.

1 INTRODUCTION

Computer vision enhances the understanding of the information depicted by images. However, the capture of the environment through sensors leads to various kinds of voluminous images and videos. Being an unsupervised feature extraction technique, AutoEncoders (AE) can be particularly useful for computer vision algorithms, as they can address the images compression to reduce the transmission delay between sensors and processing units. Further, their feature extraction asset allows not to lose details that cannot be seen by human eye, but are picked up by CNNs. However, autoencoders are generally trained assuming that the input images are artifact-free. This is not the case in real computer vision application scenarios, where digital images are subject to a wide range of noises. Here, we have to note that autoencoders can be effectively used for noise removal, if the desired output from decoder is fixed to be similar to original images. But, this cannot be exploited if the noise is introduced during image acquisition. In this case, we do not have pristine, i.e., artifact-free, images to use them as desired output of the autoencoder. The output is instead fixed to be similar to noisy images. In this context, the contributions of this paper are as follows :

- Study of the impact of the autoencoder-based noisy images compression on the performance of image classification using CNNs. We investigate in training both vanilla and deep autoencoders. In-depth analyses of involving different noise types of low, medium and high levels in the training process are also provided.
- The inclusion of a signal processing autoencoder in CNNs to get rid of the noise and preserve only the high valued signal elements, which will consequently improve the accuracy of noisy images classification and make the classifier more suitable. The

proposed method operates with no ground truth images available and improves the generalization ability of existing deep denoisers to several real noise models and image types.

The rest of this paper is organized in the following way. Section 2 is dedicated to related work enumeration. In Section 3, we detail the learning architecture used for the image compression and classification. Discussions of test results on pristine and noisy compressed images are also presented. In Section 4, we propose a method that aims to enhance the image classification performance while extracting clearer objects to be detected by CNNs. Evaluation of the proposed method performances along with comparative results are presented in Section 5. Finally, Section 6 concludes the paper.

2 CONTRIBUTIONS

Since classification presents the most common step of almost all computer vision applications, our first aim is to study the impact of noisy image compression using autoencoder on CNN classification performances. This is particularly important as a handful of works, e.g., [1] [2] [3], only study the impact of standard compression techniques, e.g., JPEG and JPEG 2000. However, artifacts of autoencoder-based compression methods have different characteristics from distortions caused by classical codecs [4], especially when images are noisy. Further, existing studies consider noise and compression artifacts separately, whereas we aim to jointly assess the impact of noisy images compression on the CNN performance. This aligns well with the real-world computer vision applications where images are often both noisy and voluminous.

In order to remove the noise harmful impact on image classification, several works have been proposed ranging from common filters, e.g., Gaussian, Median and Bilateral, to more elaborated efforts. The denoising domain is ever evolving, and a recent addition in this regard is the use of deep neural networks, based on CNNs [6] [7], autoencoders [5] [8] or both [10]. Some of the referenced methods are non-blind [8] since they learn on a separate training set and use the trained model to denoise new test samples. This makes them fail when the test images to denoise are not of the same kind as the models learnt with. [5] [6] [7] are however blind methods that overcome the shortcoming of the non-blind ones, insofar as they learn the model adaptively from the signal at hand while denoising. The main disadvantage of both blind and non-blind methods is that, one never knows how good the learned autoencoder will generalize to unseen noises [5] [9]. Most of them are good enough for some noises, such as Gaussian and impulse, but generalize poorly to noisy images with more sophisticated noise. Further, some of the aforesaid methods make use of noisy-clean image pairs [7], whereas pristine images are not always available in real-world computer vision applications.

In this paper, we propose an autoencoder-based method that extracts the signal relevant to the classifier from the noisy compressed images. Actually, we are not really proposing a denoiser where the reconstructed image is recovered from its noisy version. We rather propose a method that looks for the information that is useful to improve the accuracy of the classification and injects it to the CNN. We first make use of an autoencoder to learn the noise features from a noise dataset which includes noise images, and not noisy ones. Second, the compressed noisy images are passed, as a test set, along the model that was obtained from the noise feature extraction step. As the model is trained to distinguish the noise features, it will obviously capture the noise that exists in the compressed noisy images. It is sufficient to subtract this captured noise from the noisy images to extract the useful signal to classify. The key strength of our work is five-fold :

- Unlike some existing methods where noise-free images are necessary for them to function, our method does not require pristine images and is noiseless native images independent. This is more realistic as images captured in real-world computer vision applications are often noisy and no ground truth images are available.
- The proposed work jointly considers the noise and compression artifacts. Bearing in mind real life scenarios, it is usual for an image data to be noisy and voluminous so that it requires noise processing and compression together. To the best of our knowledge, there are no methods that address these two issues together.
- To reduce noise, our method excludes the necessity to train the model for the noisy sample itself, i.e., blind methods, or for a noisy sample other than the one to denoise, i.e., non-blind methods. Our model is learnt from the noise dataset. Thus, the proposed method performance would not be impacted by the volume of the training noisy data, either they are large or not.
- Being independent from noisy images, our method does not fail to efficiently recover different types and modalities of unseen images.
- Most of the existing methods, not to say all of them, perform well only when the noise level and type, present in the training and test noisy images, are same or differ only a little. In our case, this is no longer required. It is enough to prepare a noise dataset that contains the desired noise levels and

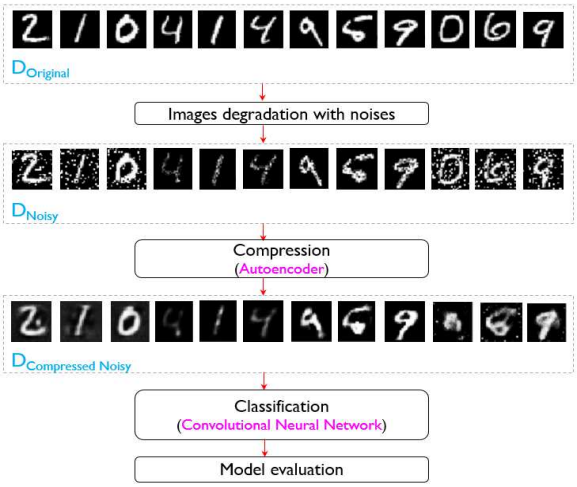


Figure 1: Steps of the learning architecture: MNIST as an example of benchmark dataset.

types. This will be first work, where we learn the autoencoder from the noise sample, and not from the noisy sample, while denoising.

3 CLASSIFICATION OF COMPRESSED NOISY IMAGES

3.1 Learning architecture

Fig. 1 depicts the steps of our learning architecture applied on different benchmark test datasets :

1. We construct the dataset D_{Noisy} that includes the distorted images by applying different common types of noises to original images of the $D_{Original}$ dataset.
2. We construct the $D_{NoisyCompressed}$ dataset that includes the compressed images of the D_{Noisy} dataset using different autoencoder configurations.
3. We pass the images of the $D_{NoisyCompressed}$ dataset along a state-of-the-art CNN in order to classify them.
4. According to results of step 3, we analyze how autoencoders can deal with the injected noises whose undesirable effects, such as artifacts, unrealistic edges, unseen lines, corners, blurred objects and disturbed background scenes, can impact the classification accuracy of CNNs. We remember that the autoencoder here is used for compression and not as a denoiser. Indeed, the noise is introduced during the capturing so that we do not hold pristine images.

The learning architecture shown in Fig. 1 is evaluated on three well-known $D_{Original}$ benchmark datasets, namely Mixed National Institute of Standards and Technology (MNIST), i.e., black and white images, Fashion MNIST, i.e., grayscale images, and CIFAR-10, i.e., color images. For each of the abovementioned datasets, we construct the D_{Noisy} training data by injecting different types and levels of noise into the pristine training $D_{Original}$ images. We are particularly interested in 5 types of commonly encountered noises, namely the Gaussian as an amplifier noise, Poisson as a shot noise, Uniform as a quantization noise, Speckle as a multiplicative noise and Salt & Pepper as an impulse noise. All these noise types are fairly presented in D_{Noisy} dataset, i.e. each of them presents 20% of the total samples. Furthermore, three levels, corresponding to standard deviation (σ) values of 10, 50, and 100, are considered for each noise. This training on a mixture of samples with multiple types of noises at different levels, rather than a certain type and level, will provide the deep neural networks models, autoencoder and CNN in our case, with stronger generalizing ability.

3.2 Used autoencoders and CNNs

The so obtained D_{Noisy} dataset is the entry of an autoencoder in order to generate the $D_{NoisyCompressed}$ dataset of Fig. 1. For MNIST and Fashion MNIST, 60.000 and 10.000 of the noisy images are used for the training and test, respectively. For CIFAR-10, 50.000 and 10.000 are the number of images used for training and test. Two different settings of the autoencoder are considered. The former presents a vanilla autoencoder with a fully-connected code layer of size 32. The latter presents a deeper autoencoder with 3 encoding fully-connected layers of respective sizes equal to 128, 64 and 32. We note that a fully-connected layers based autoencoder effectively operates for MNIST and Fashion MNIST, but not for CIFAR-10. Thus, we opt for a CNN-AE for this latter. Unlike simple handwritten digits and clothing items, there are many more features and details to extract from each CIFAR-10 image. To ensure the abovementioned settings, we make use of CNN-AEs with 1 and 3 encoding convolutional layers. These two settings are trained using Adam optimizer with a batch size 128 for 100 epochs. All our experiments use the initial learning rate of 0.1 which decays for every 20 epochs with an exponential rate of 0.1. The Mean Squared error (MSE) is used as loss function. Here, we remind that the reconstructed images are aimed to be as close as possible to noisy ones since as already mentioned no noise-free images are available.

The $D_{NoisyCompressed}$ dataset, output of the aforesaid autoencoder, is then provided as the input data to learn the CNN model. We use 70% of the $D_{NoisyCompressed}$ samples for each CNN training epoch. The remaining 30% are exploited for the test. To make the future comparisons fair, we look for the CNN model that fits each benchmark dataset. If the same CNN would be used for all datasets, this would produce unbiased results as the CNN architecture is optimized for one dataset and not

Table 1: Training strategies.

Strategies	Training images	Noise levels	AE depth
1	Pristine	-	1
2	Pristine	-	3
3	Noisy (Gaussian, Poisson, Uniform, Speckle and S&P)	10	1
4	Noisy (Gaussian, Poisson, Uniform, Speckle and S&P)	10	3
5	Noisy (Gaussian, Poisson, Uniform, Speckle and S&P)	50	1
6	Noisy (Gaussian, Poisson, Uniform, Speckle and S&P)	50	3
7	Noisy (Gaussian, Poisson, Uniform, Speckle and S&P)	100	1
8	Noisy (Gaussian, Poisson, Uniform, Speckle and S&P)	100	3

for the other. Thus, we opt for three CNN models that are optimized for each of our datasets MNIST, Fashion MNIST, and CIFAR-10 and allow respective accuracies of 99.8%, 93%, and 90%. These models are largely detailed in [11], [12] and [13].

3.3 Results and discussion

Eight different training strategies are implemented to evaluate the impact of the noisy compressed images on one of the main computer vision applications : classification. All these strategies share the same architecture of Fig. 1, but present as depicted in Table 1 different combinations of training images, noise levels and autoencoder depths. For each strategy, the classification performances will be then analyzed when the CNNs in [11], [12] and [13] are trained on pristine and compressed noisy images of MNIST, Fashion MNIST, and CIFAR-10 datasets, respectively. As indicated in section 3.2, the compression is carried out using the two autoencoder settings, namely depth 1 and 3. As a shorthand, we refer to these settings as AE_D1 and AE_D3. Classification performances are evaluated in Table 2 in terms of accuracy (%), and will serve as reference values in section 5 to assess our proposed noise recovery method. We note that each cell in Table 2 is composed of two lines that respectively correspond to the autoencoder settings : the value in the first line is related to AE_D1, whereas the value in the second line corresponds to AE_D3. The numbers after the down arrow correspond to the difference between accuracy values obtained for noisy compressed images and pristine compressed ones.

Results of Table 2 exhibit that training strategies with AE_D1, i.e., 1, 3, 5 and 7 of Table 1, allow better classification performances than AE_D3, i.e., 2, 4, 6 and 8, on both pristine and noisy images. Furthermore, the classification performances of the CNN model, trained on noisy compressed images, decrease when the noise level increases. Although expected, we could not measure how much this decay would be if we had not carried out these experiments. Compared to pristine images, the accuracy of CNN for compressed noisy MNIST images decreases, at σ value of 10, by 9.27% for AE_D1 and 7.90% for AE_D3. The decrease is about 11% and 10.96% for σ value of 50, and 14% and 14.41% for σ of 100. For Fashion MNIST, the accuracy of the CNN for compressed noisy images decreases, when compared to pristine images at level 10, by 9.60% and 8.68% for the respective AE_D1 and AE_D3. Even with the σ value 50 where the noise is still moderate, the accuracy drops almost 16.01% for AE_D1 and 17.29% for AE_D3. The same observation also stands for the CIFAR-10 dataset.

4 EFFICIENT CLASSIFICATION SCHEME FOR AE-BASED COMPRESSED NOISY IMAGES

Following the study of the noisy image autoencoder-based compression impact on classification, the proposed method to enhance obtained results is presented in this section. Compared to the flowchart of Fig. 1, we include a new step as a part of the enhanced learning architecture. As shown in Fig. 2a, this step consists in extracting the signal from the mixture of image and noise. It is actually decomposed to three sections, namely noise features learning, noise extraction then signal subtraction (*cf.* Fig. 2b) :

- **Noise features learning :** First, we construct a noise dataset that includes noise, and not noisy, images. Typically, they are images that contain noise without any other content, e.g., objects, contours, and texture. The constructed dataset is consisted of 45000 images of the same size than the training dataset images. All the five noises as well as their three levels, specified in section 3.1, are equally presented in the dataset. Second, we train a vanilla autoencoder, as defined in section 3.2, on our noise dataset to generate a model that captures the noise features.
- **Noise extraction :** The $D_{NoisyCompressed}$ dataset is no more passed, as it is, along the CNN for classification. A noise extraction step is added in order to subtract the noise from the images before their classification. We typically consider the $D_{NoisyCompressed}$ as a test set for the model that was obtained from the noise features learning step. Since the model is trained on a noise dataset, its parameters, namely weights and biases, are adjusted in a way that they represent noise features. Therefore, passing $D_{NoisyCompressed}$ dataset along the noise model as a test set would generate images that are similar to those of noise ones.
- **Signal subtraction :** The output images of the noise extraction step are subtracted from the $D_{NoisyCompressed}$ images in order to extract the signal that is useful for the classification, i.e., the absolute

	Pristine images	Compressed noisy images		
		$\sigma = 10$	$\sigma = 50$	$\sigma = 100$
MNIST	99.57	90.30 ↓ 9.27	88.57 ↓ 11.00	85.57 ↓ 14.00
	96.43	88.53 ↓ 7.90	85.47 ↓ 10.96	82.02 ↓ 14.41
Fashion MNIST	93.21	83.61 ↓ 9.60	77.20 ↓ 16.01	70.18 ↓ 23.03
	91.51	82.83 ↓ 8.68	74.22 ↓ 17.29	68.03 ↓ 23.48
CIFAR-10	90.04	79.10 ↓ 10.94	73.26 ↓ 16.78	69.16 ↓ 20.88
	86.69	75.28 ↓ 11.41	70.05 ↓ 16.64	67.72 ↓ 18.97

Table 2: Classification accuracy (%) of pristine and compressed noisy images for **MNIST**, **Fashion MNIST**, and **CIFAR-10** datasets. Each cell is composed of two lines that are respectively related to **AE_D1** and **AE_D3** settings.

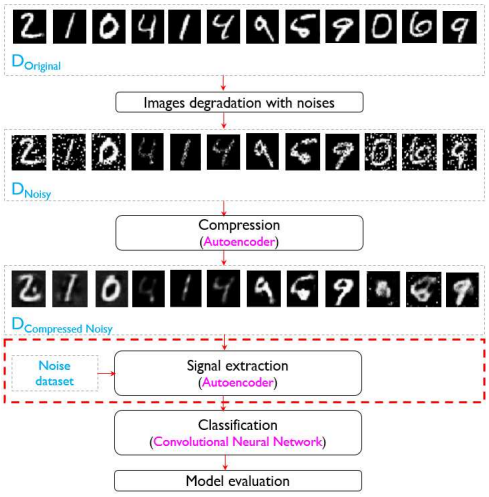
value of the subtraction. Here, we note that our aim is not the image denoising. We rather aim to take a neat signal from the image, and subtract the noise that can disrupt the CNN classification results. As we can observe in the last rectangle of Fig. 2b, the images backgrounds are also removed while carrying out the subtraction. This aligns with what we have just affirmed, the fact that the proposed method is not aimed for denoising. It is for object shape preservation to enhance classification results of CNNs. The image background removal will not affect the CNN performances since what is important here is the object to be classified.

5 EXPERIMENTATIONS

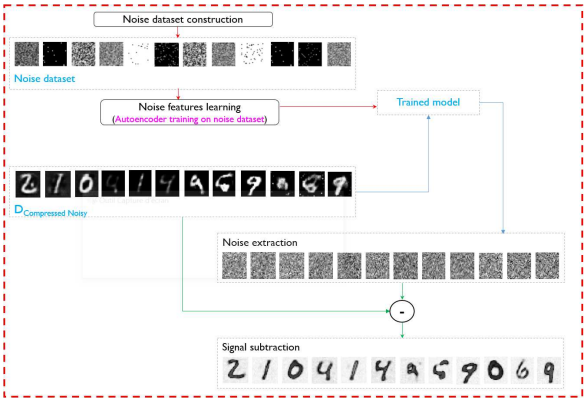
5.1 Comparison with candidate methods

Performances of the proposed method are shown through accuracy in Table 3 for classification of compressed noisy images of MNIST, Fashion MNIST, and CIFAR-10 datasets. Each cell of Table 3 is composed of seven lines. The first line corresponds to classification performances without noise processing (*cf.* Fig. 1) that were obtained from experiments of section 3.3. The six remaining lines are respectively related to classification performances with noise processing using the proposed method (*cf.* Fig. 2a), the 5×5 median, gaussian and bilateral filters, and Agostinelli et al. method [8]. For our method, the AE_D1 setting has been considered as autoencoder depth, as it considerably performs better than the AE_D3 one according to results of Table 2. Like our method, Agostinelli et al. [8] method is based on autoencoders. As a non-blind approach, we train the model in Agostinelli et al. [8] method on two separate training data that are respectively different from and more or less of similar content to test MNIST, Fashion MNIST and CIFAR-10 samples.

We typically use ImageNet [14] and EMNIST [15] whose images content respectively resembles the CIFAR-10 dataset and MNIST variants. However,



(a)



(b)

Figure 2: (a) Steps of the enhanced learning architecture : MNIST as an example of benchmark dataset, (b) Flowchart of the proposed signal extraction.

SARS-CoV-2 CT-scan dataset [16] is leveraged as totally different kind of images from our benchmark MNIST, Fashion MNIST, and CIFAR-10 test samples. Being one of the biggest COVID-19 datasets, SARS-CoV-2 CT-scan includes 2482 Computerized Tomography (CT) scans from 120 patients, with 1252 CT scans of 60 patients infected by COVID-19, and 1230 CT scan images of 60 non-infected patients by COVID-19, but presenting other pulmonary diseases. Improvements of the candidate methods over the classification without noise processing are indicated by the values after the up arrows. We remind that the CNNs in [11], [12] and [13] are used for the classification of MNIST, Fashion MNIST and CIFAR-10 for all the candidate methods.

Obtained results of Table 3 show that our method is more robust than its candidates. It succeeds in classifying images with more confident level than filters and Agostinelli et al. [8] method for all the evaluation noise rates and benchmark datasets. For MNIST dataset as example, the proposed method enables the CNN to reach an accuracy of 96.74%, 93.68% and 90.83% for respective levels 10, 50 and 100, meaning an enhancement of 6.44%, 5.11% and 5.26% in comparison with CNN used without any noise processing. These enhancements are higher than those achieved by the gaussian, median and bilateral filters, the most used noise filters. Trained on EMNIST, Agostinelli et al. [8] method (*Agostinelli et al. [8]_EMNIST*) achieves lower classification accuracies than ours. Accuracies are further lower when Agostinelli et al. [8] method is trained on CT (*Agostinelli et al. [8]_CT*) that is not similar to the MNIST test set. In fact, Agostinelli et al. method [8] performances depend on the images it was trained on. They are considerably decreased when the test images to denoise, i.e., MNIST, Fashion MNIST and CIFAR-10, are not of the same kind as the model learnt with, i.e., CT.

5.2 Results per noise type

Until now, our evaluation considered all types of noise together gathered in the test set. As already mentioned, the noise types are fairly presented in D_{Noisy} dataset, where each of them presents 20% of the total samples. In this present section, we assess the CNN performances of Fig. 2a while being trained on the mixture of the noise types, but tested on each type of noise separately. Hence, we can study how damaging a specific type of noise is when compressed by an autoencoder. The obtained results are presented in Fig. 3 in terms of accuracy for MNIST, Fashion MNIST, and CIFAR-10 at AE_D1. As can be observed, Gaussian and Uniform are noises that mostly affect the classification performances as they present the lowest accuracies for all the test datasets and noise levels. In contrary, the Speckle noise leads to the best results.

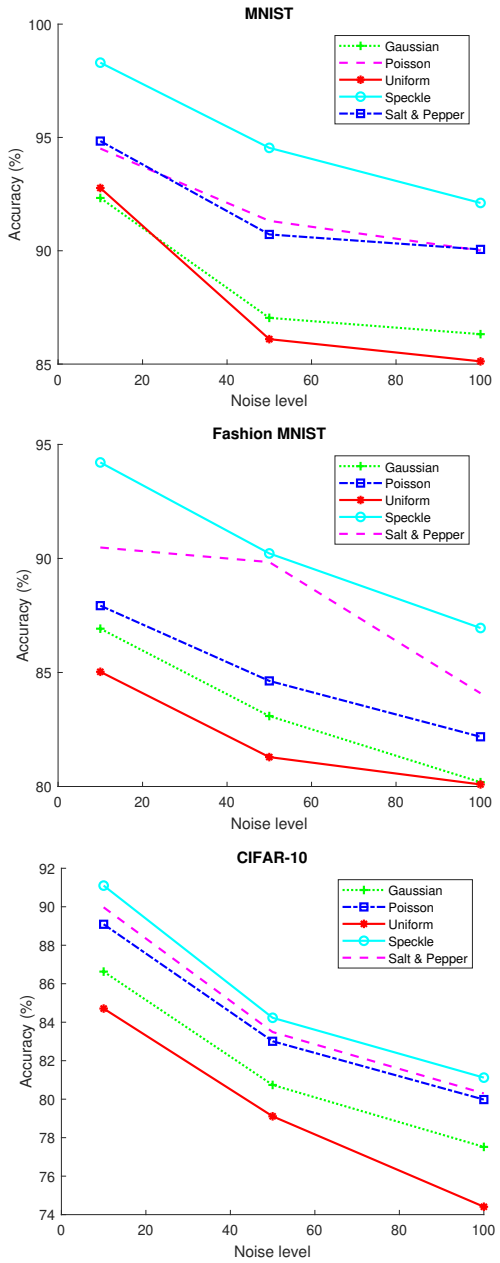


Figure 3: Classification accuracy (%) of compressed noisy images for MNIST *(top)*, (b) Fashion MNIST *(middle)*, and (c) CIFAR-10 *(bottom)* per separate noise type.

6 CONCLUSION

Image classification is a non trivial visual task, especially when it faces the presence of real life inevitable noise and compression artifacts. To address these issues, several researches have been conducted utilizing denoisers to restore original images, from noisy ones, before using CNNs for classification. We rather aim to extract, from noisy images, the pertinent signal useful for classification. The noisy compressed images are passed, as a test set, along a model that is trained on a noise dataset. The so generated images are then sub-

		Compressed noisy images		
		$\sigma = 10$	$\sigma = 50$	$\sigma = 100$
MNIST	w/o Noise processing	90.30	88.57	85.57
	Proposed method	96.74 ↑ 6.44	93.68 ↑ 5.11	90.83 ↑ 5.26
	Gaussian	94.57 ↑ 4.27	92.91 ↑ 4.34	90.25 ↑ 4.68
	Median	93.30 ↑ 3.00	91.28 ↑ 2.71	88.57 ↑ 3.00
	Bilateral	93.53 ↑ 3.23	90.72 ↑ 2.15	88.63 ↑ 3.06
	Agostinelli et al. [8]_EMNIST	94.10 ↑ 3.80	91.31 ↑ 2.74	90.02 ↑ 4.45
	Agostinelli et al. [8]_CT	91.11 ↑ 0.81	89.05 ↑ 0.48	86.41 ↑ 0.84
Fashion MNIST	w/o Noise processing	83.61	77.20	70.18
	Proposed method	90.33 ↑ 6.72	88.21 ↑ 11.01	85.37 ↑ 15.19
	Gaussian	89.18 ↑ 5.57	86.92 ↓ 9.72	84.63 ↑ 14.45
	Median	87.23 ↑ 3.62	85.78 ↓ 8.58	84.11 ↑ 13.93
	Bilateral	88.01 ↑ 4.40	85.45 ↓ 8.25	82.29 ↑ 12.11
	Agostinelli et al. [8]_EMNIST	89.30 ↑ 5.69	86.43 ↑ 9.23	84.14 ↑ 13.96
	Agostinelli et al. [8]_CT	84.44 ↑ 0.83	80.04 ↑ 2.84	77.27 ↑ 7.09
CIFAR-10	w/o Noise processing	79.10	73.26	69.16
	Proposed method	87.29 ↑ 8.19	82.89 ↑ 9.63	79.10 ↑ 9.94
	Gaussian	85.81 ↑ 6.71	81.62 ↑ 8.36	77.06 ↑ 7.90
	Median	84.79 ↑ 5.69	80.10 ↑ 6.84	76.83 ↑ 7.67
	Bilateral	83.55 ↑ 4.45	79.66 ↑ 6.40	75.26 ↑ 6.10
	Agostinelli et al. [8]_ImageNet	85.73 ↑ 6.63	79.55 ↑ 6.29	75.60 ↑ 6.44
	Agostinelli et al. [8]_CT	81.81 ↑ 2.71	75.46 ↑ 2.20	72.11 ↑ 2.95

Table 3: Classification accuracy (%) of compressed noisy images for **MNIST**, **Fashion MNIST**, and **CIFAR-10** datasets. Each cell is composed of seven lines that are respectively related to classification : without (w/o) noise processing, and with noise processing using the proposed method, gaussian filter, median filter, bilateral filter, and Agostinelli et al. [8] trained on datasets respectively similar to, i.e., EMNIST and ImageNet, and different from, i.e., CT, test samples.

tracted from the input noisy compressed images, in order to extract clearer versions that are likely to be correctly classified. Extensive experimental results on different benchmark datasets clearly demonstrate the superior accuracy of our method over main state-of-the-art methods. These promising results make the proposed method potentially useful for computer vision systems, where images are voluminous and highly exposed to several noises. The per noise experiments have been also conducted. It is concluded that the classification performances are susceptible to all noise types, albeit to varying degrees. Gaussian and Uniform, which are amplifier and quantization noises, affect the resilience of CNN when classifying compressed noisy images more than Poisson, Speckle, and Salt & Pepper ones, that are shot, multiplicative and impulse noises. Several future research directions are opened from this study. First, the simple autoencoder investigated in this study is found efficient. Various autoencoders rather than the simple counterparts can also be employed to check whether the noise model training process improves or not. Second, future researches can be conducted by leveraging the super-resolution techniques in order to retrieve details from noisy compressed images, so that we get better CNN performance.

7 REFERENCES

[1] M. Dejean-Servières, K. Desnos, K. Abdelouahab, W. Hamidouche, L. Morin, and M. Pelcat, "Study of the Impact of Standard Image Compression Techniques on Performance of Image Classification with a Convolutional Neural," Network. HAL Archives, 2018.

[2] Z. Chen, W. Lin, S. Wang, L. Xu, and L. Li, "Image Quality Assessment Guided Deep Neural Networks Training," arXiv:1708.03880v1, 2017.

[3] O. K. Al-Shaykh, and R. M. Mersereau, "Lossy compression of noisy images," IEEE Transactions on Image Processing, vol. 7, issue 12, pp. 1641–1652, 1998.

[4] G. Valenzise, A. Purica, V. Hulusic, and M. Cagnazzo, "Quality Assessment of Deep-Learning-Based Image Compression," IEEE 20^{th} International Workshop on Multimedia Signal Processing, 2018.

[5] A. Majumdar, "Blind Denoising Autoencoder," IEEE Transactions on Neural Networks and Learning Systems, vol. 30, issue 1, pp. 312–317, 2019.

[6] D. Sil, A. Dutta, and A. Chandra, "Convolutional Neural Networks for Noise Classification and Denoising of Images," IEEE Region 10 Conference (TENCON), 2019.

[7] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, "Toward Convolutional Blind Denoising of Real Photographs," IEEE Conference on CVPR, 2019.

[8] F. Agostinelli, M. R. Anderson, and H. Lee, "Adaptive multi-column deep neural networks with application to robust image denoising," Advances in Neural Information Processing Systems, 2013.

[9] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," IEEE Transactions on Image Processing, vol. 26, issue 7, pp. 3142–3155, 2017.

[10] S. S. Roy, Sk. I. Hossain, M. A. H. Akhand, and K. Murase, "A Robust System for Noisy Image Classification Combining Denoising Autoencoder and Convolutional Neural Network," International Journal of Advanced Computer Science and Applications, vol. 9, issue 1, pp. 224-235, 2018.

[11] kaggle.com/elcaiseri/mnist-simple-cnn-keras-accuracy-0-99-top-1. Last accessed : 09/10/2020.

[12] kaggle.com/fuzzywizard/fashion-mnist-cnn-keras-accuracy-93. Last accessed : 09/10/2020.

[13] appliedmachinelearning.blog/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks. Last accessed : 09/10/2020.

[14] image-net.org. Last accessed : 19/12/2020.

[15] nist.gov/itl/products-and-services/emnist-dataset. Last accessed : 19/12/2020.

[16] E. Soares, P. Angelov, S. Biaso, M. H. Froes, D. K. Abe, "SARS-CoV-2 CT-scan dataset: A large dataset of real patients ct scans for sars-cov-2 identification," medRxiv, May 2020.

Calibration and Auto-Refinement for Light Field Cameras

Yuriy Anisimov[1,2], Gerd Reis[1], Didier Stricker[1,2]

[1]German Research Center for Artificial Intelligence, Trippstadter Str. 122 67663 Kaiserslautern, Germany

[2]University of Kaiserslautern, Gottlieb-Daimler-Str. 47 67663 Kaiserslautern, Germany

{firstname.lastname}@dfki.de

ABSTRACT

The ability to create an accurate three-dimensional reconstruction of a captured scene draws attention to the principles of light fields. This paper presents an approach for light field camera calibration and rectification, based on pairwise pattern-based parameters extraction. It is followed by a correspondence-based algorithm for camera parameters refinement from arbitrary scenes using the triangulation filter and nonlinear optimization. The effectiveness of our approach is validated on both real and synthetic data.

Keywords

light field, camera calibration, camera rectification, calibration refinement

1 INTRODUCTION

Light field cameras [Ng05a] utilize a multi-view principle, focusing incoming light on the image sensor over a grid of lenses. It creates a set of proportionally shifted images that can be used to reconstruct a three-dimensional representation of the captured scene.

These are two types of light field cameras. As proposed by Adelson and Wang in [Ade92a], a light field camera can be created from an array of micro-lenses placed in front of the camera sensor. An alternative way of constructing the light field camera was presented by Wilburn *et al.* in [Wil05a] and involves placing an array of ordinary cameras with predetermined proportional distances between them. A different principle, which can be described as intermediate between the previous two, is represented in [Ani19a]. There, the light field camera is built on a single camera sensor with an array of full-size lenses placed in front of it.

Among the various applications of light fields, the estimation of depth maps attracts particular attention, as the large number of viewpoints increases the quality of the reconstruction and the simple geometry of the light fields simplifies the calculations. Due to the inaccuracies in the placement of light field camera components (micro-lens array or single-view cameras) the direct estimation of scene depth information will not be accurate. Natural lens distortion also affects the quality of the result. There are two ways to solve these problems. For the reconstruction algorithm, either the exact lens positions and geometry should be taken into consideration, or the images from such a camera can be rectified to assure constraints of images' common rotation and proportional placement.

One of the simplifications in reconstruction from multi-view systems has to do with limiting the search for matching correspondences to specific scan lines instead of searching the entire image. Therefore, the rectification-based way of lens placement compensation should be used in the multi-view systems if their view placement allows the rectification without vanishing of significant image plane parts.

Camera calibration, or camera resectioning [Har03a], is a process of retrieving camera intrinsic and extrinsic parameters. Usually, the intrinsic parameters include the focal length, coordinates of a principal point, axis skew, and distortion coefficients. Extrinsic parameters contain information about the pose of the view in the form of rotation and translation *w.r.t.* the scene origin.

These parameters may change slightly under the mechanical or thermal influences during the operation of the light field cameras. In general, the light field camera must be recalibrated to cope with such changes, but sometimes this is not possible due to camera operating limitations or conditions. In such cases, the calibration data can be checked and corrected using arbitrary scene information.

We present an approach for light field camera calibration and rectification, which is based on a well-known pattern-based method from Zhengyou Zhang [Zha00a]. The results of the calibration are used to generate look-up tables that are applied to the light field views for their rectification. The key feature of the method is its simplicity of implementation since this calibration requires a checkerboard pattern that can be produced on any conventional printer.

At the same time, designed a nonlinear optimization model for estimating the compensation of changes in the position of the light field camera lenses. This method uses features extracted from an arbitrary scene.

To improve their accuracy, we introduce a triangulation-based correspondences filtering method, as well as a chaining method for tracking them in different light field views.

Our algorithms were verified on a light field camera with single full-size lenses in an array. It can be potentially used in micro-lens-based cameras, however, the potential misplacement of lenses in a manufactured array is not as large as in cameras with full-size lenses.

The paper is structured as follows. An overview of previous work is given in Section 2, Sections 3 and 4 outline the proposed methods, and experimental results for them are presented in Section 5, followed by a conclusion in Section 6. To prevent the term "camera" from being unambiguous, we use it in reference to a light-field camera; and each image taken from a particular lens is called a "view".

2 RELATED WORK

The first paper describing the principles of light field calibration for camera arrays was proposed by Vaish *et al.* in [Vai04a]. Using planar parallax, the authors retrieve the relative positions of the light field views and use them for further processing.

Extension of stereo calibration and rectification principles for multi-view cameras is presented in the work of Kang *et al.* [Kan08a]. In our work similar principles were used for defining the common values of rectification parameters. Approach of Xu *et al.* [Xu15c] optimizes all parameters of light field camera simultaneously by constrained bundle adjustment model.

Most of the approaches for light field calibration consider usage of micro-lens-based cameras. They can't be applied directly to camera arrays due to the fact that the shift of the lenses relative to each other in such cameras is much higher than in the plenoptic cameras.

In the work of Bergamasco *et al.* [Ber15a] a parameter-free camera model is used for the light field rays representation for further triangulation-based calibration. An approach of Jin *et al.* [Jin16a] estimates the centers of sub-images, aligns all views in a grid, and performs

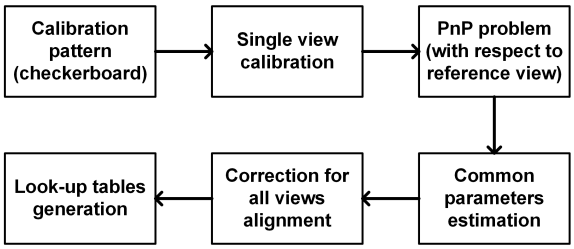


Figure 1: A pipeline of pattern-based calibration algorithm

the rotation of all views to a common plane. Bok *et al.* in [Bok16a] extracts lines from the raw-captured calibration pattern and attempts to compensate for lens distortion and misplacement inaccuracies using these features.

Likewise, Noury *et al.* in [Nou17a] employ the raw images; however, they use corners as the features for calibration procedure along with nonlinear optimization of the result. Method of O'Brien *et al.* [Obr18a] extracts disc features instead of conventional corners for retrieving the light field calibration data.

An approach by Sun *et al.* [Sun19a] works with three calibration targets at different distances together with gradient-based correspondences search. In the publication of Zhou *et al.* [Zho19b], the calibration problem is solved by estimating the original homography solution using a calibration scheme and its subsequent nonlinear optimization. Ji and Wu in [Ji19c] propose a calibration model for plenoptic cameras calibration with a conventional calibration target.

3 CALIBRATION ALGORITHM

A calibration algorithm estimates the intrinsic and extrinsic parameters of all light field views. This data is further used for the views rectification, which creates constrains for simplification of the depth estimation. The pipeline of this method is presented in Fig. 1.

3.1 Camera model

For a separated view of the light field a pinhole camera model is used [Har03a]. The general relation of two-dimensional (2D) image points and three-dimensional (3D) scene points can be described as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

$$P = K[R|t],\ m = PM \quad (2)$$

where m corresponds to homogeneous image point, K – camera calibration matrix, which consists of pixel focal

lengths f_x and f_y, principal point coordinates c_x and c_y, and axis skew s. R and t correspond to rotation matrix and translation vector respectively. These components are assembled into a projection matrix P. M stands for homogeneous world point.

This representation does not take the lens distortions into account. Therefore, in our model we compensate the radial distortions up to their second order by [Har03a]:

$$\begin{aligned} x &= x_c + (x_n - x_c)(1 + k_1 r^2 + k_2 r^4) \\ y &= y_c + (y_n - y_c)(1 + k_1 r^2 + k_2 r^4) \\ r^2 &= (x_n - x_c)^2 + (y_n - y_c)^2 \end{aligned} \tag{3}$$

where $\{x;y\}$ are the undistorted points, $\{x_n;y_n\}$ correspond to original image points, $(x_c;y_c)$ stands for radial distortion center, r is the distance from this center to the distorted image point, and k_1 and k_2 are the radial distortion coefficients.

3.2 Pattern-Based Calibration

A method, which uses a planar calibration pattern, was originally described by Zhengyou Zhang in [Zha00a]. The interest points of a calibration pattern should be such that they can be easily found by a corner detection algorithm. A pattern with a known structure, such as a checkerboard, is used. The measured physical size of the pattern elements is required for solving a task of view position determination with respect to the calibration target. It is used further to provide the correct values of the focal length, with which the pattern was captured.

First, the corners of the located checkerboard need to be found. Due to the high contrast of the black-and-white pattern, its elements can be easily segmented based on histogram analysis. Corners' position is determined by the algorithm from [Suz85a]. A sub-pixel refinement step, described in [Foe87a], is used for obtaining accurate correspondences. This information is used to find the optimal intrinsic parameters of each view.

Having a set of points $\{m\}$, which size corresponds to the number of corners detected in the checkerboard, and its 3-dimensional matches $\{M\}$, common for all images, the algorithm computes the intrinsic matrix K and the distortion coefficients vector d in a way that these values reduce the reprojection error between the original and projected points with K and d, assuming that images are taken with zero rotation and translation *w.r.t.* scene origin.

By using Eq. 2 with estimated intrinsic matrix K, and $R = I$, $t = [0\ 0\ 0]$, where I stands for identity matrix of size 3×3, the reprojections of 3D points can be found. They are used afterward for estimating the root-mean-square error between originally detected points and the reprojected ones.

The 3D coordinates of pattern points $\{M\}$ are assumed to be known. Their placement is defined by physical distances between real-world corners of the checkerboard, which can be expressed from the size of squares.

The algorithm results in the intrinsic values (focal length, camera center) of each light field view together with the distortion coefficient per lens. In order to reduce the reprojection error, the result is further optimized using the Levenberg-Marquardt algorithm [Mar63].

It has been empirically established that this method requires dozens of images with different pattern positions to accurately estimate the calibration parameters. The order of magnitude of this number can be explained by the need for the entire image space to be covered by the sum of the pattern positions among all captured frames.

Results of single-camera calibration are further used for estimation of the relative parameters of every view *w.r.t.* the reference. During this step we fix extrinsic parameters of the reference view, so that $R_{ref} = I, t_{ref} = [0\ 0\ 0]$, and compute relative rotation and translation of every other view $\{R_i;t_i\}, i = 1..N$, where N is a total number of views in the camera, based on their position in relation to scene center. It is possible since all views capture the same pattern position in every image.

By having an assumption of known 3D points the Perspective-n-Point (PnP) problem [Fis81a] is solved for obtaining the rotation and translation of every view *w.r.t.* reference. As before, the results are subject to Levenberg-Marquardt optimization.

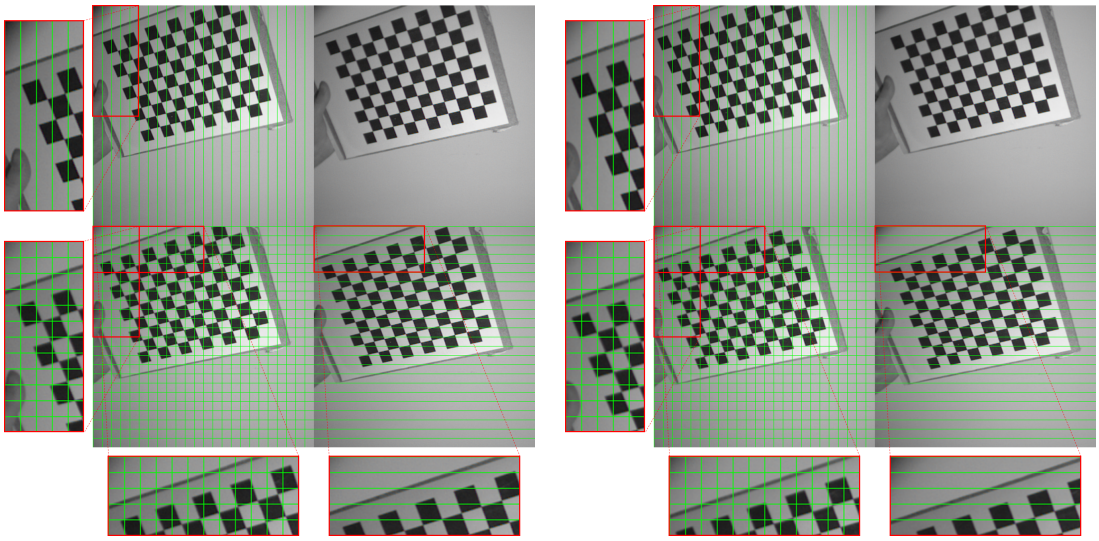
In the end, we obtain a set of original rotation and translation vectors $\{r^O, t^O\}$, which are used for the processing in Section 4. Rotation vectors are obtained from rotations matrices R^O by applying the Rodrigues transformation [Rod40a]. It is used for the simplicity of calculations in the next steps.

3.3 Multi-View Rectification

The next step of the presented algorithm is the estimation of optimal values of the intrinsic and extrinsic parameters for all views. It is necessary for the alignment of all views on the same image plane with proportional distances between every view.

First, all light field views should be rectified with a common intrinsic matrix K_r, which is defined as the following:

$$\begin{aligned} f_r^x = \frac{\sum_{i=1}^N f_i^x}{N}; f_r^y = \frac{\sum_{i=1}^N f_i^y}{N}; \\ c_r^x = \frac{w}{2}; c_r^y = \frac{h}{2}; \\ K_r = \begin{bmatrix} f_r^x & 0 & c_r^x \\ 0 & f_r^y & c_r^y \\ 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{4}$$



(a) (b)

Figure 2: A subset of (a) original and (b) rectified light field after applying the algorithm from Section 3. The corners of the checkerboard are properly aligned, which confirms the rectification

where N is the number of light field views, and w,h correspond to width and height of every view.

Rotation and translation vectors $\{r^O, t^O\}$ are used for getting an optimal value of common rotation and translation vector, to which all views would be brought by the rectification.

Common rotation vector r_r is estimated as an average of all rotation vectors as:

$$r_r = \frac{\sum_{i=1}^{N} r_i^O}{N-1}, \tag{5}$$

and further converted to rotation matrix R_r by applying Rodrigues transform.

Computations of common translation vector involve the spatial information of every view. Since the translation vectors are defined *w.r.t.* reference view, we estimate the relative values by involving a and b as vertical and horizontal spatial dimensions of light field, where $a \times b = N$.

For a specific light field view $i = 1..N$, the translation vector t_i^V is formed from per-axis components as $t_i^V = [t_i^{V_x}\ t_i^{V_y}\ t_i^{V_z}]$ and estimated as a mean of all components not in the same row or column as:

$$\begin{gathered} \bar{a} = \lfloor i/a \rfloor; \bar{b} = i \bmod b; \\ t_i^{V_x} = \begin{cases} t_i^{O_x}/(\hat{b}-\bar{b}), \hat{b}-\bar{b} \neq 0 \\ 0, \hat{b}-\bar{b} = 0 \end{cases} \\ t_i^{V_y} = \begin{cases} t_i^{O_y}/(\hat{a}-\bar{a}), \hat{a}-\bar{a} \neq 0 \\ 0, \hat{a}-\bar{a} = 0 \end{cases} \\ t_i^{V_z} = 0 \end{gathered}, \tag{6}$$

where $\hat{a}$ and $\hat{b}$ stand for spatial coordinates of reference view, defined in Section 3.2.

All translation vectors form a set t^V. Common translation vector $\bar{t_r} = [t_r^x\ t_r^y\ 0]$ is estimated as:

$$\begin{gathered} S_{t^x} = \sum_{i=1}^{N} (1 - \delta_{t_i^{V_x},0}); S_{t^y} = \sum_{i=1}^{N} (1 - \delta_{t_i^{V_y},0}) \\ t_r^x = \frac{\sum_{i=1}^{N} t_i^{V_x}}{S_{t^x}}; t_r^y = \frac{\sum_{i=1}^{N} t_i^{V_y}}{S_{t^y}} \end{gathered}, \tag{7}$$

where S_{t_x} and S_{t_y} are numbers of particular non-zero components in t^V, found using Kronecker delta:

$$\delta_{x,y} = \begin{cases} 0, x \neq y \\ 1, x = y \end{cases}. \tag{8}$$

This vector is a basis for the set of per-view translation vectors t^p:

$$t_i^{p_x} = t_r^x(\hat{b}-\bar{b}); t_i^{p_y} = t_r^y(\hat{a}-\bar{a}); t_i^{p_z} = 0. \tag{9}$$

Using the results from Eq. 4–9 we estimate the rectified projection matrix P_r per every light field view $i = 1..N$ as [Har03a]:

$$\begin{gathered} \bar{R_i} = R_r R_i^{O^T} \\ \bar{t_i} = \bar{R_i} t_i^p \\ P_i^r = K_r[\bar{R_i}|\bar{t_i}] \end{gathered} \tag{10}$$

The resulting camera projection matrix represents the ideal position of its corresponding light field view.

The further remapping of pixels to the proper position is done by applying a look-up table, which stores per-pixel coordinates of the rectified image and generated using P_i^r, distortion coefficients and original intrinsic and extrinsic values. Fig. 2 shows, how light field images changes after applying the described pipeline.

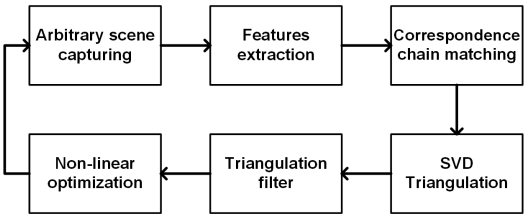


Figure 3: A pipeline of auto-refinement algorithm

4 CALIBRATION AUTO-REFINEMENT

An auto-refinement algorithm uses the previously found reference calibration (Section 3) and tries to estimate, how the calibration parameters need to be compensated for the current configuration of the multi-view camera. It is needed in cases when the placement of views was changed during the camera exploitation, *e.g.* camera is mounted on the car and it is a subject of shakes and other mechanical influences. A pipeline of this method is presented in Fig. 3.

4.1 Correspondence Chain Matching

The important criteria for the selection of correspondences detection algorithm were the robustness of features in real-world images and the running time of the algorithm. Among existing methods, the combination of "Good features to track" method for features extraction [Shi94a] with Kanade-Lucas-Tomasi (KLT) feature tracker [Tom91a] were chosen.

Exploiting the multi-view nature of the light field, in particular the fact that the projection of 3D points from the scene can be seen in all views, the determined features are tracked between different views in a chain manner. Correspondences from the reference view are verified in the neighboring view on the same axis, tracked ones are searched in the next view, and on the last view in one row features are matched in the upper one. This principle is demonstrated in Fig. 4.

Such a method is needed for reducing the number of possibly mismatched correspondences, which may occur especially in the small-baseline multi-view systems, like the camera used in Section 5. This method also helps for the verification of the correspondence inaccuracies, since the very strong matches have to be preserved in all views.

Additional filtration, based on the estimation of fundamental matrix between correspondences in adjacent views, and exclusion of non-matched features is applied. Having a fundamental matrix F, estimated by *e.g.* Random Sample Consensus method [Fis81a], the points from neighboring view x_1 and x_2 are checked by the value of $x_2^T F x_1$ being lower than a certain threshold.

There's a small number of features, which were extracted more than once from the image. To remove the possible influence of such correspondences, we preliminary check the result of the feature detection algorithm

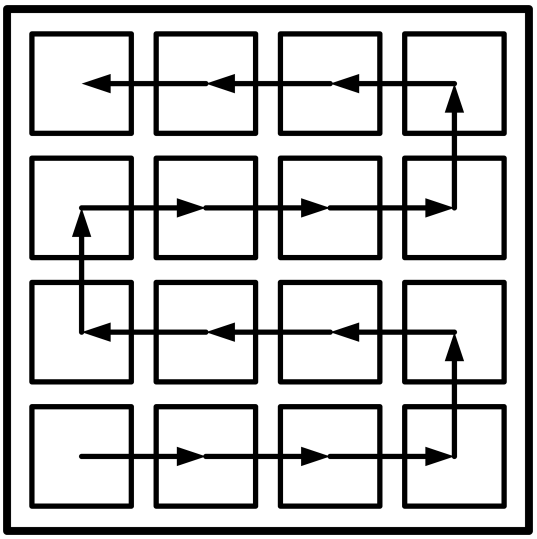


Figure 4: Visualization of views traversing in chain manner

by searching of the nearby correspondences using the Euclidean distance between points.

By empirically setting a certain threshold for these distances we can efficiently filter out the closely placed features. For our experiments it was set to $\sqrt{2}$.

4.2 Triangulation Filter

Previously described methods of filtration can eliminate a big amount of wrongly estimated correspondences. However, some false matches, especially the ones placed close or on the textureless areas, can survive these checks. To eliminate such mismatches we propose a filter, based on the re-projection of triangulated points.

The filtered points are triangulated based on original intrinsic values K_i and rotation and translation vectors R_{Oi}, t_{Oi}, $i = 1..N$. A projection matrix P_i^O is composed as $K_i[R_i^O|t_i^O]$. For every correspondence $m_i = [x_i, y_i]$, matched in every view out of N, by taking a vector P_{oi}^{rT} for every row of the corresponding projection matrix the matrix A is composed as follows [Har03a]:

$$A = \begin{bmatrix} x_1 P_{O1}^{3T} - P_{O1}^{1T} \\ y_1 P_{O1}^{3T} - P_{O1}^{2T} \\ \vdots \\ x_N P_{ON}^{3T} - P_{ON}^{1T} \\ y_N P_{ON}^{3T} - P_{ON}^{2T} \end{bmatrix} \tag{11}$$

Singular Value Decomposition is applied to this matrix, the triangulated points are extracted from the smallest singular value of A. These points are used afterward for the optimization result in Section 4.3.

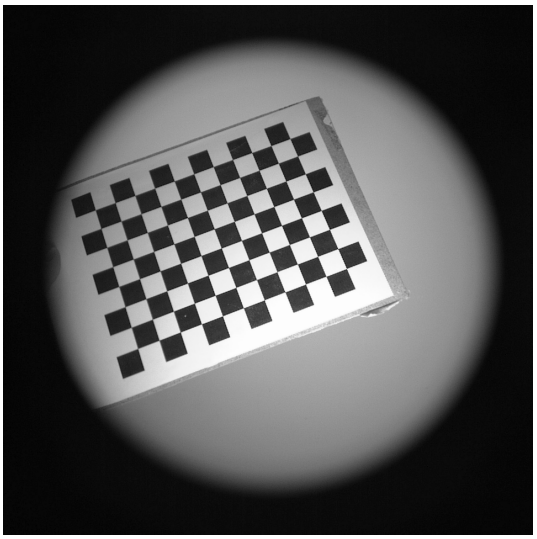


Figure 5: An example of checkerboard pattern, used for light field camera calibration

Using the P_O we project the triangulated points M_t to 2D space and estimate the Euclidean distance between original and projected points:

$$\begin{aligned} m_t &= PM_t \\ dist_t &= \|m_t - m_r\|, \end{aligned} \tag{12}$$

where m_r stands for an original correspondence from reference view.

We find a median of all distances between each correspondence and its projection, and filter out all matches, for which the distance is bigger than the median value.

4.3 Bundle Adjustment

The bundle adjustment problem [Tri99a] is solved for estimating the compensation intrinsic and extrinsic values. For every initially found point x_{ik}, $k = 1..M$, where M is a total number of points, in all views $i = 1..N$ we try to minimize reprojection error as:

$$\sum_{i=1}^{N}\sum_{k=1}^{M} \|x_{ik} - Q(X_k, K_i, d_i, R_i, t_i)\| \tag{13}$$

, where $Q()$ projects the 3D points to 2D plane, involving Eq. 1 – 3.

5 EXPERIMENTS

5.1 Calibration algorithm

For the tests of pattern-based calibration algorithm we used a light field camera from [Ani19a]. It composed of 4x4 lenses, providing a 960x960 pixels RGB image per view, which afterward converted to grayscale for the sake of calculations simplicity. A 12x9 checkerboard pattern with 20x20 mm squares was used for the calibration. An example of the calibration pattern is

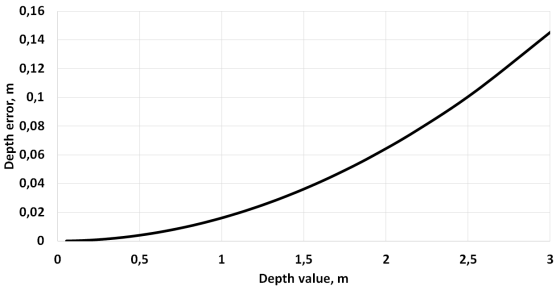


Figure 6: Depth error for $E_{PnP} = 0.246$ pixels

presented on Fig. 5. We captured 40 scenes with this pattern for testing purposes.

To control the accuracy of the estimated intrinsic matrix K and distortion coefficients d, the reprojection error is computed as in Eq. 13 for every light field view. Similar computations are done during the estimation of the extrinsic values.

For the test dataset the average reprojection errors for monocular calibration and PnP problem were $E_{mono} = 0.159$ and $E_{PnP} = 0.246$ pixels respectively.

A comparison of our method was done with the method of Xu *et al.* [Xu15c]. For the same test dataset we have obtained reprojection errors of $E_{mono} = 0.153$ and $E_{PnP} = 0.156$ pixels.

We can state that optimization of all parameters together for the precisely estimated points make sense in terms of accuracy. However, the potential drawback is related to higher running time of the joined optimization.

To verify influence of the reprojection error to the actual depth reconstruction we found a depth error with common focal length values $f = f_x = f_y = 850$ pix and baseline between the two most distant views on the same axis $b = 0.018$ m as:

$$\Delta Z = \frac{fb}{d - E_{pnp}} + \frac{fb}{d + E_{pnp}}, \tag{14}$$

, where d is the disparity value, *i.e.* displacement between pixels, which can be further converted to actual depth value. This error is visualized on Fig. 6.

On a target range of the test camera, which is 0.5-2.0 m [Ani19a], the estimated reprojection introduces an inaccuracy in the amount of 0.004-0.065 m, which is lower then the depth accuracy on this specific distance range.

5.2 Auto-refinement algorithm

A dataset of synthetic light fields was used for verification of auto-refinement stability. The rectified and undistorted images with known intrinsic parameters were generated using Blender and the

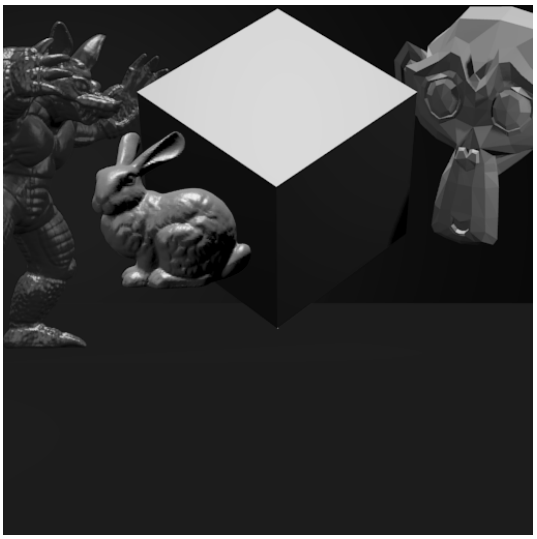


Figure 7: An example of synthetic scene for verifying the auto-refinement algorithm

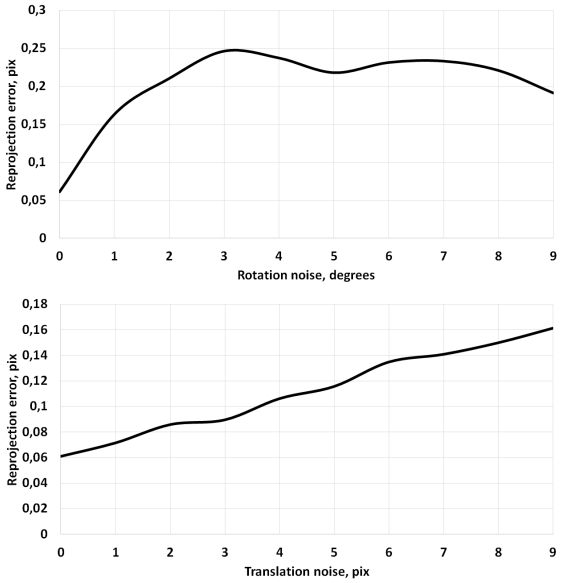


Figure 8: Reprojection error of auto-refinement algorithm dependently of the applied noise for synthetic images

script from 4-dimensional Light Field Benchmark [4DLFB] [Hon16a]. 5x5 light fields of size 512x512 pixels per view with a baseline of 100 mm between adjacent views were generated from a simple scene with different overlapping objects, as demonstrated in Fig. 7. A sequence of ten images was used for the tests.

In average, 1/10 of originally detected points were filtered by correspondence chain matching, out of which half of the points survived the triangulation-based filtration. Processing of one frame takes around one second.

The difference between original and refined intrinsic and extrinsic parameters was measured and considered as non-informative, since no significant difference between original and refined values was found. It can be explained by the quality of correspondences, which is in general good for the synthetic data.

To check the accuracy of the auto-refinement algorithm, we applied rotation and translation noise to the captured frames and measured the average reprojection errors. Results are demonstrated on Fig. 8.

In total both types of noise create acceptable level of reprojection error. In a similar to pattern-based calibration manner we have evaluated the depth error for maximum reprojection error from the rotation noise, result of which is presented on Fig. 9.

For the noise simulation of rotation was applied only to Z-axis. Rotations on X and Y axes are the subject of tangential distortion. It occurs when the lens array is not parallel to the camera sensor plane. This type of distortion is assumed to be zero in the applied camera model.

5.3 Discussion

The pattern-based calibration method used gives adequate results in terms of the resulting reprojection error. Empirically, we have found that the overall calibration quality depends largely on the quality of the calibration target, especially its flatness.

For the proper calibration we have come to the number of 25-30 pattern images in one sequence. All areas on the light field views should be covered with pattern images in various positions to ensure correct estimation of internal values.

One of the assumptions of the algorithms was the similarity of the lens parameters for each view. For cases with a significant shift (in terms of rotation or translation) of at least one of the views over the others, averaging over extrinsic values cannot be used; it should be replaced by nonlinear methods.

Experiments with the auto-refinement algorithm on synthetic images show that the reprojection error increases in proportion to the level of lens shift, while changes in their rotation affect only up to some point with a plateau thereafter.

During the auto-refinement experiments, we noticed that optimizing all the parameters together leads to incorrect results. This was the motivation for dividing the optimization procedure into three steps applied to the same model. First, only 3D points are optimized, while intrinsic and extrinsic camera values are fixed. This condition is relaxed in the second part, where the intrinsic values of all views are optimized. Finally, we optimize all camera parameters together. All of the above steps are repeated with new captured scenes. It can be

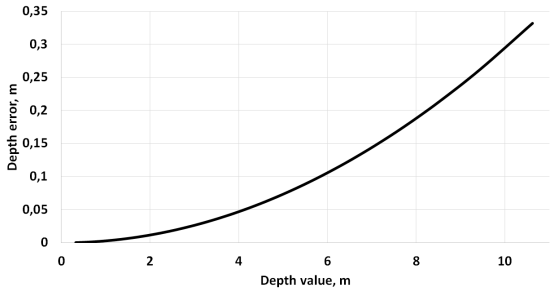


Figure 9: Auto-refinement depth error

stopped either by using a certain number of iterations or by reaching the desired reprojection error below a threshold value.

Several limitations of the auto-refinement algorithm were identified during the tests. It does not work well with repeating textures and with small distances between detected matches, *e.g.* on keyboards. This problem can be solved either by applying additional filtering measures or by changing the match detection algorithm. In addition, the coverage of the image area by the correspondence is important to obtain correct lens geometry, a similar requirement stands for pattern-based calibration. Additional correspondence distribution checks can be made to discard images without proper coverage. Because of this, the distortion coefficients cannot be correctly corrected by our auto-rectification method and remain locked during the optimization stage.

We have tested the auto-refinement algorithm with and without triangulation filter. Without filtering the results were totally incorrect, so they are not included in part of the experiments.

5.4 Implementation

The algorithm was implemented in C++, uses only a central processing unit (CPU), and requires additional libraries such as OpenCV [OpenCV] for image-related routines and Ceres Solver [Ceres] for solving nonlinear optimization problems. The algorithm was tested in Windows 7 with Intel Xeon CPU E3-1245 V2, additional tests were done on the NVIDIA Jetson TX2 and AGX Xavier platforms. Memory requirements directly depend on the size of the provided light field.

6 CONCLUSION

In this paper we presented the calibration and rectification pipeline for light field cameras together with the method for calibration parameters refinement from the arbitrary scenes. Additional filtration measures like triangulation filter and two-dimensional technics were outlined. The algorithms were evaluated using synthetic and real-world data, the results of the experiments support the idea of algorithms' utilization for real-world light field calibration and its refinement. The presented calibration and auto-refinement principles can be further extended to other multi-view systems with preserved similarity of views alignment. To further our research we plan to overcome the mentioned limitations and conduct additional experiments on other similar multi-view systems. For example, method from [Ddv08a] can be adapted to deal with distortions from arbitrary scene.

7 ACKNOWLEDGMENTS

This work was partially funded by the Federal Ministry of Education and Research (Germany) in the context of projects DAKARA (13N14318) and VIDETE (01IW18002). The authors are grateful to Jason Raphael Rambach and Torben Fetzer for the provided support.

8 REFERENCES

[Ade92a] E. H. Adelson, and J. Y. A. Wang. Single lens stereo with a plenoptic camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 99–106, IEEE, 1992.

[Ani19a] Y. Anisimov, O. Wasenmüller, and D. Stricker. A compact light field camera for real-time depth estimation *In International Conference on Computer Analysis of Images and Patterns*, pp. 52–63. Springer, 2019.

[Ber15a] F. Bergamasco, A. Albarelli, L. Cosmo, A. Torsello, E. Rodola, and D. Cremers, Adopting an unconstrained ray model in light-field cameras for 3D shape reconstruction. *2015 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3003–3012, IEEE, 2015.

[Bok16a] Y. Bok, H.-G. Jeon, and I. S. Kweon. Geometric calibration of micro-lens-based light field cameras using line features. *2016 IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 39(2), pp. 287–300, 2016.

[Ddv08a] F. Ddvernay, and O. Faugeras. Straight lines have to be straight: Automatic calibration and removal of distortion from scenes of structured environments. *Machine Vision and Applications*, vol. 5, pp. 14–24 Springer Verlag, 2013.

[Fis81a] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, vol. 24(6), pp. 381–395, 1981.

[Foe87a] W. Förstner, and E. C. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*, pp. 281–305, 1987.

[Har03a] R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. *Cambridge University Press*, ISBN 0521540518, 2003.

[Hon16a] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. *2016 Asian Conference on Computer Vision*, pp. 19–34, Springer, 2016.

[Ji19c] Y. Ji, and J. Wu. Calibration method of light-field camera for photogrammetry application. *Measurement*, vol. 148, pp. 106943, Elsevier, 2019.

[Jin16a] J. Jin, Y. Cao, W. Cai, W. Zheng, and P. Zhou. An effective rectification method for lenselet-based plenoptic cameras. *Optoelectronic Imaging and Multimedia Technology IV*, vol. 10020, SPIE, 2016.

[Kan08a] Y.-S. Kang, C. Lee, and Y.-S. Ho. An efficient rectification algorithm for multi-view images in parallel camera array. *2008 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video*, pp. 61–64. IEEE, 2008.

[Mar63] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, vol. 11(2), pp. 431–441, 1963.

[Ng05a] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. *Doctoral dissertation*, Stanford University, 2005.

[Nou17a] C.-A. Noury, C. Teulière, and M. Dhome. Light-field camera calibration from raw images. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–8, IEEE, 2017.

[Obr18a] S. O'brien, J. Trumpf, V. Ila, and R. Mahony. Calibrating light-field cameras using plenoptic disc features. *2018 International Conference on 3D Vision (3DV)*, pp. 286–294, IEEE, 2018.

[Rod40a] O. Rodrigues. On the geometrical laws that govern the displacements of a solid system in space, and on the change of coordinates resulting from these displacements considered independently of the causes that can produce them. *J Math Pures Appl*, vol. 5, pp. 380–440, 1840.

[Shi94a] J. Shi, and C. Tomasi. Good features to track. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 593–600, IEEE, 1994.

[Sun19a] X. Sun, X. Jin, P. Wang, Y. Chen, and Q. Dai. Blind calibration for focused plenoptic cameras. *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 115–120. IEEE, 2019.

[Suz85a] S. Suzuki. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, vol. 30(1), pp. 32–46. Elsevier, 1985.

[Tom91a] C. Tomasi and T. Kanade. Detection and tracking of point features. *Technical Report CMU-CS-91-132, Carnegie, Mellon University*, 1991.

[Tri99a] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment-a modern synthesis. In *International workshop on vision algorithms*, pp. 298–372, Springer, 1999.

[Vai04a] V. Vaish, B. Wilburn, N. Joshi, and M. Levoy. Using plane+parallax for calibrating dense camera arrays. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2004.

[Wil05a] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. High performance imaging using large camera arrays. *ACM SIGGRAPH 2005 Papers*, pp. 765-776, ACM, 2005.

[Xu15c] Y. Xu, K. Maeno, H. Nagahara, and R.I. Taniguchi. Camera array calibration for light field acquisition. *Frontiers of Computer Science*, vol. 9(5) pp. 691–702, Springer, 2015.

[Zha00a] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22(11) pp. 1330–1334, IEEE, 2000.

[Zho19a] P. Zhou, W. Cai, Y. Yu, Y. Zhang, and G. Zhou. A two-step calibration method of lenslet-based light field cameras. *Optics and Lasers in Engineering*, vol. 115, pp. 190–196, 2019.

[Zho19b] P. Zhou, Z. Yang, W. Cai, Y. Yu, and G. Zhou. Light field calibration and 3d shape measurement based on epipolar-space. *Optics express*, vol. 27(7), pp. 10171–10184, 2019.

[4DLFB] 4d Light Field Benchmark. `http://hci-lightfield.iwr.uni-heidelberg.de`. Accessed: 06.05.2021.

[Ceres] Google Ceres. `http://ceres-solver.org/`. Accessed: 06.05.2021.

[OpenCV] OpenCV. `https://opencv.org/`. Accessed: 06.05.2021.

Service-based Processing of Gigapixel Images

Florian Fregien
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam,
Germany
florian.fregien@student.hpi.de

Sebastian Pasewaldt
Digital Masterpieces GmbH
Potsdam, Germany
mail@digitalmasterpieces.com

Jürgen Döllner
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam,
Germany
juergen.doellner@hpi.de

Matthias Trapp
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam,
Germany
matthias.trapp@hpi.de

ABSTRACT

With the ongoing improvement of digital cameras and smartphones, more and more people can acquire high-resolution digital images. Due to their size and high performance requirements, such Gigapixel Images (GPIs) are often challenging to process and explore compared to conventional low resolution images. To address this problem, this paper presents a service-based approach for GPI processing in a device-independent way using cloud-based processing. For it, the concept, design, and implementation of GPI processing functionality into service-based architectures is presented and evaluated with respect to advantages, limitations, and runtime performance.

Keywords: Gigapixel Images, Image Processing, Web Technologies, Service-based Processing

1 INTRODUCTION

Nowadays, the acquisition of digital images is an essential part of everyday life. With respect to this, the acquisition technologies have been constantly improving, resulting in an increase of spatial and temporal resolution and precision. Especially the increased (spatial) resolution demands for efficient and scalable approaches for their processing and display. Besides desktop-publishing, advertising and marketing, high-resolution images play an important role in the field of cultural heritage [15], medical analysis [11], and geospatial science [20].

1.1 Problem Statement

One example of such high-resolution images are GPIs, which often comprise several billion pixels. Acquisition of such images is usually more complex and its processing and exploration places high demands on a system [8]. Standard consumer hardware is often not sufficient to process and display GPIs due to the high memory requirements. This is especially true for mobile devices such as smartphones that are limited in processing performance and memory.

To approach this problem, specific software is required for managing and distributing performance-intensive processing tasks. Regarding this, advances in cloud-computing technology and cloud-based microservices represent a suitable alternative to monolithic on device software systems.

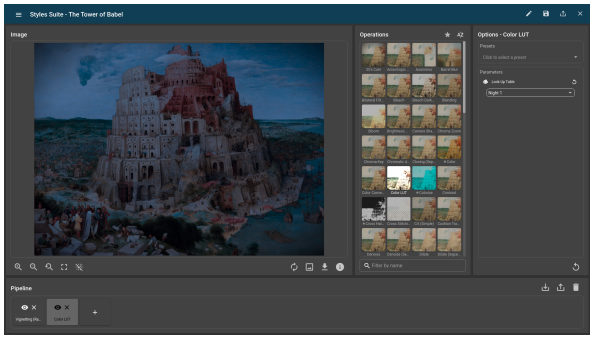


Figure 1: Exemplary screenshot of our service-based application for the exploration and processing of GPIs. It shows the components of our web-based front-end that displays the results of a color grading technique that is applied to a GPI comprising approximately $30\,000 \times 20\,000$ pixels. The provisioning and processing of the GPI was performed using back-end services.

In the context of this work, the technical challenges of integrating a system for processing GPIs into a cloud-based platform is addressed. For this purpose, the representation and storage of GPI have to be addressed first. Further, microservices architectures for image processing are be extended to support GPI, e.g., using a tile-based approach. Furthermore, interfaces are be developed and implemented that enable the communication between front-end and back-end systems.

1.2 Approach and Contributions

This paper describes how the exploration and processing of GPIs can be integrated into to service-based architectures (Figure 1). Therefore, it investigates how such images can be stored and processed in the back-end and accessed by a web-based front-end. It describes the specifics of integrating point-based, neighborhood-based, and global image processing techniques in such a system.

The remainder of this paper is organized as follows. Section 2 describes the concept of GPIs as well as previous and related work on the topic of service-based image processing. Section 3 describes a microservice architecture for processing GPIs. It covers the storage of the images in the back-end, how they can be processed by an image processor, and how the interaction with a front-end is performed in order to display the images efficiently. Section 4 describes the implementation of these concepts and how the processing of the images for point-based, neighborhood-based, and global operations is enabled. Section 5 then evaluates the implementation in terms of runtime and discusses the advantages and disadvantages of the implementation compared to a traditional on-device monolithic approach. Finally, Section 6 summarizes the paper and present future research ideas.

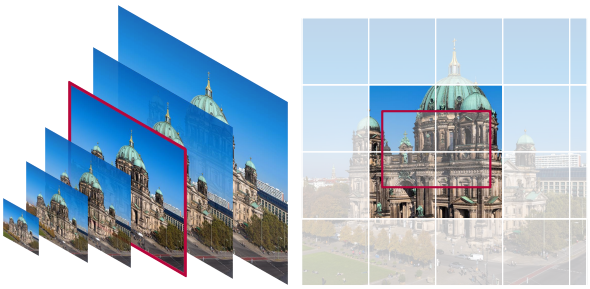
2 RELATED WORK

2.1 Gigapixel Images

While smartphone cameras can capture images with several million pixels, a specific camera setup and software pipeline for automatic composition allows images with several billion pixels to be captured, even with a standard camera and lens [8, 3]. In this work, we refer to the work of Kopf *et al.* [8], who describe how the acquisition and display of GPIs can be implemented. To integrate a GPI-viewer into a service-based architecture, three main requirements are considered: (1) a lossless display to ensure that all information could be extracted, (2) smooth panning & zooming, and (3) responsiveness for an efficient navigation in a GPI [14].

An image pyramid is suitable for storing and loading the image content, whereby the layers are divided into tiles of a smaller number of pixels (Figure 2) [1]. Tile-based approaches enable memory-efficient viewing and processing of GPIs [13]. A GPI viewer loads the tiles closest to the current display first and stores them in a cache. When the user interacts via panning or zooming, the required tiles are loaded from the cache and, if necessary, additional tiles are loaded from the memory. When the cache is full, the tiles furthest away from the current view are removed from it again.

The tile-based approach enables further improvements such as parallel processing of image tiles [4]. Such approaches can also be used to implement image analysis or image segmentation techniques for GPIs [7].



(a) (b)

Figure 2: Concept of image pyramid and tiled image: (a) shows an image pyramid for a GPI. The highlighted layer represents the current zoom level and is used for display; (b) shows the subdivision of the image layer in different tiles. The red bordered rectangle illustrates the viewport and the visible portion of the GPI. Only the tiles contained in the rectangle have to be processed.

2.2 Technologies for GPIs

There are different software libraries available for fetching and displaying GPIs, i.e., for handling such in back-end as well as front-end. The open source project *OpenSeadragon* [2] is based on the Deep Zoom technology, which was originally developed by Seadragon Software and later by Microsoft Live Labs and Google [6]. It is developed in JavaScript (JS) and can be integrated into an existing web application. It supports multiple formats for zooming images and offers various interaction options for users of desktop and mobile devices, including zooming and panning. We forked the project for adjustments required by our microservice platform.

2.3 Microservice Infrastructures

In recent years, microservices have become increasingly popular. Unlike monolithic systems, microservices are autonomous modules that perform various tasks with respect to the business logic. The advantages of using microservices are (1) increased scalability of the components, (2) easy deployment and maintainability as well as, (3) the possibility to introduce various technologies into one system [18].

For our work, we are extending a microservice platform for cloud-based visual analysis and processing that was first presented by Richter *et al.* [16]. Based on that, Wegen *et al.* [19] present an approach for performing service-based image processing using software rendering to balance cost-performance relation.

3 CONCEPT

3.1 GPI Representation

The presented service-based application for image processing is primarily used to stylize photos. We chose the Deep Zoom Image (DZI) format that uses an image pyramid to generate a scale-space of the image. If a user requests to view the entire image for overview, it must be possible to display it on a screen with resolution lower than this of the GPI. A smaller version of the image is better suited for this purpose, as it is also small enough to be loaded into Random Access Memory (RAM). Using the DZI format, the reduction of the resolution per layer is always halved until reaching a 1×1-pixel image. The individual layers are further divided into tiles using a given resolution and overlap (Figure 3). The redundancy introduced by the overlap avoid errors during processing and display.

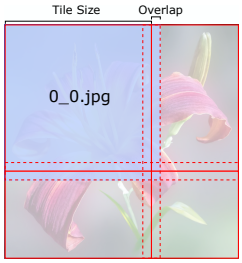


Figure 3: Tiled image as used in DZI format.

Since GPIs have a high spatial resolution, the file size tends to be usually large. With a lossy compression method as used by Joint Picture Experts Group (JPEG) file format, the file size can be kept small. For an image with several Gigapixels (GPs), the file size is still in the Gigabyte range [8]. For this reason, it is feasible not to store these images in a local file system, but to store them in a cloud-based storage solution such as Amazon Web Services (AWS) or Firebase. For our implementation, we decided to use AWS. This allows us to provide a system that is easily scalable and the required storage can be made available as required.

A DZI consists of several files, which are organized and stored in a well-defined structure. For this reason, and to keep communication between different services to a minimum, it makes sense to send these files in one "package" if possible. Of course, additional operations are necessary for a complete DZI, such as requesting the complete DZI for an export functionality or deleting it. Furthermore, we also need to be able to perform operations on individual files of a DZI, such as requesting the definition file or individual tiles for display or image processing. This is a difference to the saving of normal images that we have to consider during the later implementation (Section 4.2).

3.2 Service-based Architecture

To enable processing and exploration of cloud-stored GPIs, a suitable infrastructure is required to enable communication between the following components.

3.2.1 Resource Resolver Service (RRS)

Direct access to a third-party storage management system with a vendor-specific Application Programming Interface (API) is difficult to maintain and extend. To approach this, the RRS service is responsible for managing the files in a storage and abstracts from the vendor specific APIs. It accepts and stores complete DZIs encoded as archives (e.g., ZIP) to preserve the respective folder structure. Thus, a DZI can be transferred with a single message to the service, which can then unpack the archive and store the individual files in the cloud. It also request all files from the storage, compress them into an archive, and deliver them to other services within the system. Furthermore, the RRS can deliver individual files from an archive or accept individual files to add them for an archive export.

3.2.2 Resource Manager Service (RMS)

The RMS serves as an interface between data storage and other services and manages access via user roles. It acts as a gateway and regulates what can be requested from and sent to the RRS service. Specific to GPIs, this service ensures that only valid files are uploaded, i.e., it supports the concept of multi-resolution images to decide whether a request is valid or not. Further, it ensures that only authorized users have access to individual files or to the complete DZI. Finally, if both conditions are met, the service forwards all requested Create, Read, Update, Delete (CRUD) operations to the RRS.

3.2.3 Image Processor Service (IPS)

This service connects the services above and serves as an interface to a front-end. It forwards data requests to the RMS and sends processing requests to the Graphics Processing Unit (GPU)-Processor. The processed results are stored using the RMS and delivered to the front-end for display. Furthermore, it can perform simple image transformations, such as cropping or resizing. The IPS is also responsible for converting images into the DZI format, compositing of DZI tiles, and exporting a GPI.

3.2.4 GPU-Processor

The GPU-Processor is responsible for processing GPI tiles with hardware-accelerated graphics APIs such as OpenGL or Vulkan. For it, (1) tiles are loaded as textures into the Video Random Access Memory (VRAM), (2) the processing operations defined as Visual Computing Assets (VCAs) are applied as shader programs [5], and (3) the results are read back into RAM and returned as a response.

3.3 Interactive Exploration of GPIs

A front-end for processing GPIs should support at least the following functions: (1) selection and upload to the

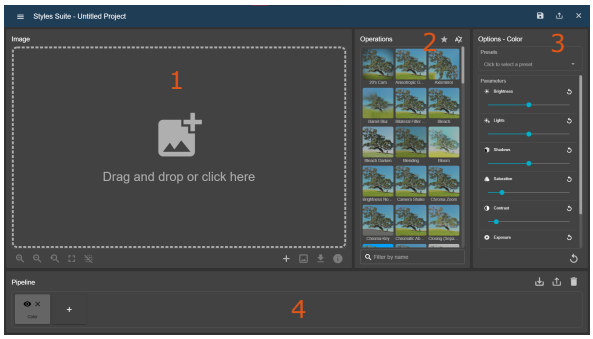


Figure 4: Proposed front-end for displaying and processing GPIs with the following components: (1) Image Canvas, (2) Operations View, (3) Options View, and (4) Pipeline View.

cloud, (2) display and processing, (3) presentation of options for export and download, as well as (4) options to select, compose, and configure different processing operations and their parameters. The following components provide these functionality (Figure 4):

Image Canvas: To address functions 1 to 3, a GPI can be selected and uploaded in a background task. Subsequently, image tiles are requested and displayed accordingly by supporting panning & zooming. For it, the Image Canvas decides which tiles from which image pyramid level should be displayed. Furthermore, actions such as downloading a DZI as an archive, converting it into a normal image, replacing the image, or displaying image metadata can be performed. It also provides explicit control of the zoom levels and full-screen display.

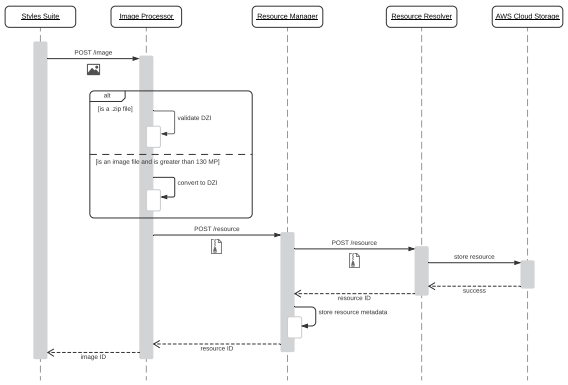
Operation View: To implement function 4, the Operation View depicts a list of processing operations applicable to an GPI. Upon operation selection, it is added to the processing pipeline and the Option View is updated accordingly. Simultaneously, processing is triggered and the result is displayed.

Option View: The Options View displays the various parameters for a selected operation. The user can adjust these as desired and trigger GPI processing subsequently. For easy selection of visually appealing parameter values, presets can be selected.

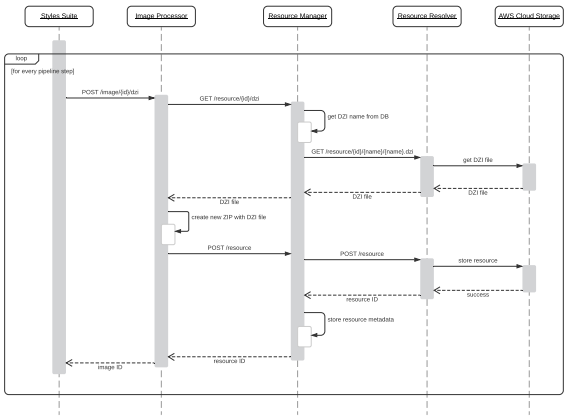
Pipeline View: To allow more complex image transformations, the system allows for combining different operation within a pipeline. For it, the Pipeline View can be used to add and select additional operations from the Operation View. Further actions include clearing the pipeline, and importing or exporting the pipeline.

3.4 GPI Processing Workflow Overview

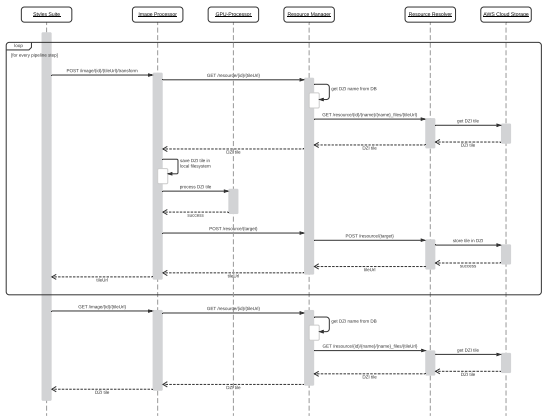
This section describes how the above components interact to process GPIs (cf. Figure 5).



(a) DZI Uploading



(b) GPI Preparation



(c) DZI Processing

Figure 5: Sequence diagrams for the data flow and communication of the respective microservice components for uploading a DZI (a), data preparation for processing (b), and the actual DZI processing (c) (please zoom).

3.4.1 *GPI Upload and Display*

First, the user selects a GPI file via the Image Canvas. The selected file is then sent to the IPS. If the image is available as an archive, it is checked whether it is a valid DZI by validating the definition file and ensuring

that all required tiles are present. If the image has been uploaded as a normal image, the IPS checks if it exceeds a certain number-of-pixels and converts it to the DZI format if necessary. In our implementation, this limit is 130 Megapixels, which is approx. 16K resolution. The valid DZI is then transmitted to the RMS, which forwards it to the RRS. The RRS provides the DZI with an Identifier (ID) and unpacks all files from the archive and stores them under this identifier in the cloud storage.

The ID is sent back to the RMS, which then stores the image in a database with metadata, such as name of the image and ID of the owner. The image ID is then sent back to the IPS, which sends it to the front-end (Figure 5a). Subsequently, the front-end requests the DZI definition file using the ID. The definition file can then be passed to the Seadragon viewer in the Image Canvas. Based on the information from the definition file, the Seadragon viewer computes which image layers and which tiles are available per layer. During interaction, the viewer identifies the tiles to be displayed and request these accordingly. If all required tiles have been requested, they are displayed to the user in the Seadragon viewer as if they were a complete image.

3.4.2 GPI Data-Preparation

A user can create a processing pipeline comprising several operations with different parameters to be applied to the DZI in a non-destructive way. To keep the computation effort low, the individual tiles are only processed on demand. First, a copy of the DZI is created by the IPS for each operation in the pipeline. For it, the IPS requests the respective file from the cloud storage and compresses it into a new archive. This is then stored as a new DZI at the RMS and RRS, the resulting ID is sent to the front-end and stored for the respective pipeline steps. In this way, processed tiles can be written to the new DZIs. Since the original resolution of the image, the tile resolution, and the overlap in the processing steps do not change, the definition file is also the same between all images in the pipeline (Figure 5b).

3.4.3 GPI Processing

Now the definition file of the last pipeline image, which is the result image, can be requested. After reading the definition file, the Seadragon viewer now requests the required tiles that are currently not yet available in the result DZI. For each tile of the requested tiles, the following steps are executed: For each pipeline step, the ID of the previous pipeline image and the tile (level, tile name) is sent to the IPS together with the operation specification (ID, parameters). In addition, the ID of the current pipeline step is sent as the "target" to be able to store the processed tile in the respective pipeline DZI. The IPS now requests the tile from the RMS (and the RMS from the RRS) and sends it to the GPU-Processor together with the operation information. The GPU-Processor applies the operation to the tile. The IPS sends the result back to the RMS to save the tile for the given "target" DZI. When all pipeline steps for a tile have been completed according to this principle, the result tile is finally requested from the IPS via RMS and RRS and subsequently displayed by the Seadragon viewer (Figure 5c).

4 IMPLEMENTATION DETAILS

This section covers development technology (Section 4.1), Representational State Transfer (REST) routes for communication between front- and back-end (Section 4.2), as well as specifics of image processing details (Section 4.3) of our prototypical implementation.

4.1 Back-end & Front-end Technology

For the implementation of the service-based architecture for processing GPIs, the following technologies are used. The respective microservices are developed using JS and executed by the JS runtime environment NodeJS. It offers various modules for handling local file systems or for executing programs as child processes. Further, Express is used to define web service APIs based on REST. The Axios library enables the microservices to communicate over HyperText Transfer Protocol (HTTP). The IPS service communicates with the GPU-Processor via the WebSocket protocol. For high-level image processing, such as reading image metadata or converting images to DZI format, the Sharp library is used [10]. For it, a C++ binding for the VIPS image-processing library [12] provides an API that can be used in NodeJS. As a low-level image processing library, libvips implements more complex operations such as converting a DZI into a normal image by merging the individual tiles. To implement the web application front-end, the Angular framework (based on TypeScript) is used in combination with an adapted fork of the open source library OpenSeadragon.

4.2 Interfacing Front-end & Back-end

This section describes the REST-interface offered by the IPS to handle GPIs using the following routes:

`GET /image/:id` This route returns a DZI addressed by a given ID as a ZIP archive. The IPS service uses the ID to request the image from the RMS. As a parameter, the ID must be specified as a Universally Unique Identifier (UUID).

`POST /image` This route takes as parameter a file which is either a normal image or an DZI archive. In the case of a normal image, the size of the image

is checked. If it exceeds a limit, the image is converted to DZI format using the Sharp library, which provides the `tile()` function for conversion. Then the DZI is transferred to the RMS for cloud storage. The resource ID, obtained by the RMS response, is forwarded to the IPS as a request response.

`GET /image/:id/dzi` This route returns the definition file of a DZI using a given ID. This request is forwarded directly to the RMS.

`GET /image/:id/:level/:tile` This route returns a specific tile of a DZI using a given ID. The name of the DZI is also added here by the RMS. Beside the ID of the DZI the `tileUrl`, which consists of the level of the tile in the image pyramid and the tile coordinates in the form `x_y.format` (where `format` is element of {`png`, `jpeg`, ...}), has to be specified.

`POST /image/:id/dzi` This route copies a DZI of a given ID and removes all files except the definition file from the copy. For it, the IPS service requests the definition file of the DZI with the ID specified in the path, compresses this file into a new archive, and uploads it back to the RMS for storage. The resulting ID is forwarded as response.

`PUT /image/:id/dzi` Similar to the POST route, all files except the definition file (i.e., all tiles) are deleted from the DZI under the ID given in the path.

`GET /image/:id/export` This route exports the DZI using the given ID to a normal image and delivers it as response. It is assumed that at least all tiles of the maximum level of the image pyramid are present. How the tiles are joined to form an image is discussed in Section 4.3.3.

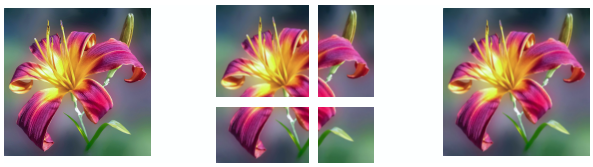
`POST /image/:id/:level/:tile/transform` This route is used to process a tile in the GPU-Processor. The tile with the given `tileUrl` is requested from the DZI with the given ID at the RMS. Further parameters are sent as `FormData` object in the body of the request.

4.3 GPI Processing Operations

4.3.1 Operation Types

There are different types of operations that the GPU-Processor can apply: point-based, neighborhood-based, and global operations.

Point-based Operations: For point-based operations, a pixel is only computed based on the original pixel. An example of a point-based operation is "Grayscale". In a simple implementation, the average of the Red, Green, and Blue (RGB) values of the pixel is computed and assigned to all three channels. Since point-based operations do only operate on one pixel at a time, they can directly be applied to tile-based processing without adjusting the implementation.



(a) Mean Blur operation (kernel of 4 pixels) applied to a DZI level.

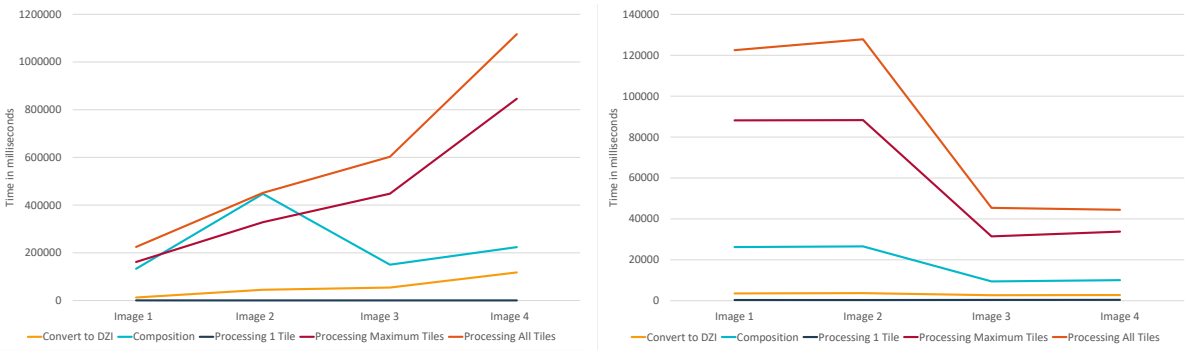
(b) Vignette operation (with a radial size of 0.58 and a radial smoothness of 0.21) applied to one level of a DZI without UV correction.

(c) Vignette operation (with a radial size of 0.58 and a radial smoothness of 0.21) applied to one level of a DZI with UV correction.

Figure 6: Left column shows the original image, middle column shows the operations applied to the single tiles, and right column shows the results after compositing.

Neighborhood-based Operations: These operations use not only the values of a pixel, but also the values of pixels that are in the neighborhood of that pixel. The computation is based on a kernel that can be of any size and weight to determine how the pixels in the neighborhood are contributing. As an example operation, we consider the "Mean Blur" filter. In this operation, the average of all RGB values of the pixels affected by the kernel is computed and set as RGB value for the output pixel. For a larger kernel, the effect is usually stronger, but the computation also takes longer, because more pixels have to be requested.

With tile-based processing, such operations cannot be used as straightforward as for the point-based operations. A pixel at the edge of a tile needs information from the neighboring tile for correct computation because the kernel overlaps. To solve this problem, we can take advantage of the overlap option in the DZI format. The overlap, which is set when an image is converted to a DZI, appends a certain number of overlap-pixels of the neighboring tile (cf. Figure 3). This allows the kernel to access the neighboring pixels for the edge of the tile if the overlap is



(a) Test Case 1 (b) Test Case 2

Figure 7: Duration of operations in milliseconds.

greater than or equal to the kernel radius. Figure 6a shows how to apply the Mean Blur filter to an image consisting of four tiles.

Global Operations: A global operation considers the entire image, e.g., tone-mapping or compositing/blending. Operations that render a texture over the input image use UV coordinates to determine the size and position of the texture. Since the UV coordinates must be in the interval $[0,1]$, the image size is needed to scale the absolute pixel coordinates. Here the coordinate origin is in the lower left corner of the image and in the upper right corner is the point $(1,1)$. An exemplary operation for this category is the "Vignette".

4.3.2 Tile-based Processing

If all tiles are processed individually and such an operation is applied, it will only be computed in relation to these tiles, because the UV coordinates will be determined relative to the size of the tile. This leads to unwanted results, as shown in Figure 6b. To approach the problem, the UV coordinates are recomputed relative to the entire image prior to performing the operation. For it, we pass the original size of the image and the absolute position of the tile to the GPU-Processor.

To compute the correct UV coordinates, two steps are required: (1) the normalized coordinates in $[0,1]^2$ must be scaled relative to the tile resolution by the original size and (2) the coordinates must be translated so that they are relative to the tile in the complete image. For efficiency, these computations are performed on the GPU using vertex shader as follows. First, two quantities that do not depend on vertex coordinates are computed and and represented as uniform variables: (1) the *relative tile resolution* is obtained by dividing the absolute tile resolution in pixels by the size of the original image; (2) similar, absolute coordinates of the tile are converted into relative coordinates. In the vertex shader, these are used by uniform variables for texture sampling, e.g., `u_TileSize` and `u_TilePosition`. For it, first the tile position is flipped, because the origin of the tiles is in the upper left corner, but the origin of the UV coordinates is in the lower left corner. Subsequently, the UV coordinates are scaled to the tile using `u_TileSize` and translated by a tile-offset. During rasterization, the coordinates are passed to the fragment shader(s) that perform the processing. Figure 6c shows the result for the image of Figure 6b using the converted UV coordinates.

4.3.3 Tile Compositing

The export route converts a DZI into a normal image by using the NodeJS library Sharp in combination with libvips. For it, (1) the number of tiles exist at the maximum level of the image pyramid is computed, (2) all respective tiles are requested from the RMS, and (3) stored in the local file system. Subsequently, a child process is spawned for each tile, using libvips to remove the overlap from the tiles. Finally, all tiles are joined to a single output image.

5 RESULTS & DISCUSSION

5.1 Runtime Performance Evaluation

In this section, the runtime performance of our implementation is evaluated regarding two aspects: (1) the time consumed to process GPIs and (2) how the timings depends on different operations regarding the tile resolution and overlap. For both test cases, we use a dedicated GPU-server equipped with a Xeon E5-2637 v4, 3.50 GHz processor (8 cores), 64 GB RAM, and a NVIDIA Quadro M6000 24 GB VRAM.

5.1.1 Test Case 1

Setup: For this test, we measure the runtime for the following operations: (1) image conversion to DZI format, (2) processing of a single tile, (3) processing of all tiles at the maximum level, (4) processing of all tiles of the DZI, and (5) compositing and exporting the tiles to a complete image. For processing, we use a point-based operation, i.e., color adjustments via color

Table 1: Input data for Test Case 1

	Image 1	Image 2	Image 3	Image 4
Resolution (Pixels)	34 561 × 15 620	35 690 × 28 030	34 861 × 44 360	64 172 × 45 559
Gigapixel	0.54 GP	1.00 GP	1.55 GP	2.92 GP
Size (MB)	129.4	894.9	939.6	1601.6

look-up tables [17]. Table 1 shows the four test images and their metadata. The tile resolution is 512 pixels and the overlap is 2 pixels. The timings were determined using the functions `console.time()` and `console.timeEnd()`. Three consecutive measurements were obtained for each operation and the average was calculated subsequently.

Results: Figure 7a shows the duration of the operations in milliseconds. The operations "Convert to DZI", "Processing Maximum Tiles", and "Processing All Tiles" are linear with respect to the input size of the image. The operation "Processing Single Tile" is constant at about 500 ms. This also meets the expectation, since the tiles are all the same size and should therefore be processed at about the same speed. The "Composition" operation also appears to be linear in the size of the images, but Image 2 is an outlier in this measurement. The reason for this must be the file format, which impacts gathering and transmission times. Even if all images are in JPEG format, they can have different compression rates, which can increase the file size of the image and its complexity. The file size (Table 1) also shows this: Although Image 2 has just under twice as many pixels as Image 1, the file size is almost seven times higher. Furthermore, Image 3 has about 50 % more pixels than Image 2 and yet their file sizes are almost the same. However, according to the measurements, this difference does not play a role when converting to the DZI format. This is probably due to the fact that only small parts are removed from the image and stored in a tile. Composition, on the other hand, is the process of merging a large image from all the tiles, which means that the actual saving of the image is more time-consuming.

5.1.2 Test Case 2

Setup: For the second test case we used a GPI of 13 206 × 6676 pixels (0.088 GP, 53.3 MB). For test purposes, we decided to use an image with a smaller resolution. We converted this image to the DZI format with different parameters for tile resolution and overlap. The test sizes are shown in Table 2. For these four variants, we tested the same operations as in Test Case 1 and computed the average from three subsequent measurements.

Table 2: Input resolution in pixels for Test Case 2.

	Image 1	Image 2	Image 3	Image 4
Tile Resolution	256	256	512	512
Overlap	1	16	1	16

Results: Figure 7b shows the absolute duration of the operations in ms, while Figure 8 shows the file sizes of the four DZIs. The measurement results show that the processing of a tile takes between approx. 300 ms to 400 ms. A greater overlap has the consequence that both the file size and the duration of the operations increase. This is because a higher overlap increases the redundancy of the pixels and therefore more has to be stored and calculated. It is noticeable that converting to a DZI was about one second faster with the 512 pixels tiles than with the 256 pixels tiles. The other operations – "Processing Maximum Tiles", "Processing All Tiles", and "Composition" – were even over 60 % faster for the 512 pixels tiles than for the 256 pixels tiles. Because the duration of the operations is directly proportional to the number of tiles, the operations for Image 3 and 4 are faster because there are fewer tiles due to the larger tile resolution.

5.2 Discussion

5.2.1 Service-based vs. Monolithic Approach

In the following, we will briefly discuss the advantages and disadvantages of our service-based implementation compared to a classic approach with only one service. The described microservice infrastructure enabled the development of a modular system following the Separation-of-Concerns pattern [9]. This allows to multiply different parts of the system as required, thus enabling parallel processing that increases performance compared to a monolithic system. Through the connection to the cloud storage service AWS, additional storage is available that can be expanded on demand. A

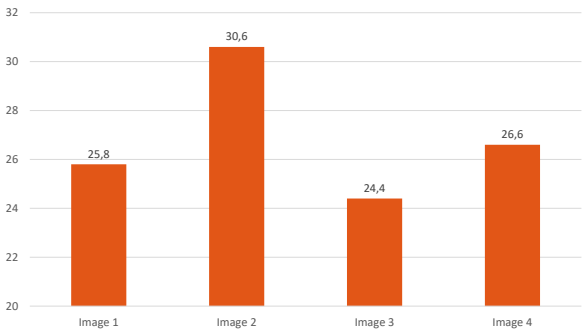


Figure 8: Test Case 2 – File size of converted images in DZI format in Megabytes.

disadvantage of the proposed approach, however, is that GPIs are required to be sent through several services, which increases the overall traffic and increases runtime. In our test system all services run on one server only, i.e., that this server is no less busy than a server on which everything would run as one service anyway. In other words, the potential of our implementation in terms of performance only becomes apparent when the services are used as a distributed system with several servers.

5.2.2 Tile Resolution and Overlap

In our implementation of DZI format conversion, constant values for tile resolution and overlap are used. Changing these parameters later is time-consuming, since the DZI would first have to be reassembled and then converted back into the DZI format. This requires to find suitable values for both parameters to process the tiles as efficiently as possible and to avoid artifacts, e.g., due to kernel sizes of neighborhood operations larger then the actual overlap. The tile resolution should not be chosen too small, because this increases the number of tiles and thus the effort when converting from and to DZI format. However, the overlap should be as small as possible because it only contains additional (redundant) data.

The results of Test Case 2 show that choosing a higher tile resolution can have a positive effect on runtime and memory consumption. This is because the higher tile resolution means that fewer tiles are created during the conversion, and therefore fewer steps per operation have to be performed. However, with a difference of less than 100 ms, the runtime for processing a 512 pixels tile is remarkably close to the runtime for a 256 pixels tile. This means that if the tile resolution is not too large, the runtime can be reduced without having to make major losses when processing a tile.

A higher value for the overlap increases runtime and file size, since more pixels are stored redundantly. A key factor in the choice of the overlap is the kernel size of neighborhood-based operations. If a kernel requires n neighboring pixels in each direction, the overlap should be at least n, to process the edge pixels of the tile correctly.

5.2.3 Context-sensitive Global Operations

Using tile-based processing, our implementation allows for the application of global operations to a DZI. The approach converts the UV coordinates transparent to the processing shaders. However, this approach does not work for global operations that require random access to all parts of an GPI, e.g., “Mirroring”. To approach this problem, one could give the GPU-Processor more image context about a DZI, so that it can request additional tiles if required.

6 CONCLUSIONS & FUTURE WORK

This paper describes how service-based architectures for image processing can be extended to allow the processing of GPI. It shows how the storage of GPIs in the DZI format can be implemented with a cloud-based approach and microservices for transfer, preparation, and tile-based processing. For it, the differences between specific features of point-based, neighborhood-based, and global operations are discussed. A runtime analysis shows that efficient processing and display of GPIs is possible in a service-based infrastructure. This work can serve as a basis for further research and development in this area.

While the presented approach and prototypical implementation is sufficient for processing and exploration of GPIs, the system can be extended further. To enable the processing of high-resolution panoramic images, the support of additional projection methods (e.g., equirectangular or stereographic) are required. A service-based approach has the advantage that even less powerful devices can process images using potentially complex operations. However, this has the disadvantage that additional network traffic is generated by the communication between the individual microservices, which can result in waiting times for the user. Further, the service-based architecture allows the various microservices to be multiplied as required, thus enabling parallel processing of requests. This can also be exploited for processing GPIs, e.g., by processing different tiles in parallel. This however results in further challenges, e.g., how the tiles are transferred or how tile-access is managed between different processors.

ACKNOWLEDGMENTS

We thank Josafat-Mattias Burmeister for his support. This work has been funded by the German Federal Ministry of Education and Research (BMBF) through grants 01IS18092 (“mdViPro”) and 01IS19006 (“KI-Labor ITSE”).

7 REFERENCES

[1] Edward Adelson, Charles Anderson, James Bergen, Peter Burt, and Joan Ogden. Pyramid methods in image processing. *RCA Eng.*, 29, 11 1983.

[2] OpenSeadragon Contributors. Openseadragon 2.4.2, 2021. last visited: 03/30/2021.

[3] O. S. Cossairt, D. Miau, and S. K. Nayar. Gigapixel computational imaging. In *2011 IEEE International Conference on Computational Photography (ICCP)*, pages 1–8, 2011.

[4] Zhenlong Du, Xiaoli Li, Xiaojian Yang, and Kangkang Shen. A parallel multigrid poisson pde solver for gigapixel image editing. In Yunquan

Zhang, Kenli Li, and Zheng Xiao, editors, *High Performance Computing*, pages 89–98, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[5] Tobias Dürschmid, Maximilian Söchting, Amir Semmo, Matthias Trapp, and Jürgen Döllner. ProsumerFX: Mobile Design of Image Stylization Components. In *Proceedings SIGGRAPH ASIA Mobile Graphics and Interactive Applications (MGIA)*, pages 1:1–1:8, New York, 2017. ACM.

[6] Paul Khouri-Saba, Antoine Vandecreme, Mary Brady, Kiran Bhadriraju, and Peter Bajcsy. Deep zoom tool for advanced interactivity with high-resolution images. *SPIE Newsroom*, page 1, 05 2013.

[7] Enrico Kienel and Guido Brunnett. Tile-based Image Forces for Active Contours on GPU. In P. Alliez and M. Magnor, editors, *Eurographics 2009 - Short Papers*. The Eurographics Association, 2009.

[8] Johannes Kopf, Matt Uyttendaele, Oliver Deussen, and Michael Cohen. Capturing and viewing gigapixel images. *ACM Transactions on Graphics (TOG)*, 26:93, 08 2007.

[9] Phillip Laplante. *What Every Engineer Should Know About Software Engineering*. CRC Press, 2007.

[10] libvips Contributors. libvips, 2021. last visited: 03/30/2021.

[11] Yun Liu, Krishna Gadepalli, Mohammad Norouzi, George E. Dahl, Timo Kohlberger, Aleksey Boyko, Subhashini Venugopalan, Aleksei Timofeev, Philip Q. Nelson, Gregory S. Corrado, Jason D. Hipp, Lily Peng, and Martin C. Stumpe. Detecting cancer metastases on gigapixel pathology images. *CoRR*, abs/1703.02442, 2017.

[12] K Martinez and J Cupitt. Vips ? a highly tuned image processing software architecture. In *IEEE International Conference on Image Processing (01/09/05)*, pages 574–577, 2005. Event Dates: Sept. 2005.

[13] Mayur Patel. Memory-constrained image-processing architecture. *Dr. Dobb's Journal (DDJ)*, 22:24, 26–29, 07 1997.

[14] Dominik Perpeet and Jan Wassenberg. Engineering the ideal gigapixel image viewer. In *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS 2011)*, Maui, Hawaii, 12 2011.

[15] Nancy Proctor. The google art project: A new generation of museums on the web? *Curator: The Museum Journal*, 54(2):215–221, 2011.

[16] Marvin Richter, Maximilian Söchting, Amir Semmo, Jürgen Döllner, and Matthias Trapp. Service-based Processing and Provisioning of Image-Abstraction Techniques. In *Proceedings International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 97–106, Plzen, Czech Republic, 2018. Computer Science Research Notes (CSRN).

[17] Jeremy Selan. Using Lookup Tables to Accelerate Color Transformations. In *GPU Gems*, pages 381–392. Addison-Wesley, 2004.

[18] Markos Viggiato, Ricardo Terra, Henrique Rocha, Marco Tulio Valente, and Eduardo Figueiredo. Microservices in practice: A survey study. *CoRR*, abs/1808.04836, 2018.

[19] Ole Wegen, Matthias Trapp, Jürgen Döllner, and Sebastian Pasewaldt. Performance Evaluation and Comparison of Service-based Image Processing based on Software Rendering. In *Proceedings International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 127–136, Plzen, Czech Republic, 2019. Computer Science Research Notes (CSRN).

[20] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. Geosparkviz: A scalable geospatial data visualization framework in the apache spark ecosystem. In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, SSDBM '18, New York, NY, USA, 2018. Association for Computing Machinery.

Transfer Learning and Hyperparameter Optimization for Instance Segmentation with RGB-D Images in Reflective Elevator Environments

Lukas Reithmeier
Research Group Advanced Information Systems and Technology
Softwarepark 11
4232 Hagenberg, Austria
lukas.reithmeier@fh-hagenberg.at

Oliver Krauss
Research Group Advanced Information Systems and Technology
Softwarepark 11
4232 Hagenberg, Austria
oliver.krauss@fh-hagenberg.at

Gerald Adam Zwettler
Dept. of Software Eng., University of Applied Sciences Upper Austria
Softwarepark 11
4232 Hagenberg, Austria
gerald.zwettler@fh-hagenberg.at

ABSTRACT

Elevators, a vital means for urban transportation, are generally lacking proper emergency call systems besides an emergency button. In the case of unconscious or otherwise incapacitated passengers this can lead to lethal situations. A camera-based surveillance system with AI-based alerts utilizing an elevator state machine can help passengers unable to initiate an emergency call. In this research work, the applicability of RGB-D images as input for instance segmentation in the highly reflective environment of an elevator cabin is evaluated. For object segmentation, a Region-based Convolution Neural Network (R-CNN) deep learning model is adapted to use depth input data besides RGB by applying transfer learning, hyperparameter optimization and re-training on a newly prepared elevator image dataset. Evaluations prove that with the chosen strategy, the accuracy of R-CNN instance segmentation is applicable on RGB-D data, thereby resolving lack of image quality in the noise affected and reflective elevator cabins. The mean average precision (mAP) of 0.753 is increased to 0.768 after the incorporation of additional depth data and with additional FuseNet-FPN backbone on RGB-D the mAP is further increased to 0.794. With the proposed instance segmentation model, reliable elevator surveillance becomes feasible as first prototypes and on-road tests proof.

Keywords

Instance Segmentation, RGB-D data, Transfer Learning, Reflective Environments

1 INTRODUCTION

Elevators are an essential part of modern urban life to enable accessibility in buildings with multiple stories. In Austria, public buildings with more than one story require an integrated elevator system, equipped with an emergency button to establish a call center voice connection with minimal effort [Nat06a, Mos18a]. These emergency buttons show a key drawback. When the person in need of help is no longer able to push the button, e.g. an elderly person has fallen and can't get up, or a person has fallen unconscious. Thus, a viable demand for camera-driven and preferably fully-automated emergency detection arises.

In the context of an industry-oriented project, research on the algorithmic and technological foundation for a reliable and automated emergency system is being conducted. With the camera systems mounted in corners of the elevator cabin, unorthodox views on the passengers arise. This point of view is rarely trained in current deep learning (DL) models for instance segmentation. Furthermore, the generally metallic and thus highly reflective nature of the elevator cabins presents a significant challenge for the task of analysing RGB video feed.

The research questions are:

- *Can depth information be used to improve the performance of instance segmentation in highly reflective and noisy elevator environments?*
- *Does incorporation of depth data generally increase the performance of instance segmentation?*
- *Can the use of a FuseNet-FPN as a backbone network together with transfer learning and hyperparameter optimization improve the performance of Mask R-CNN adapting to RGB-D?*

1.1 State of the Art

Instance segmentation, i.e. the detection of all pixels belonging to a specific class instance, is a highly challenging and relevant task in computer vision that has been a focus in research since the very first days of digital imaging. With the use of static camera systems, object segmentation is achievable from background modelling to calculate a difference image from the actual frame [Wu10a] with local intensity variance, motion and overlapping color spectrum hardening practical applicability. In the past, the focus was laid on semi-automated single-feature approaches such as flood-fill [Gon01a, Gom07a] incorporating the image intensities or Live-Wire [Bar97a] contour tracing based on the image edges. With Grab Cut a highly relevant tool for instance segmentation on RGB images was introduced, reducing the demand for user-interaction to zero in optimal scenarios [Rot04a, Tan15a].

Incorporating the expected object shape besides the image characteristics, as proposed for active shape [Gin02a] and active appearance models [Coo01a], allowed advances e.g. in the field of face detection and instance segmentation in general. Thus, knowing the statistical shape of the target structure, segmentation can be implicitly defined as registration problem [Xia16a, All18a]. With registration strategies, deep learning (DL) and frame-to-frame propagation, nowadays instance segmentation in RGB video sequences become feasible as object-tracking tasks [Hou19a].

To date, with the evolution of hardware and availability of machine learning frameworks, instance segmentation is generally considered as a machine learning topic. With more and more hidden layers introduced and the availability of training datasets and pre-trained models, convolutional neural networks became the gold standard in segmentation and classification. A broad range of published network types is available and applicable for various input data and application domains. With the progression of recurrent networks [Rum87a] allowing to construct a spatio-temporal memory of the processed input sequence, cf. long-short-term-memory (LSTM) [Hoc97a], significant advances in optical character recognition (OCR) and video processing in general have been possible [Sab18a].

For the task of image instance segmentation on visual input data, a broad range of different network types have been published in the past. With larger network structures increasing the number of hidden layers, application of simple layer-wise back-propagation leads to the so called vanishing gradient problem, i.e. update influence becomes marginal for the lower layers. This problem is counteracted by He et al. presenting the Residual Network (ResNet) [He16a] introducing bypassing of the updates and thus allowing an increased depth of 100+ layers. The common principle of filter pyramids for instance detection at varying scale is proposed by Lin et al. with the Feature Pyramid Networks (FPN) [Lin16a]. While FPN is practical for detection and classification tasks, pyramid up-scaling as introduced by Encoder/Decoder pattern of the U-net [Ron15a] finally features instance segmentation tasks. Besides using a filter pyramid for multi-resolution object localization, the Region Proposal Network (RPN) [Uij13a, Ren15a] detects objects invariant to translation and scale together with their bounding box and an objectness score as confidence metric. The RPN is thereby introduced by Ren et al. to speed up the R-CNN initially introduced by Graves et al. [Gra13a] then denoted as the Faster R-CNN [Ren15a].

The Fully Convolutional Instance-aware Semantic Segmentation (FCIS) introduced by Li et al. [Li16a] extends the Region-based Fully Convolutional Networks (R-FCN) introduced by Dai et al. [Dai16a] to perform instance segmentation besides object location. With the Mask R-CNN as evolution introduced by He et al. in 2017 [He17a], Faster R-CNN is adapted for instance segmentation, too. Instead of sliding windows, single-shot approaches utilize a scale bipyramid as introduced for TensorMask by Chen [Che19a]. Single-shot approaches are utilized for Single Shot MultiBox Detector (SSD) [Liu15a], RetinaNet [Lin17b] and YOLO [Red15a] respectively in a similar way, too.

Nevertheless, these approaches for instance detection and segmentation show high accuracy in case of good image quality only. With reflections and a bad SNR, even the sophisticated DL models for image processing often fail [Ahm11a]. Thus, specific pre-processing is necessitated, e.g. removing the shadows in traffic surveillance videos [Kil92a] or removing mirror-based reflections [Chi18a]. To overcome visual noise in image data and to improve accuracy in general, incorporation of depth data with RGB-D images is an adaequate strategy [Ye17a].

1.2 Related Work

With RGB-D data provided, Watershed transform [Beu79a] is applicable onto the varying local depth profiles leading to a first rough fragmentation of the scene. These pre-segmented regions can be utilized as input for semantic classifiers to detect humans and classify the body pose. As machine learning classifiers, Support Vectors [Cor95a], Random Forest [Ho95a] and XGBoost [Che16a] are applied.

Although most of the DL models only consider 3-channel RGB data as input, some CNNs incorporating depth data have already been published [Bo13a, Cou13a]. Thus, to incorporate arbitrary DL models for instance segmentation, transfer learning [Wei16a] or hyperparameter optimization [Los16a] are applicable to re-train the model with data augmentation or GAN

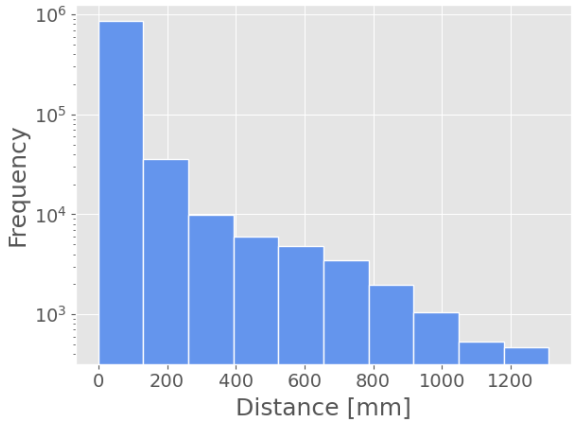


Figure 1: Pixel-wise standard deviation in the depth images of a recording of a closed elevator. The frequency is scaled logarithmically.

synthesis as a potential strategy in case of insufficient amount of training data in medicine [Zwe20a]. For the domain of elevator surveillance, lack of data generally is immanent due to the uncommon perspective.

Depth perception data has already been successfully applied to detect the doors of an elevator, including the difference between elevator and outside floor heights with precision $< 20mm$ and the opening state of the door as an accurate and generic alternative to mechanical sensor systems in the field of predictive maintenance. This is important for emergencies as this can mean that the elevator will not move to another floor if the door is blocked, or that first response help is being provided in case of a medical emergency [Ign99a].

An attempt has also been made to conduct classification purely with RGB data, omitting depth information utilizing different methods such as K-Nearest Neighbor, Support Vector Machine, Random Forest and Neural Networks. This has been shown to work at detection success of 92% in non-reflective environments. The noise introduced by reflections as well as movement of the elevator, i.e. vibrations also apply to the camera and greatly reduces the detection rate [Sti99a].

2 MATERIAL

To train the machine learning models we use two datasets. Since there are no RGB-D datasets containing samples recorded in highly reflective and noisy elevator scenes, we introduce the Elevator RGB-D dataset. In addition we use the SUN RGB-D dataset [Son15a]. The Elevator RGB-D dataset consists of 1138 samples, where each of these samples is composed of a three-channel RGB image, a one-channel depth image and labels. Each label consists of one or more 2D polygons with a corresponding class label.

The RGB and depth images are derived from 21 recorded RGB-D videos from scenes of humans and objects in four different elevators. The RGB-D videos are recorded using the Intel RealSense D-400 RGB-D camera [Kes17a]. One sample per second of recorded video is extracted. Samples without a label are removed from the dataset. The noise in the depth images varies between each elevator. The majority of the pixels in a recording have a very minor standard deviation, as seen in Figure 1. But some pixels in the recording have a standard deviation of over one meter. This recording contains RGB-D images of a closed elevator without changes in lighting conditions.

The depth images are aligned with the RGB image, so that the position of objects in the depth image corresponds to the position of objects in the RGB image. The depth image contains values from 0 to 65535, where the value of a pixel represents the distance from an object to the camera in millimeters.

The samples of the Elevator RGB-D dataset are labeled to contain objects of nine different classes $\{$ *human_standing*, *human_lying*, *human_sitting*, *human_other*, *object*, *box*, *bag*, *chair*, *plant*$\}$, where *object* can be any object that is not a *box*, *bag*, *chair* or *plant*. Class *human_other* describes humans in any pose different to standing/lying/sitting such as e.g. crouching or indistinct/unknown poses. Furthermore, *human_standing* also includes walking humans, as long as they are in an upright pose. The number of objects per class are not distributed evenly, as seen in Figure 2. The object categories such as bag or box are utilized to detect potential hazardous objects left in the elevator cabin, too. Especially at airports static objects need to be treated as a threat and thus triggering an emergency. The label with the most occurrences *human_standing* is thrice as common as the label with the second most occurrences. Samples of this dataset can be seen in Figure 3.

The Elevator RGB-D dataset is inherently biased, since the RGB-D recordings only contained images of 10 different male persons with light skin color wearing various different outfits, also including beards or glasses. This bias can prevent generalization of image recognition models trained using this dataset [Tor11a, Tom15a].

The samples have different sizes, therefore all RGB and depth images are resized and zero-padded to a size of 512×512. The depth images are normalized to a range of 0 to 255 using min-max normalization as described by Patro and Sahu [Pat15a]. Furthermore, to allow processing the Mask R-CNN, each of the polygons of the n labels of a sample is converted to a binary mask per label. The n masks of a sample are combined to an n-channel image. Additionally, a n-dimensional vector with an integer representation c of the class label is derived from the labels with zero encoding background.

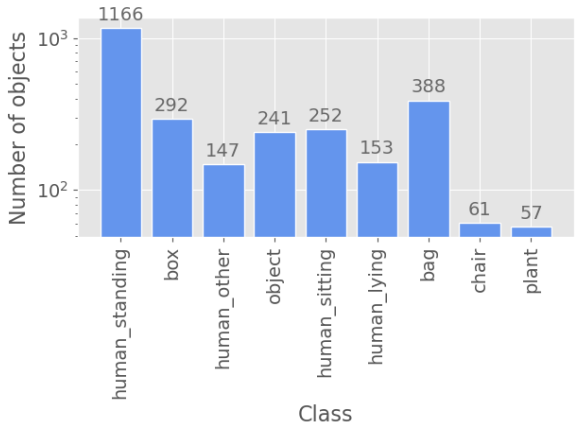


Figure 2: Number of objects per class in the Elevator RGB-D dataset, scaled logarithmically with *human_lying* and *human_sitting* theerby replicating medical emergency situations.

Both the Elevator RGB-D dataset and the SUN RGB-D dataset are randomly split into three different categories for training, validation and testing. The SUN RGB-D dataset has 5285 training samples, 475 validation samples and 4575 test samples. The Elevator RGB-D dataset has 797 training samples, 228 validation samples and 113 test samples, all randomly assigned.

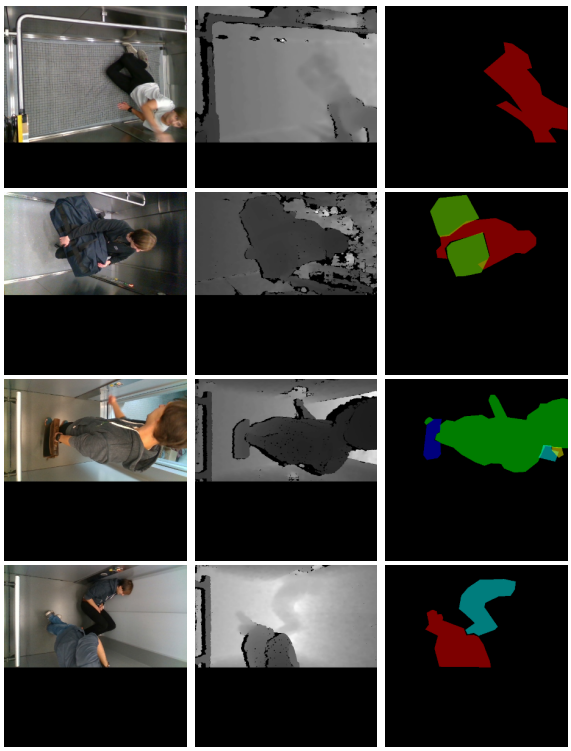


Figure 3: Samples of the Elevator RGB-D dataset. Each sample consists of an RGB image (left), a depth image (middle) and labels (right).

3 METHODS

To answer the question, whether depth data can be used to improve the performance of instance segmentation in highly reflective and noisy elevator environments, three different approaches of the Mask R-CNN model [He17a] are compared. The approach RGB uses solely the three-channel RGB images as input of the Mask R-CNN, the approach D3 uses the depth image duplicated over three channels to achieve tensor conformity at the cost of redundancy, the approach RGBD uses a four-channel image consisting of the three channels of the RGB image and one corresponding depth channel. These three approaches use a ResNet-FPN [He16a] as backbone network. In addition to these three approaches the approach RGBD-F also uses a four-channel RGB-D image as input, but also uses an adapted FuseNet-FPN as a backbone network to answer the question whether this usage can improve the performance of Mask R-CNN with RGB-D images in highly reflective noisy elevator environments.

The adapted FuseNet-FPN consists of the encoder part of the FuseNet introduced by Hazirbas et al. [Haz16a] and thus consists of five different stages. The filter kernels have a size of 7×7 in the first stage and 3×3 in the following stages. The ResNet and the FuseNet are used as a feature pyramid networks (FPN) [Lin17a].

Usually transfer learning using weights of a ResNet-101 or a ResNet-50 trained with the MS COCO dataset [Lin14a] is applied to initialize the Mask R-CNN model [He17a]. But since four different approaches with RGB and depth information are used and the MS COCO dataset only contains RGB information, initializing the weights would undermine the comparability of these approaches. Therefore, the training procedure consists of the following three steps:

1. A hyperparameter optimization is performed using the SUN RGB-D dataset.
2. The models are pre-trained using the optimized hyperparameters and the SUN RGB-D dataset.
3. Transfer learning performed by transferring the weights learned training the models with the SUN RGB-D dataset and further training the models with the Elevator RGB-D dataset.

3.1 Hyperparameter Optimization

The approaches differ in input data and backbone architecture. Thus, their hyperparameters are optimized separately. The hyperparameter space consists of the backbone architecture, the number of regions of interest per image, the minimum detection confidence and choice of the optimizer.

The backbone architecture determines the kind of CNN that is used to extract features from the input image.

The approaches RGB, D3 and RGBD can either use a ResNet101 with a mini-batch size of 1, a ResNet50 with a mini-batch size of 1 or a ResNet50 with a mini-batch size of 2. The backbone architecture of the approach RGBD-F is a FuseNet that is separated into five different stages. The number of filters in each Conv layer for the five stages $(f_1, f_2, f_3, f_4, f_5)$ can be one of the three different configurations:

$$(f_1, f_2, f_3, f_4, f_5) \in \begin{matrix} \{(32, 32, 64, 128, 256), \\ (64, 64, 128, 256, 512), \\ (128, 128, 256, 512, 1024)\}. \end{matrix} \quad (1)$$

The minimum detection confidence determines whether regions of interest are considered to be a valid result. Regions of interest with a class score below the minimum detection confidence are discarded. The empirically chosen minimum detection confidence ranges from 0.6 to 0.8. The maximum number of regions of interest per image can either be 50, 100 or 200. As optimizer either a SGD [Kie52a] with mini-batches and momentum or an ADAM optimizer [Kin14a] can be used. In total, the hyperparameter space consists of 54 different configurations.

Tree-structured Parzen Estimators (TPE) [Ber11a] is used to minimize the total number of evaluations needed for a use-able result. TPE is a sequential model-based optimization algorithm (SMBO) commonly used to optimize hyper parameters of machine learning algorithms. TPE searches a minimum in a surrogate model of the hyperparameter space, which it evaluates by training a machine learning model with said minimum as hyperparameter configuration. The resulting score value is used to adapt the surrogate model. The number of evaluations per approach is chosen to be 10. To evaluate a hyperparameter configuration a model is trained for 100 epochs, however an epoch during hyperparameter optimization is set to use only 500 samples. Afterwards the sample-wise F1 scores are calculated using all the validation samples. The mean of these sample-wise F1 scores is used as metric that the TPE algorithm has to maximize. F1 scores are calculated using the bounding boxes with an intersection over union (IoU) threshold of 0.5.

3.2 Pre-training

In order to perform transfer learning, the models are pre-trained using the SUN RGB-D dataset and the optimized hyperparameters. One model per approach is trained for 50 epochs. One epoch during training using the SUN RGB-D dataset consists of 2643 mini-batches. After 25 epochs learning rate decay reduces the learning rate from 0.005 to 0.001. The Region Proposal Network of the Mask R-CNN uses 15 anchor boxes that are defined with scales $s \in \{16, 32, 64, 128, 256\}$ and aspect ratios $ar \in \{0.5, 1, 2\}$. The approach RGBD-F uses a dropout rate of 0.3.

Stochastic Gradient Descent (SGD) [Rob51a] and Adaptive Moment Estimation (ADAM) [Kin14a] are optimizers commonly used to train convolutional neural networks. We used ADAM and SGD as choices in the hyperparameter optimization. Although ADAM optimizer produced the best results during the hyperparameter optimization, SGD with mini-batch and momentum is used to train the models, since training with the ADAM optimizer performs worse on the validation samples. The ADAM optimizer initially has a lower validation loss than the SGD optimizer. But the validation loss of the ADAM optimizer stagnates over the course of the training, whereas the validation loss of the SGD optimizer drops, while both training losses converge towards a minimum, indicating overfitting.

3.3 Transfer Learning and Training

The Elevator RGB-D dataset is only a tenth in size compared to the SUN RGB-D dataset. Therefore, transfer learning is applied, so that the pre-trained weights can be reused and the models trained on the Elevator RGB-D dataset converge towards a minimum faster and produce better results.

The model weights learned during the pre-training are reused to initialize the models for the training. For each approach a model is trained for another 50 epochs using the SGD optimizer. An epoch during training using the Elevator RGB-D dataset consists of 399 mini-batches. Learning rate decay reduces the learning rate from 0.005 to 0.0001 after 25 epochs.

3.4 Evaluation

The four approaches are evaluated after the training by calculating the mean average precision, the mean recall and the mean F1 score. The mean average precision (mAP) is calculated as described by Padilla et al. [Pad20a]. Similar to the mAP, the mean average recall (mAR) is calculated by averaging the average of the recall for each class for each sample. The mean F1 score is calculated by averaging the sample-wise F1 scores. The sample-wise F1 score is calculated using the average precision and the average recall of a sample. Additionally a precision-recall curve is calculated for each approach as described by Padilla et al. [Pad20a]. To derive these metrics the bounding boxes generated by the Mask R-CNN, the ground-truth bounding boxes, an IoU threshold of 0.5 and the test samples of the Elevator RGB-D dataset are used.

3.5 Implementation Details

To annotate the Elevator RGB-D dataset with labels the Label Studio software by Tkachenko et al. [Tka20a] is used. OpenCV [Bradski00a] is used to implement

	RoIs	DMC	Backbone
RGB	100	0.6	ResNet50 - BS2
D3	200	0.7	ResNet50 - BS2
RGBD	50	0.8	ResNet50 - BS2
RGBD-F	50	0.8	[32,32,64,128,256]

Table 1: Optimal configurations for each of the four approaches all utilizing ADAM optimizer.

Input Resolution	PTE	mAP	mAR	F1 score
256×256	50	0.031	0.272	0.129
512×512	50	0.131	0.219	0.268

Table 2: A comparison of models of the approach RGB using input images of different resolution.

Approach	mAP	mAR	F1 score
RGB	0.131	0.219	0.268
D3	0.155	0.241	0.312
RGBD	0.126	**0.271**	0.320
RGBD-F	**0.156**	0.258	**0.331**

Table 3: Mean average precision (mAP), their mean average recall (mAR) and F1 score for the approaches after pre-training.

Pre-trained on	none	SUN RGB-D	MS COCO
Epochs	0	50	320
ResNet layers	50	50	101
mAP	0.661	0.753	**0.862**
mAR	0.569	0.560	**0.585**
F1 score	0.535	0.579	**0.636**

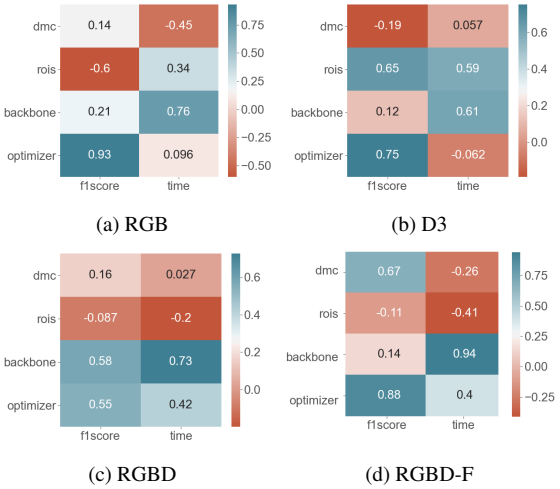
Table 4: A comparison of models of the approach RGB using transfer learning and without transfer learning.

the preprocessing steps. The Matterport Mask R-CNN implementation by Abdulla [Abd17a] is used to train and evaluate a Mask R-CNN model. The Matterport Mask R-CNN is altered to be compatible with TensorFlow version 2.0 made by Abadi et al. [Aba15a]. To perform the hyperparameter optimization using Tree-structured Parzen Estimators, *hyperopt* by Bergstra et al. [Ber13a] is used. Models are trained on a single *NVIDIA GeForce RTX 2070* with 8GB of memory.

4 RESULTS

Table 1 shows the optimal hyperparameters for each approach. In this table each configuration consists of the maximum number of regions per image (RoIs), the minimum detection confidence (DMC), the backbone architecture and the optimizer. The correlations between the hyperparameters, as well as the F1 score and the runtime of each evaluation run (time) for each approach can be seen in Figure 4. These correlations are calculated using Pearson's R [Pear96a].

Since the mini-batch size is limited by the GPU memory, only a mini-batch size of 2 is possible with 512×512 sized input images. As seen in Table 2 reducing the size of the images to 256×256 to increase the size



(a) RGB (b) D3

(c) RGBD (d) RGBD-F

Figure 4: The correlations between the hyperparameters for the four approaches.

of the mini-batches to 8 worsens the results, at least for the approach RGB. Table 2 shows models of the approach RGB using input images of different resolution, with the number of epochs that they were pre-trained (PTE), their mean average precision (mAP), their mean average recall (mAR) and their F1 score after 50 epochs of training using the SUN RGB-D dataset. The scores mAP, mAR and F1 are calculated using the test data of the SUN RGB-D dataset.

The approaches that use RGB-D images have better F1 scores than the approaches that solely rely on the RGB image and the depth image. The depth images of the SUN RGB-D dataset do not contain as many reflections and noise as the depth images of the Elevator RGB-D dataset. Therefore, one cannot conclude that the approaches RGBD and RGBD-F perform similarly well on the Elevator RGB-D dataset, see Table 3.

Using transfer learning to initialize the models with weights pre-trained with the SUN RGB-D dataset leads to better results, as seen in Table 4. This table shows models with and without pre-trained weights the dataset that they were pre-trained on, the number of epochs that they were pre-trained, their mean average precision (mAP), their mean average recall (mAR) and their F1 score after 50 epochs of training using the Elevator RGB-D dataset. mAP, mAR and F1 score are calculated using the test data of the Elevator RGB-D dataset. As seen in this table using transfer learning with weights pre-trained on the MS COCO dataset, provided by Abdulla [Abd17a], yields the best results after training for additional 50 epochs, although, this validation is only done for the approach RGB.

The RGB-D images utilizing a FuseNet-FPN backbone network perform best on the Elevator RGB-D dataset with a mean average precision of 0.794, a mean average recall of 0.565 and an mean F1 score of 0.599, see Table 5. Whereas relying on solely the RGB images or

Approach	mAP	mAR	F1 Score
RGB	0.753	0.560	0.579
D3	0.707	0.558	0.569
RGBD	0.768	0.558	0.578
RGBD-F	**0.794**	**0.565**	**0.599**

Table 5: Approaches evaluated w.r.t. their mAP, mAR and their mean F1 score.

the RGB images in combination with the depth image but with a ResNet50-FPN backbone network leads to similar results. Both are worse than the results of the approach RGBD-F. The approach that solely uses the depth images achieves the worst results. The precision-recall curves, as seen in Figure 5, supports this observation.

Thus, one can conclude that using RGB and RGB-D data for instance segmentation with Mask R-CNN in an elevator environment leads to equal results, whereas relying solely on depth images achieves worse results. Therefore, depth information does not improve the performance of instance segmentation with Mask R-CNN in highly reflective and noisy elevator environments. Furthermore, the usage of a FuseNet-FPN as a backbone network improves the resulting model in a highly reflective elevator environment. Figure 6 shows images segmented with these models.

5 CONCLUSION AND OUTLOOK

In this paper a comparison is done on whether instance segmentation using the Mask R-CNN algorithm can be improved by the usage of depth images. The images are recorded in highly noisy and reflective elevator environments. To perform this comparison four different approaches are compared. The four approaches use:

1. RGB image with a ResNet-FPN backbone
2. depth image duplicated on three channels with a ResNet-FPN backbone network
3. RGB-D image with a ResNet-FPN backbone

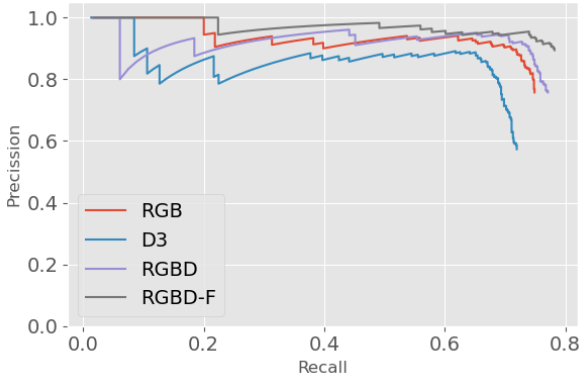


Figure 5: Precision recall curves of the four approaches.

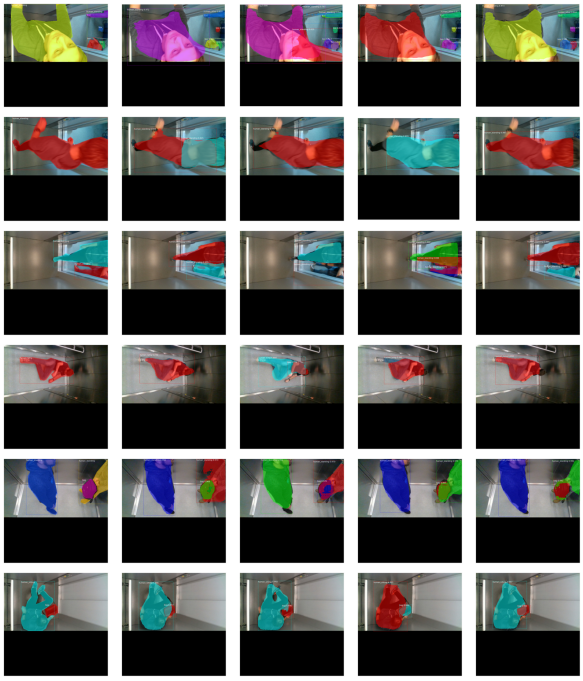


Figure 6: Samples of the Elevator RGB-D dataset instance-wise segmented with the pre-trained models. From left to right: ground truth, RGB, D3, RGBD, RGBD-F. The colors correspond to instances.

4. RGB-D image with an adapted FuseNet-FPN backbone network.

For each of these four approaches first the hyperparameters are optimized, pre-training is performed using the SUN RGB-D dataset, which contains images from indoor scenes and using transfer learning the pre-trained weights are further trained using images from an elevator environment. This comparison is done to answer the following research questions:

- *Can depth information be used to improve the performance of instance segmentation in highly reflective elevator environments which generally show a bad signal-to-noise ratio (SNR)?*

  The approach using an RGB image produces comparable results to the approach using RGB-D images, when training the Mask R-CNN models using a ResNet50-FPN as backbone network with a difference in the F1 score of 0.001. Solely relying on the depth image produces worse results than these two approaches.

- *Does incorporation of depth data generally increase the performance of instance segmentation?*

  After the pre-training using images from indoor scenes, the approach using the RGB image leads to the worst F1 score of all four approaches. The approach using the depth image in addition to

the RGB image results in a better F1 score. The approach using solely the depth image leads to a F1 score between these two approaches.

- *Can the use of a FuseNet-FPN as a backbone network together with transfer learning and hyperparameter optimization improve the performance of Mask R-CNN adapting to RGB-D images?*

  The approach using a FuseNet-FPN as a backbone network and RGB-D images results in a better F1 score than the approach using a ResNet50-FPN and RGB-D images using the images from an elevator environment as well as images from an indoor scene.

5.1 Outlook

Hyperparameter optimization using an SMBO algorithm can lead to configurations that tend to increase overfitting, since only one score metric is optimized. To improve the hyperparameter optimization, the evaluation runs during hyperparameter optimization can be performed for as long as the pre-training. This negates the necessity of the pre-training, since each iteration of the SMBO algorithm produces the pre-trained weights. Although this will also increase the time needed for the hyperparameter optimization by the increase in the total number of learned training data.

This problem that occurred during the hyperparameter optimization lead to the switch from the ADAM optimizer to the SGD optimizer, since the SGD optimizer is less likely to converge towards a local minimum [Wu16a, Kes17a, Aki17a]. Alternatively, this problem can be avoided by switching from ADAM to SGD during training [Wu16a, Kes17a] or using an optimizer with this switching behavior [Luo19a].

Since the mini-batch size is limited by the GPU memory only a mini-batch size of 2 is possible with 512×512 sized input images. The training process thus can be further improved by using multiple GPUs in parallel. He et al. used 8 GPUs in parallel to train the Mask R-CNN on the MS COCO dataset [He17a]. Loshchilov et al. use 30 GPUs in parallel to perform hyperparameter optimization of CNNs [Los16a].

The models are each only trained for 50 epochs to provide a fair comparison between them. They can be improved by increasing the number of epochs and use early-stopping if overfitting occurs. Additionally, pre-training can also be extended, which as shown in table 4 can improve the performance of the resulting model.

6 ACKNOWLEDGMENTS

Our thanks to the Austrian Research Promotion Agency FFG for facilitating project EDEN with basic funding program number 875284, fed by research budget of the Federal Republic of Austria.

7 REFERENCES

[Aba15a] Abadi, Martin et al. TensorFlow: Large-scale machine learning on heterogeneous systems, Software available from tensorflow.org, 2015.

[Abd17a] Abdulla, Waleed. Mask R-CNN for object detection and instance segmentation on Keras and Tensorflow. https://github.com/matterport/Mask_RCNN.

[Ahm11a] Ahmed, Mohamed, and Pitie, F.. Reflection detection in image sequences. In Proc. of the IEEE Comp. So. Conf. on Computer Vision and Pattern Recognition, pages 705-712, 2011.

[Aki17a] Akiba, Takuya, Suzuki, Shuji, and Fukuda, Keisuke. Extremely large minibatch SGD: training Resnet-50 on Imagenet in 15 minutes. CoRR, abs/1711.04325, 2017.

[All18a] Alldieck, Thiemo, Magnor, Marcus A., Xu, Weipeng, Theobalt, Christian, and Pons-Moll, Gerard. Video based reconstruction of 3d people models. CoRR, abs/1803.04758, 2018.

[Bar97a] Barrett, William A., and Mortensen, Eric N. Interactive livewire boundary extraction. Medical Image Analysis, 1(4):331-341, 1997.

[Ber11a] Bergstra, James, Bardenet, Remi, Bengio, Yoshua, and Kegl, Balazs. Algorithms for hyper-parameter optimization. In Proc. of the 24th Int. Conf. on Neural Information Processing Systems, NIPS'11, page 2546-2554, 2011.

[Ber13a] Bergstra, James, Yamins, D., and Cox, D.D.. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proc. of the 30th International Conf. on Machine Learning - Volume 28, ICML'13. JMLR.org, 2013.

[Beu79a] Beucher, Serge, and Lantuejoul, Christian. Use of watersheds in contour detection. In Int. Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, 132, 1979.

[Bo13a] Bo, Liefeng, Ren, Xiaofeng, and Fox, Dieter. Unsupervised Feature Learning for RGB-D Based Object Recognition, pages 387-402. Springer International Publishing, Heidelberg, 2013.

[Bradski00a] Bradski, G.. The opencv library. Dr. Dobb's Journal of Software Tools, 2000.

[Che16a] Chen, Tianqi, and Guestrin, Carlos. Xgboost: A scalable tree boosting system. In Proc. of the 22nd ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining, pages 785-794, 2016.

[Che19a] Chen, Xinlei, Girshick, Ross B., He, Kaiming, and Dollar, Piotr. Tensormask: A foundation for dense object segmentation. CoRR, abs/1903.12174, 2019.

[Chi18a] Chi, Zhixiang, Wu, Xiaolin, Shu, Xiao, and Gu, Jinjin. Single image reflection removal using deep encoder-decoder network. CoRR, abs/1802.00094, 2018.

[Coo01a] Cootes, T.F., Edwards, G.J., and Taylor, Christopher. Active appearance models. IEEE Trans. on Pattern Analysis and Machine Intelligence, 23:681-685, 2001.

[Cor95a] Cortes, Corinna, and Vapnik, Vladimir. Support-vector networks. Mach. Learn., 20(3):273-297, 1995.

[Cou13a] Couprie, Camille, Farabet, Clement, Najman, Laurent, and LeCun, Yann. Indoor semantic segmentation using depth information, 2013.

[Dai16a] Dai, Jifeng, Li, Yi, He, Kaiming, and Sun, Jian. R-FCN: Object detection via region-based fully convolutional networks, in CoRR, 2016.

[Gin02a] Ginneken, B. van, Frangi, A. F., Staal, J. J., ter Haar Romeny, B. M., and Viergever, M. A.. Active shape model segmentation with optimal features. IEEE Trans. on Medical Imaging, 21(8):924-933, 2002.

[Gom07a] Gomez, Octavio, Gonzalez, Jesus A., and Morales, Eduardo F.. Image segmentation using automatic seeded region growing and instance-based learning. In: Progress in Pattern Recognition, Image Analysis and Applications, pages 192-201, Springer, Berlin Heidelberg, 2007

[Gon01a] Gonzalez, Rafael C., and Woods, Richard E.. Digital Image Processing. Addison-Wesley Longman Publishing, USA, 2nd ed., 2001.

[Gra13a] Graves, Alex. Generating sequences with recurrent neural networks. CoRR, abs/1308.0850, 2013.

[Haz16a] Hazirbas, Caner, Ma, Lingni, Domokos, Csaba, and Cremers, Daniel. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In ACCV, 2016.

[He16a] He, K., Zhang, X., Ren, S., and Sun, J.. Deep residual learning for image recognition. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pages 770-778, 2016.

[He17a] He, Kaiming, Gkioxari, Georgia, Dollar, P., and Girshick, Ross B.. Mask r-cnn. IEEE International Conf. on Computer Vision (ICCV), pages 2980-2988, 2017.

[Ho95a] Ho, Tin Kam. Random decision forests. In Proc. of the 3rd Int. Conf. on Document Analysis and Recognition, volume 1 of ICDAR '95, page 278, USA, IEEE Computer Society, 1995.

[Hoc97a] Hochreiter, Sepp, and Schmidhuber, Jurgen. Long short-term memory. Neural Comput., 9(8):1735-1780, Nov. 1997.

[Hou19a] Hou, Rui, Chen, Chen, Sukthankar, Rahul, and Shah, Mubarak. An efficient 3d CNN for action/object segmentation in video. CoRR, abs/1907.08895, 2019.

[Ign99a] ignace20192019 Jordens. State classification of elevator doors to assist emergency detection in elevator networks, 2019.

[Kes17a] Keselman, L., Woodfill, J. I., Grunnet-Jepsen, A., and Bhowmik, A.. Intel(r) realsense(tm) stereoscopic depth cameras. In IEEE Conf. on Computer Vision and Pattern Recognition Workshops, pages 1267-1276, 2017.

[Kes17a] Keskar, Nitish Shirish, and Socher, Richard. Improving generalization performance by switching from adam to SGD. CoRR, abs/1712.07628, 2017.

[Kie52a] Kiefer, J., and Wolfowitz, J.. Stochastic estimation of the maximum of a regression function. Ann. Math. Statist., 23(3):462-466, 09 1952.

[Kil92a] Kilger, M.. A shadow handler in a video-based real-time traffic monitoring system. In Proc. IEEE Workshop on Applications of Computer Vision, pages 11-18, 1992.

[Kin14a] Kingma, Diederik, and Ba, Jimmy. Adam: A method for stochastic optimization. International Conf. on Learning Representations, 12 2014.

[Li16a] Li, Yi, Qi, Haozhi, Dai, Jifeng, Ji, Xiangyang, and Wei, Yichen. Fully convolutional instance-aware semantic segmentation, 2016.

[Lin14a] Lin, Tsung-Yi, Dollar, Piotr, Girshick, Ross B., He, Kaiming, Hariharan, Bharath, and Belongie, Serge J.. Feature pyramid networks for object detection. CoRR, abs/1612.03144, 2016.

[Lin16a] Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S.. Feature pyramid networks for object detection. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pages 936-944, 2017.

[Lin17a] Lin, Tsung-Yi, Goyal, Priya, Girshick, Ross B., He, Kaiming, and Dollar, Piotr. Focal loss for dense object detection. CoRR, abs/1708.02002, 2017.

[Lin17b] Lin, Tsung-Yi, Maire, Michael, Belongie, Serge J., Bourdev, Lubomir D., Girshick, Ross B., Hays, James, Perona, Pietro, Ramanan, Deva, Dollar, Piotr, and Zitnick, C. Lawrence. Microsoft COCO: common objects in context. CoRR, abs/1405.0312, 2014.

[Liu15a] Liu, Wei, Anguelov, Dragomir, Erhan, Dumitru, Szegedy, Christian, Reed, Scott E., Fu, Cheng-Yang, and Berg, Alexander C.. SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015.

[Los16a] Loshchilov, Ilya, and Hutter, Frank. Cma-es for hyperparameter optimization of deep neural networks. In Conference Track Proceedings, 4th Int. Conf. on Learning Representations, 2016.

[Luo19a] Luo, Liangchen, Xiong, Yuanhao, Liu, Yan, and Sun, Xu. Adaptive gradient methods with dynamic bound of learning rate. In Proc. of the 7th International Conf. on Learning Representations, New Orleans, Louisiana, 2019.

[Mos18a] Moser, Herbert, and Rebel, Johann. Leitfaden fuer personenaufzuege und personenhebeeinrichtungen. Technical report, MA 34 Bau -und Gebaeudemanagement, Stadt Wien, 2018.

[Nat06a] Nationalrat Oesterreich. Bundesgesetz ueber die Gleichstellung von Menschen mit Behinderungen (Bundesbehindertengleichstellungsgesetz - bgstg), 2006.

[Pad20a] Padilla, R., Netto, S. L., and da Silva, E. A. B.. A survey on performance metrics for object-detection algorithms. In International Conf. on Systems, Signals and Image Processing (IWSSIP), pages 237-242, 2020.

[Pat15a] Patro, S. Gopal, and Sahu, Kishore Kumar. Normalization: A preprocessing stage. IARJSET, 03 2015.

[Pear96a] Pearson, Karl. Mathematical contributions to the theory of evolution. Philos Trans R Soc Lond Series A, 187:253-318, 1896.

[Red15a] Redmon, Joseph, Divvala, Santosh Kumar, Girshick, Ross B., and Farhadi, Ali. You only look once: Unified, real-time object detection. CoRR, abs/1506.02640, 2015.

[Ren15a] Ren, Shaoqing, He, Kaiming, Girshick, Ross B., and Sun, Jian. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR, abs/1506.01497, 2015.

[Rob51a] Robbins, Herbert, and Monro, Sutton. A stochastic approximation method. Ann. Math. Statist., 22(3):400-407, 09 1951.

[Ron15a] Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation. CoRR, abs/1505.04597, 2015.

[Rot04a] Rother, Carsten, Kolmogorov, Vladimir, and Blake, Andrew. "grabcut": Interactive foreground extraction using iterated graph cuts. In ACM SIGGRAPH 2004 Papers, SIGGRAPH '04, page 309-314, New York, NY, USA, 2004.

[Rum87a] Rumelhart, D. E., and McClelland, J. L.. Learning Internal Representations by Error Propagation, pages 318-362. MIT Press, 1987.

[Sab18a] Sabir, Ekraam, Rawls, Stephen, and Natarajan, Prem. Implicit language model in LSTM for OCR. CoRR, abs/1805.09441, 2018.

[Son15a] Song, S., Lichtenberg, S. P., and Xiao, J.. Sun rgb-d: A rgbd scene understanding benchmark suite. In IEEE Conf. on Computer Vision and Pattern Recognition, pages 567-576, 2015.

[Sti99a] Stigler, Daniel. Evaluation of classifiers for use in emergency detection systems, bachelors thesis, University of Applied Sciences Upper Austria, 2018.

[Tan15a] Tang, M., Ayed, I. B., Marin, D., and Boykov, Y.. Secrets of grabcut and kernel k-means. In IEEE International Conf. on Computer Vision (ICCV), pages 1555-1563, 2015.

[Tka20a] Tkachenko, Maxim, Malyuk, Mikhail, Shevchenko, Nikita, Holmanyuk, Andrey, and Liubimov, Nikolai. Label Studio: Data labeling software, Open source software available from https://github.com/heartexlabs/label-studio, 2020.

[Tom15a] Tommasi, Tatiana, Patricia, Novi, Caputo, Barbara, and Tuytelaars, Tinne. A Deeper Look at Dataset Bias, pages 37-55. Springer, 2017.

[Tor11a] Torralba, A. and Efros, A. A.. Unbiased look at dataset bias. In CVPR, pp. 1521-1528, 2011.

[Uij13a] Uijlings, Jasper, Sande, K., Gevers, T., and Smeulders, Arnold. Selective search for object recognition. International Journal of Computer Vision, 104:154-171, 09 2013.

[Wei16a] Weiss, Karl, Khoshgoftaar, Taghi, and Wang, DingDing. A survey of transfer learning. Journal of Big Data, 3, 12 2016.

[Wu10a] Wu, Mingjun, and Peng, Xianrong. Spatio-temporal context for codebook-based dynamic background subtraction. AEU - International Journal of Electronics and Communications, 64(8):739-747, 2010.

[Wu16a] Wu, Yonghui et al.. Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR, abs/1609.08144, 2016.

[Xia16a] Xiang, Yu, Kim, Wonhui, Chen, Wei, Ji, Jingwei, Choy, Christopher, Su, Hao, Mottaghi, Roozbeh, Guibas, Leonidas, and Savarese, Silvio. Objectnet3d: A large scale database for 3d object recognition. In ECCV 2016, pages 160-176, Springer Int. Publishing, 2016.

[Ye17a] Ye, L., Liu, Z., and Wang, Y.. Depth-aware object instance segmentation. In IEEE Int. Conf. on Image Processing (ICIP), pages 325-329, 2017.

[Zwe20a] Zwettler, Gerald Adam, Holmes III, David R., and Backfrieder, Werner. Strategies for training deep learning models in medical domains with small reference datasets. Journal of WSCG, 28 (1-2):37-46, 2020.

Contactless heart rate estimation from facial video using skin detection and multi-resolution analysis

Khawla BENSALAH
National Engineering School of Sfax
Research Lab: Math, Earth Sciences, Modeling and Intelligent Systems, Faculty of Sciences of Gafsa
University of Sfax, BP 1173, Sfax, Tunisia
khawlabensalah8@gmail.com

Mohamed OTHMANI
Faculty of Sciences of Gafsa, University of Gafsa
BP 2100,Gafsa, Tunisia
mohamed.othmani@yahoo.fr

Monji KHERALLAH
Faculty of Sciences of Sfax, University of Sfax
BP 1173,Sfax, Tunisia
monji.kherallah@fss.usf.tn

ABSTRACT

This paper introduces a remote Photoplethysmography (rPPG), which is used to estimate human heart rate without any physical contact, has been extensively applied in multiple fields like medical diagnosis, analysis of human emotions, rehabilitation training programs, biometric, and fitness assessments. The rPPG signals are usually extracted from facial videos. However, it is still a challenging task due to several contributing factors, e.g., variation in skin tone, lighting condition, and subject's motion. Accordingly, in this work, a novel approach based on deep learning skin detection method and the discrete wavelet transform (DWT) is employed to precisely estimate heart rate from facial videos. In the proposed method, by implementing the DWT, the signal is decomposed into approximations and details parts thereby it helps in analyzing it at different frequency bands with different resolutions. The results derived from the experiments show that our proposed method outperforms the state-of-the-art methods on the UBFC-RPPG database.

Keywords

rPPG, Heart rate estimation, facial video, Deep learning, Discrete wavelet transform, skin detection.

1 INTRODUCTION

In the recent years, remote heart rate estimation from facial videos (rPPG) has been increasingly investigated. Remote photoplethysmography (rPPG) traces the blood volume pulse remotely by tracking changes in the skin reflectance during the cardiac cycle by the means of a digital camera, although these color changes are not noticeable to the necked human eye. This color variation helps to determine the required information to measure the heart rate. Heart Rate(HR) estimation is an important component to determine the health state and the well-being of an individual and it is usually evaluated as normal if it is between 60 to 100 beats per minute (bpm) [1]. (HR) analysis has always been an area of interest not only for the medical applications but also for other applications from a variety of other fields such as include telemetry supervising of elderly people/newborn babies [2], the athletes involved in sports activities [3], emotional recognition [4], human authentication [5], etc. Traditionally, we assure heart rate estimation with equipment, such as an electrocardiogram (ECG) or conventional photoplethysmography (PPG), however, it requires electrodes/sensors attached to the skin surface, gel, and experienced nurses or doctors, and usually causes skin discomfort. Hence, rPPG has gained more and more interest because of its several advantages as comfort, non-complicated, non-invasive and inexpensive characteristics. Additionally, in some cases, such as burns, severe infections, etc the employ of a sensor is not always possible. Despite the diversity of methods, the main challenge consists in performing robustness to variation in skin tone, natural motion and illumination variation that results in undesirable noise and artifacts in the extracted heart rate. In this paper, we present a novel approach to achieve heart rate detection (HR) from facial videos. Our approach consists of 6 following steps: the first step consists of performing face alignment and detecting the face with viola

and jones algorithms. Since the PPG information is more accurate in some part of the face like the cheek and the forehead, we choose to focus the ROI on the forehead. Accordingly, the second step was in detecting the forehead in all the frames. In Step3, we apply our algorithm of skin detection based on convolutional neural networks which is detailed in [6]. In Step 4, the rPPG waveform is obtained from the mean of the sequence green channeled video from the skin-region. In the fifth step, the discrete wavelet transform (DWT) is employed to precisely estimate heart rate from facial videos. A novel spatiotemporal representation is introduced by implementing the DWT, the signal is decomposed into approximations and details parts thereby it helps in analyzing it at different frequency bands with different resolutions. In the final step, we estimate the heart rate by identifying the maximal frequency using spectral representation.

2 RELATED WORK

Remote heart rate measurement based-video was the first attempt by Verkruysse et al.[7]. Poh et al.[8] used Blind Source Separation (BSS) technique to extract PPG signal, which yields a dynamic combination of color channels. Independent component analysis (ICA) was applied to average spatially R, G, and B signals of the facial image sequence. After that, the authors extend their work in [9] to refine the PPG signal by invoking a series of temporal filtering. The extracted PPG signal was more refined further by bandpass filtering. Bandpass filters cover a predefined range of expected HRs at the operational frequency of 0.75 to 4.5 Hz [10]. A number of other methods such as the joint approximate diagonalization of eigenmatrices (JADE)[7] or FastICA [11], and extensions like joint blind source separation (JBSS)[12], Spatio-temporal ICA [13], constrained ICA [14], radical ICA [15], robust ICA [16], Project ICA [17], and Principal Component Analysis (PCA) [11] have been used to (rPPG) signal. Another group of methods proposed a set of model-based methods regarding color vectors such as the chrominance model (CHROM) [18], blood volume pulse signature (PBV) model, spatial subspace rotation (S2R), and a plane orthogonal to the skin (POS) model [19], POS is conceived to extract the heart signal based on a skin reflection model in normalized RGB space. As other researchers [20] have found, the selection of ROI has a major impact on the quality of the (rPPG) signal. Haque and al. [21] have focused on extracting facial landmark points by combining the trajectories of the supervised descent method (SDM) and generating stable trajectories (GFT) method for heart rate estimation. Other recent works have applied deep learning to extract heart rates directly from camera images. They rely on the strength of deep networks to acquire knowledge of which areas in the image correspond to heart rate. HR-CNN [22] uses a two-step convolutional neural network. To estimate a heart rate from a sequence of facial images. DeepPhys [23] propose an end-to-end method to extract heart and breathing rate from videos using a deep convolutional network. RhythmNet [24] uses a spatial-temporal representation encoding the HR signals to form the CNN, and they use Gated Recurrent Unit (GRU) to consider the relationship of adjacent HR measurements from video sequences. AutoHR [25] employs neural architecture search (NAS) to discover temporal difference convolution (TDC) as a strong backbone to capture intrinsic rPPG signal from frames. Bousefsaf and al.[26] present a convolutional 3D networks, without any special frame processing, and a particular training procedure which involves only synthetic data is proposed. Hsu and al.[27] employed Short-Time Fourier Transform (STFT) to transform the signals to time-frequency representation, which is cascaded with Convolutional Neural Network (CNN). Lee and al.[28] introduce meta-learning framework for remote heart rate estimation which involves three modules: convolutional encoder, rPPG estimator and synthetic gradient generator. The first module was used to infer the rPPG signal but the two later were used during transductive learning.

3 METHODOLOGY

This section explains the overall methodology applied to estimate heart rate remotely using face videos; We follow a six-step process from face detection from video frames to heart rate estimation in this work. Fig.1 shows the flowchart of our proposed method, which is described in the following subsections.

3.1 Face Detection and face alignment

The first step of the proposed method is face alignment. Normally head it is exposed to different motion classified as internal such as (smiling/laughing, talking, screaming, eating, etc), and external related to different voluntary head motion. Thereby, we need to perform face alignment as seen in Fig.2, by using a simple method that focuses only on areas around the eyes. Three basic affine transformations are carried out: rotation, translation, and scaling. For eye detection, we have used OpenCV Haar cascade configurations (eye detection module). After that, we have calculated the center of the detected eyes and then drawing 2 lines the first connects the two central points of the eyes, the second is a horizontal line that forms an angle. Our aim is to rotate the image based on this angle. To calculate the angle, we first need to determine the length of the two legs of a right triangle. Then we can obtain the required angle using the following formulas:

$$\begin{cases} \Delta_x = x_{\{right\ eye\}} - x_{\{left\ eye\}} \\ \Delta_y = y_{\{right\ eye\}} - y_{\{left\ eye\}} \end{cases} \quad (1)$$

Figure 1: The flowchart of the proposed method.

$$\theta = \arctan\left(\frac{\Delta_x}{\Delta_y}\right) \tag{2}$$

we have specified in which direction our image will rotate. If the ordinate Y the left eye is bigger than the ordinate Y of the right eye, rotation is in the clockwise direction. Else, the rotation will be in the counterclockwise direction. After rotation, we did the same thing with all the other images that we are processing. Finally, we scale our images based on that ratio. The second step is face detection and cropping which is carried out for each frame. The detection and tracking have been implemented by using the Viola-Jones face detection method [31] ; This algorithm is extremely rapid and achieves high detection rates.

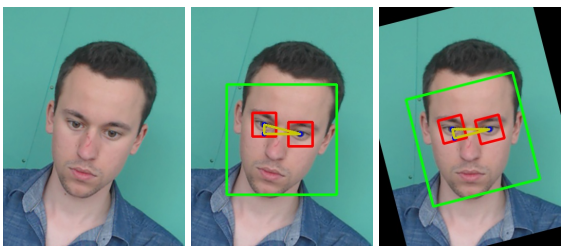


Figure 2: An example from UBFC-RPPG dataset illustrating face alignement step

3.2 Heart Rate Estimation by Forehead location

The chosen region for estimating heart rate is the forehead, the position of the forehead is determined by the coordinates of the two eyes. We have used OpenCV Haar cascade configurations (eye detection module) for eye detection. After determining the coordinate of the two eyes, we have searched for the center of both of them, the orthogonal on the center of the left eye represents the left limit of the forehead rectangle, and the orthogonal on the center of the right eye represents the right side of the forehead rectangle. The outer limit of the forehead is represented by the line of y-axis where y = 0. The lower limit of the forehead rectangle is determined by the line y = (height of the image/4).

3.3 Skin detection

Cardiac activity causes a subtle color variation only on the facial skin during the cardiac cycle. It is, therefore, crucial to observe only the pixels that belong to the skin while ignoring the others like background, clothes, teeth, hair, and other unrelated parts. Consequently, it is important to consider that skin detection is a very crucial step in remote heart rate estimation. It improves the quality of the PPG signal by increasing the Signal-to-Noise Ratio SNR. In a previous work, we suggested a novel approach based on a convolutional neural network (CNN) for skin detection[6]. To reduce false positives, our training strategy consists of patch-based train-

ing which is robust to background clutter and detects skin pixels precisely. We proposed an efficient algorithm to detect and extract skin pixels from the whole RGB image after integrating the trained model. Our purpose was in achieving a very precise human skin detection. Despite the fact that skin detection represents a challenging problem because it can vary dramatically in its appearance due to many factors such as illumination conditions, pose variations, race, aging, and complex background. The proposed approach overcomes most of the difficulties in skin detection. Table 1 illustrate the architecture of the proposed skin detection CNN model. The Processing algorithm overview is shown in Fig.3.

3.4 rPPG signal

The green channel the RGB space transcribes better the PPG signal rather than the red and blue channels, and it provides a good signal noise ratio SNR [7] because the green light penetrates human skin better than blue light and it is absorbed by hemoglobin better than red light [29]. After performing the skin detection algorithm, we were able to extract rPPG signal by calculating the mean green values of pixels inside each frame, which concludes a temporal raw signal T_s:

$$T_s = [m_1(g), m_2(g), m_3(g), ..., m_n(g)] \tag{3}$$

Where n denotes the index of frame and $m(g)$ correspond to the mean green value of pixels at each frame of the video.

$$m_i(g) = \left(\frac{\sum_{(x,y)\in ROI} I_{g(x,y)}}{|(x,y)|} \right) \tag{4}$$

Where $I_{g(x,y)}$ denotes the green channel intensity of each pixel, and $|(x,y)|$ represents the number of skin pixels for each frame. The source signals were passed by band-pass filtered (Hamming window filter with low- and high-frequency in the range (0.75Hz-4.5Hz) which corresponds to 45 bpm and 270 bpm respectively).

3.5 rPPG Signal Multilevel decomposition and heart rate estimation

Even though Short-Time Fourier Transform (STFT) is the most used tool for frequency analysis, but it is an inappropriate choice for non-stationary signals. Naturally, the physiological signals are non-stationary such as rPPG, STFT which provide a constant resolution at all frequencies and it loses time resolution in the spectral domain. Additionally, the fixed-length window is an inconvenient factor since the frequency behavior is not known, to overcome this shortcoming and to perform different resolutions analysis, we adopt the wavelet technique.

The basic formula of wavelet transform can be written as:

$$Wt(a,b) = \frac{1}{\sqrt{a}} \int_{(-\infty)}^{(+\infty)} X(t)\psi^*(a,b)dt \tag{5}$$

and the complex conjugate of the mother wavelet is defined as:

$$\psi^*(a,b) = \frac{1}{\sqrt{a}} \psi \left(\frac{t-b}{a} \right) \tag{6}$$

where X(t) is the processed signal, a is the scaling parameter and b is the translation parameter. ψ^* is the mother wavelet, which plays the role of both a window and basis function, and is then scaled or shifted by the parameter a to regulate window length. Narrow windows ease detecting the presence of high-frequency components and wide windows facilitate extracting low frequencies components. In this work, we employed Discrete wavelet transform (DWT) because it offers a highly efficient wavelet representation. In Discrete wavelet transform, scale is only modified in the power of base 2 (a = 1, 2, 4, 8,), and translation is also done in comparative relation to the scale. Furthermore, DWT is not shift-invariant. The DWT can be defined as:

$$Wt(j,k) = \frac{1}{\sqrt{2^j}} \int_{(-\infty)}^{(+\infty)} X(t)\psi^*(j,k)dt \tag{7}$$

For the discretized mother wavelet eq.6 Can be rewritten as:

$$\psi^*(j,k) = \frac{1}{\sqrt{2^j}} \psi \left(\frac{t-k2^j}{2^j} \right) \tag{8}$$

Where the scale and the shift parameters are commonly used as 2^j and $k2^j$. The signal is decomposed into approximation and detail coefficients in the first level. The decomposition for the remaining levels is realized using only the approximation coefficients. At each level, the decomposition is performed by using low-pass and high-pass filters. The output coefficients are named approximation coefficient and detail coefficient. These coefficients assure reconstruction of the original signal with a process called inverse discrete wavelet transform (IDWT). In this paper, the Coiflet wavelet [32] was chosen as the mother wavelet with a four levels decomposition is performed. As illustrated in the wavelet decomposition tree Fig.4. Coiflets were originally derived from the Daubechies wavelet. Coiflets wavelet family from 1^{st} to 5^{th} order, It uses many overlapping windows. It is based on six scaling and wavelet function coefficients, The Coiflet wavelet also follows the mirror technique. The signal decomposition was made with (coif1) and the reconstruction was made with the same wavelet (coif1) but we choose only the third and the fourth detailed coefficient (D3 and D4). The result is shown in Fig.5. The multi-resolution representation with the discrete wavelet transforms is illustrated

Table 1: The architecture of the proposed skin detection CNN model.

Layers No.	Type	Kernel size	No. Kernels	Stride	Outupt shape
Layer 1	Conv 1	1×1	16	1	$1 \times 1 \times 16$
Layer 2	Conv 2	2×2	32	1	$2 \times 2 \times 32$
Layer 3	Conv 3	1×1	64	1	$2 \times 2 \times 64$
Layer 4	Max pooling 1	2×2		0	$1 \times 1 \times 64$
Layer 5	Flatten 1				64
Layer 6	Dense 1		128		128
Layer 7	Dense 2		64		64
Layer 8	Dense 3		1		1

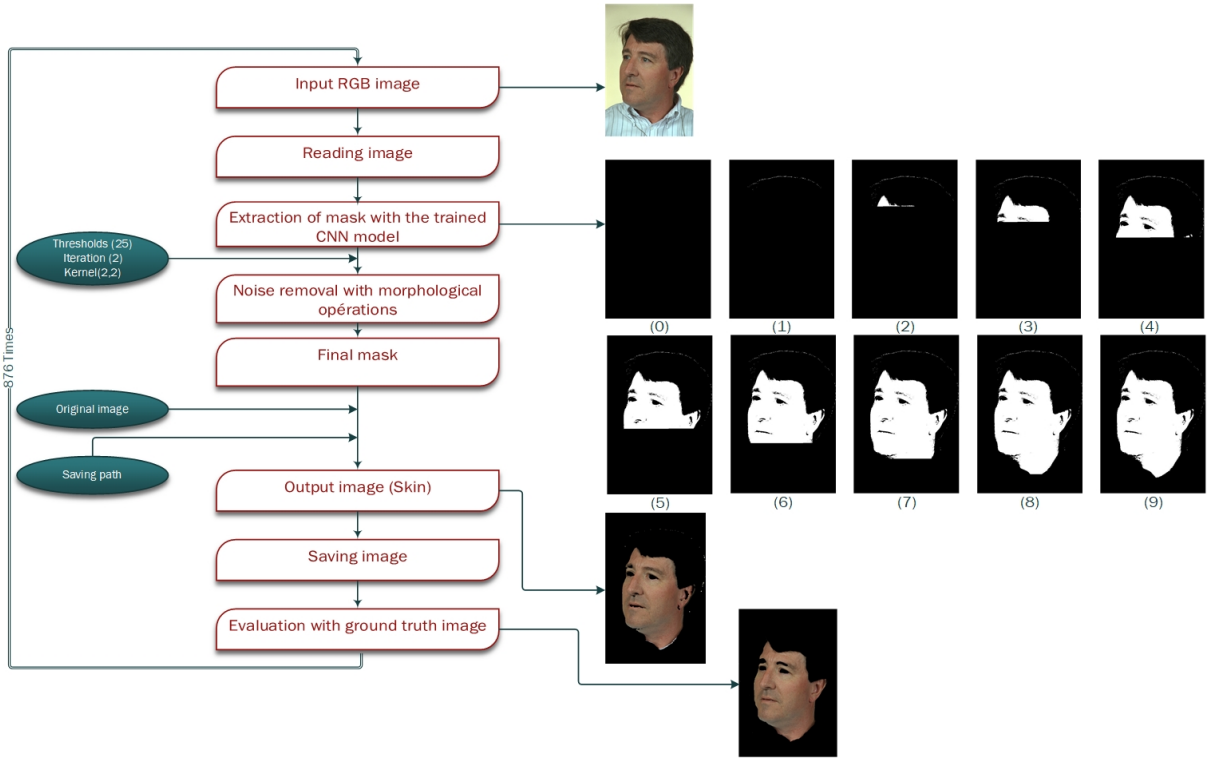


Figure 3: Framework of the proposed skin detection algorithm. The skin map is constructed progressively and displayed in steps 0-9. Each pixel of the image is predicted with the trained model.

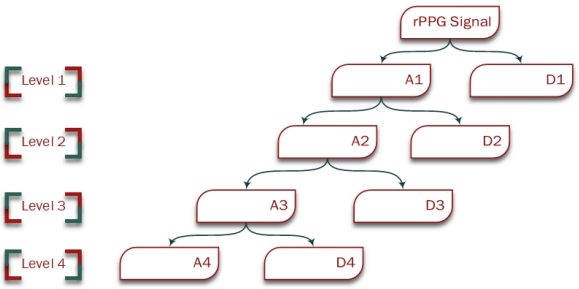


Figure 4: Discrete Wavelet Transform Decomposition Tree of four levels.

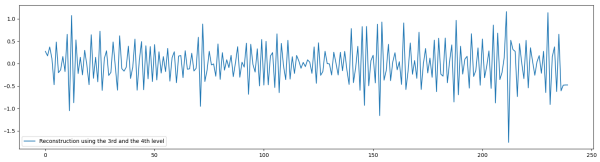


Figure 5: rPPG signal reconstruction with the third and the fourth detailed coefficient using discrete wavelet transform.

in spectrogram as seen in Fig.6. We can estimate heart rate with peak detection. Peaks can then be simply identified using a diversity of peak detection algorithms but most of them require the selection of an appropriate threshold value to avoid incorrect peak detection. Once the peaks are detected the distance between the peaks

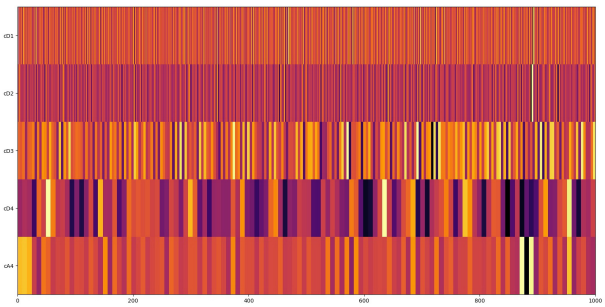


Figure 6: Spectrogram illustrating the four level decompositions.

can be used for the calculation of heart rate from the PPG signals using eq.9:

$$Heartbeatrate = 60 \times \frac{Fs}{intsys} \tag{9}$$

where Fs is the sampling rate and intsys corresponds to the distance between the systolic peaks. Otherwise, for this aim, using frequency analysis, the frequency equivalent to the index with the highest spectral power is selected as an estimate for the HR. In the Fig.7 we see that the maximum power of the first detailed level coefficient is at frequency bin 1.602. This, in turn, gives HR $1.602 \times 60 = 96.12\ bpm$.

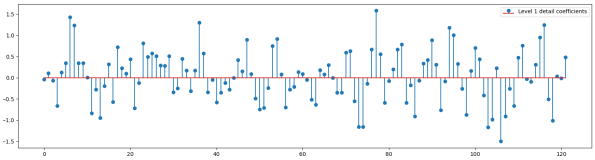


Figure 7: Spectral representation of detailed coefficient level 1

4 EXPERIMENTAL RESULTS

We conducted experiments to demonstrate the effectiveness of the proposed method on UBFC-rPPG signal datasets [30], which is publicly available, and is dedicated to rPPG algorithm evaluation. It is composed of 43 uncompressed videos, where each video is joined with Ground truth heart rate for evaluation. Each video have a duration of 60 seconds and corresponds to 30 frames per second (fps) with a resolution of 640 $\times$ 480 in 8-bits RGB format. Subjects recorded while playing a stressful game. It is worth noting that the rPPG signal is much noisier if we don't apply human skin detection and face alignment algorithm to fix the subject's voluntary movements. At the next stage, the decomposition-reconstruction operation is performed by using the wavelet transformation approach where coiflet wavelets gave a satisfying results. We generate different spectral representation at different level as shown in Fig. from 8 to 11. The

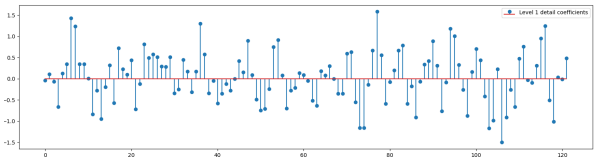


Figure 8: Spectral representation of detailed coefficient at level 1

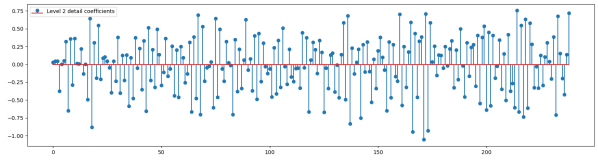


Figure 9: Spectral representation of detailed coefficient at level 2

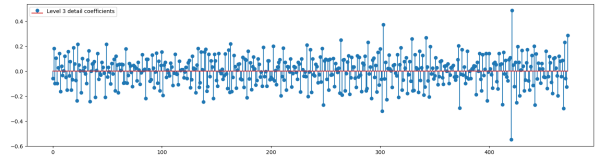


Figure 10: Spectral representation of detailed coefficient at level 3

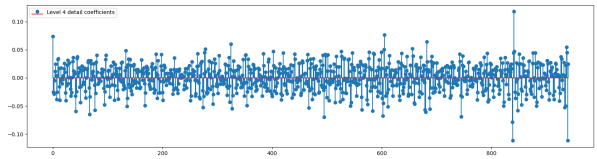


Figure 11: Spectral representation of detailed coefficient at level 4

performance of the proposed approach is compared to the state-of-the-art methods on the experimental dataset listed in Table 2. We have measured the accuracy in terms of Mean Absolute Error (MAE)and mean squared error (RMSE), our method generated the lower error, which relavate a high accuracy. We report results with two other metrics standard deviation of error (SD) and Pearson correlation coefficient (R), From the results we can determine that the proposed method showed consistent performance although the data acquisition scenarios (subject mouvement,closed eyes,luminosity variation). Lastly, the proposed Skindetection-DWT method was not tested yet on the most challenging use case scenarios that imply a lot of motion. This is mostly due to the fact that to the best of our knowledge there is no datasets for the motion robustness publicly available.

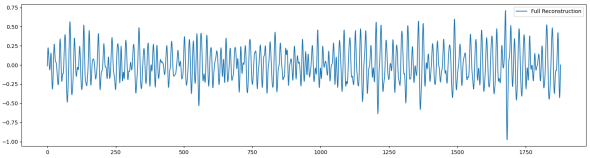


Figure 12: rPPG signal full reconstruction using all coefficients

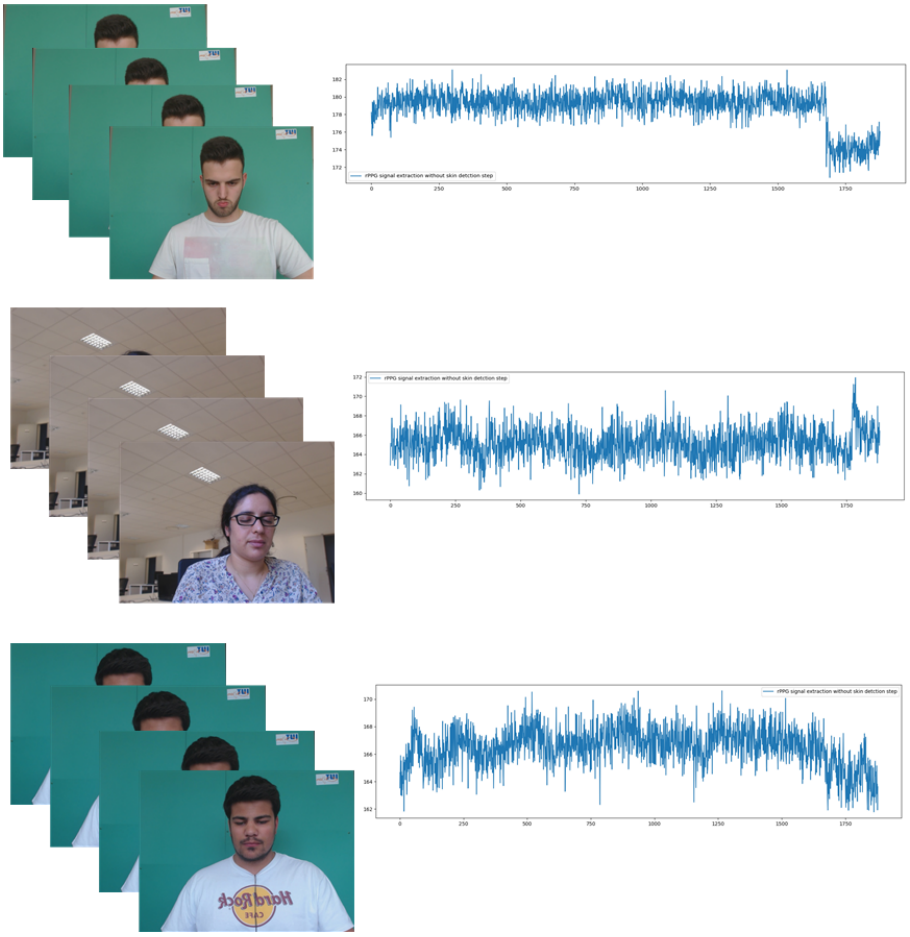


Figure 13: rPPG signals extraction without skin detction step

Table 2: Performance comparison between the proposed method and the state of the art methods of heart rate estimation on UBFC-rPPG dataset.

Method	HR(bpm)			
	SD	MAE	RMSE	R
GREEN[7]	20.2	10.2	20.6	-
ICA[8]	18.6	8.43	18.8	-
CHROM[18]	19.1	10.6	20.3	-
POS[19]	10.4	4.12	10.5	-
3D[26]	8.55	5.45	8.64	-
End-of-end(baseline)[28]	13.70	12.78	13.30	0.27
Meta-rPPG(inductive)[28]	14.17	13.23	14.63	0.35
Meta-rPPG(proto only)[28]	9.17	7.82	9.37	0.48
Meta-rPPG(synth only)[28]	11.92	9.11	11.55	0.42
Meta-rPPG(proto + synth)[28]	7.12	5.79	7.42	**0.53**
Ours	**5.32**	**3.24**	**5.64**	0.46

5 CONCLUSION

This paper discusses a novel approach for heart rate estimation using facial videos. The green channel yields the most prominent cardiac cycles and it was extracted from the forehead region. Since the most important problem associated with rPPG signal was variation in skin tone, lighting condition, and subject's motion, the presence of these three factors represents many interferences and noise on the resulting signal. Accordingly, a previously proposed approach to skin detection is applied which relies on the strength of the convolutional neural network, also the face alignment al-

gorithm improves the efficiency of the proposed signal, then we adopted the multi-resolution representation with the discrete wavelet transform using spectrogram, so it keeps equally the time and the frequency information, and its facilities signal analysis. The signal decomposition and reconstruction were made with coiflet wavelet, the reconstruction of the signal was made only with the two final detailed coefficients.
The choice was relying on the visualization of the relation between the detailed coefficients of the wavelet transform.
In the final step we evaluate our heart rate estimation with the state of art methods on a public dataset UBFC-RPPG.

6 REFERENCES

[1] G. Casalino, G. Castellano, V. Pasquadibisceglie, and G. Zaza: Contactless real-time monitoring of cardiovascular risk using video imaging and fuzzy inference rules.Information, vol. 10, no. 1, p. 9, 2019. 1.

[2] Chaichulee, S., Villarroel, M., Jorge, J. o., Arteta, C., McCormick, K., Zisserman, A., Tarassenko, L. (2019): Cardio-respiratory signal extraction from video camera data for continuous non-contact vital sign monitoring using deep learning. Physiological Measurement, 40(11). https://doi.org/10.1088/1361-6579/ab525c

[3] Kwon, S., Kwon, Y. T., Kim, Y. S., Lim, H. R., Mahmood, M., Yeo, W. H. (2020): Skin-conformal, soft material enabled bioelectronic system with minimized motion artifacts for reliable health and performance monitoring of athletes. Biosensors and Bioelectronics, 151(December 2019), 111981. https://doi.org/10.1016/j.bios.2019.111981

[4] Z. Tong, X. Chen, Z. He, K. Tong, Z. Fang and X. Wang :Emotion Recognition Based on Photoplethysmogram and Electroencephalogram. IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 2018, pp. 402-407, doi: 10.1109/COMPSAC.2018.10266.

[5] Sancho J, Alesanco a, Garcia J: Biometric Authentication Using the PPG: A Long-Term Feasibility Study. Sensors (Basel). 2018

[6] Salah, K.B., Othmani, M. Kherallah, M. A novel approach for human skin detection using convolutional neural network. Vis Comput (2021). https://doi.org/10.1007/s00371-021-02108-3

[7] Verkruysse W, Svaasand LO, Nelson JS: Remote plethysmographic imaging using ambient light. Optics Express, 2008, 16(26): 21434-21445

[8] Poh M-Z, McDuff DJ, Picard RW: Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. Optics Express, 2010, 18(10): 10762-10774

[9] M. Z. Poh, D. J. McDuff, and R.W. Picard: Advancements in noncontact, multiparameter physiological measurements using a webcam. IEEE Transactions on Biomedical Engineering, 58(1):7-11, 2011

[10] McDuff D, Gontarek S, Picard RW:Remote detection of photoplethysmographic systolic and diastolic peaks using a digital camera. IEEE Trans Biomed Eng 2014;61:2948-54.

[11] M. Lewandowska, J. Ruminski, T. Kocejko, and J. Nowak: Measuring pulse rate with a webcam a non-contact method for evaluating cardiac activity.federated conference on computer science and information systems (FedCSIS), pp. 405-410, IEEE, 2011.

[12] J. Cheng, X. Chen, L. Xu, Z.J. Wang :Illumination variation-resistant video-based heart rate measurement using joint blind source separation and ensemble empirical mode decomposition. IEEE Journal of Biomedical and Health Informatics, 2017, pp. 1422-1433

[13] Sun Y, Hu S, Azorin-Peris V, Greenwald S, Chambers J, Zhu Y: Motion-compensated noncontact imaging photoplethysmography to monitor cardiorespiratory status during exercise. J Biomed Opt 2011;16:77010

[14] Tsouri GR, Kyal S, Dianat S, Mestha LK: Constrained independent component analysis approach to nonobtrusive pulse rate measurements. J Biomed Opt 2012;17:077011.

[15] Holton BD, Mannapperuma K, Lesniewski PJ, Thomas JC: Signal recovery in imaging photoplethysmography. Physiol Meas 2013;34:1499-511

[16] Christinaki E, Giannakakis G, Chiarugi F, Pediaditis M, Iatraki G, Manousos D, et al: Comparison of blind source separation algorithms for optical heart rate monitoring., pp. 339-42, ICST, 2014.

[17] Qi, L., Yu, H., Xu, L., Mpanda, R. S., Greenwald, S. E. (2019): Robust heart-rate estimation from facial videos using Project ICA. Physiological Measurement, 40(8). https://doi.org/10.1088/1361-6579/ab2c9f.

[18] De Haan, G., Jeanne, V.: Robust pulse rate from chrominance-based rppg. IEEE Transactions on Biomedical Engineering 60(10),

2878-2886 (2013).

[19] X. Chen, J. Cheng, R. Song, Y. Liu, R. Ward, and Z. J. Wang: Video-based heart rate measurement: Recent advances and future prospects, IEEE Transactions on Instrumentation and Measurement, 2018.

[20] S. Ren, X. Cao, Y. Wei, and J. Sun :Face alignment at 3000 fps via regressing local binary features. In the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1685-1692, 2014.

[21] Haque, M. A., Irani, R., Nasrollahi, K., Moeslund, T. B. (2016):Heartbeat Rate Measurement from Facial Video. IEEE Intelligent Systems, 31(3), 40-48.

[22] R. Spetlik, J. Cech, V. Franc, and J. Matas: Visual heart rate estimation with convolutional neural network.08 2018

[23] W. Chen and D. McDuff: Deepphys: Video-based physiological measurement using convolutional attention networks, in Proceedings of the European Conference on Computer Vision (ECCV), pp. 349-365, 2018

[24] Niu, X., Shan, S., Han, H., Chen, X.: Rhythmnet: End-to-end heart rate estimation from face via spatial-temporal representation. IEEE Transactions on Image Processing (2019)

[25] Yu, Z.; Li, X.; Niu, X.; Shi, J.; Zhao, G. AutoHR: A Strong End-to-end Baseline for Remote Heart Rate Measurement with Neural Searching. arXiv 2020, arXiv:2004.12292

[26] Bousefsaf, F., Pruski, A., Maaoui, C. (2019): 3D convolutional neural networks for remote pulse rate measurement and mapping from facial video. Applied Sciences (Switzerland), 9(20), 1-21. https://doi.org/10.3390/app9204364

[27] Hsu, G. S. J., Xie, R. C., Ambikapathi, A. M., Chou, K. J. (2020): A deep learning framework for heart rate estimation from facial videos. Neurocomputing, 417, 155-166. https://doi.org/10.1016/j.neucom.2020.07.012

[28] Lee, E., Chen, E., Lee, C. Y. (2020): Meta-rPPG: Remote Heart Rate Estimation Using a Transductive Meta-learner. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12372 LNCS, 392-409. https://doi.org/10.1007/978-3-030-58583-9-24

[29] Lister, T., Wright, P. A., Chappell, P. H. (2012): Optical properties of human skin. Journal of biomedical optics, 17(9), 090901.

[30] Bobbia S, Macwan R, Benezeth Y, Mansouri A, Dubois J.: Unsupervised skin tissue segmentation for remote photoplethysmography. Pattern Recognit Lett 2017.

[31] P. Viola, Ms. Jones : Rapid object detection using a boosted cascade of simple features,Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.

[32] Shyh-Jier Huang and Cheng-Tao Hsieh :Coiflet wavelet transform applied to inspect power system disturbance-generated signals,in IEEE Transactions on Aerospace and Electronic Systems, vol. 38, no. 1, pp. 204-210, Jan. 2002, doi: 10.1109/7.993240.

Updating 3D Planning Data based on Detected Differences between Real and Planning Data of Building Interiors

Andreas Dietze
Fulda University of Applied Sciences
Leipziger Straße 123
36037 Fulda, Germany
andreas.dietze@cs.hs-fulda.de

Paul Grimm
Darmstadt University of Applied Sciences
Haardtring 100
64295 Darmstadt, Germany
paul.grimm@h-da.de

Yvonne Jung
Fulda University of Applied Sciences
Leipziger Straße 123
36037 Fulda, Germany
yvonne.jung@cs.hs-fulda.de

ABSTRACT

This paper presents a system to determine differences between 3D reconstructed interiors and their corresponding 3D planning data with the aim of correcting identified differences and updating the 3D planning data based on these deviations. Therefore, a point-based comparison algorithm was developed with which deviations can be recognized regardless of the topology of the data used. Usually, resolution and topology of a 3D reconstruction do not match the CAD data. Here, our solution overcomes this problem by segmenting and extracting objects relevant for comparison (e.g., doors, windows) from the reconstruction and planning data separately with a subsequent analysis of the proximity of these objects to connected walls within the corresponding data set. Starting from the connection points of a segmented object to its walls, adjacent spatial data is located for a correction of detected differences to update the 3D planning data. The quality of the result of the developed process is shown in different examples localizing doors and windows to find deviations. In addition, detected differences between the planning and the measurement data are visualized and compared with the ground truth state of the building interior.

Keywords

3D Geometry Comparison, Geometry Deviation Feedback, Object Recognition, Computer Vision, Point Clouds, Mobile Mixed Reality

1 INTRODUCTION

In many industrial and commercial areas, Mixed Reality techniques are nowadays used in the planning or production process as well as for construction monitoring and quality control, e.g. to discover deviations that arise during a construction process at an early stage. If such deviations cannot be fixed, this could be a problem for later maintenance. Thus, our approach allows an update of the planning data respectively. Especially an early detection of differences to the original blueprints, like shifted windows, doors, or walls, can help to reduce or avoid high follow-up costs and support adherence to the schedules. This also involves a modernization or redesign of existing buildings or interiors, for example when doors are newly set or sealed. While other systems for digital construction monitoring are often limited to a visualization of the monitoring and documentation progress [ZHK+14, GFPMS09] or a visualization and comparison of detected differences between measurement and planning data [Bos10, DFFW18], the focus of the system presented in this paper, in addition to detect, visualize, and compare differences [KER+17, DGJ20], is to correct differences with a subsequent feedback into the 3D planning data.

Acquiring the depth data for a 3D reconstruction can be performed in various ways, such as by photogrammetry (Structure from Motion), Time of Flight, or Structured Light [LJS+20]. Determined spatial data is represented in the form of 3D point clouds or a 3D model is constructed from the point cloud in further post-processing [GSC+07, NIH+11]. Various analytical, region-based and geometric methods (model fitting) are used for cluster analysis of point clouds, in particular for segmenting and extracting certain elements of a scene [GMR17, SWL+16]. In addition, also more and more approaches based on machine learning are applied [ZLY19]. For a comparison of 3D data, Doboš et al. [DFFW18] described a method that recognizes differences between 3D models in the screen space based on different data such as color, depth, normals and texture coordinates and visualizes them for the user. Furthermore, Tuttas et al. [TBBS14, TSBB15, TBBS17] de-

scribed an approach that compares point clouds from real scenes with planning data in order to enable automated building documentation. The point clouds are recorded by photogrammetry and Structure from Motion. The main contribution of this paper is an algorithm that detects deviations from planning data compared to scanning data from the real world either to correct the deviations at the construction site or to use this information to update the planning data. In the latter case the differences found are used to adjust the original planning data respectively (e.g., for maintenance). Therefore, differences between planning data and captured real data are compared using a point-cloud-based data representation and the resulting differences are used to change the planning data on a geometry level.

2 PREPROCESSING

The first task of the whole system pipeline is to clean up the captured data. In our case the capturing is carried out using a Microsoft HoloLens, but our system is not limited to it, so that other 3D scanners, for example Occipital's Structure Sensor[1] for an Apple iPad, could be used. During this preprocessing, the captured point cloud is cleared of outliers and points that have arisen due to a misinterpretation of the depth data, which for example can occur due to highly reflective or transparent surfaces. After this preprocessing the identification, localization, segmentation, and extraction of objects relevant for comparison is carried out. The removal of outliers that are not part of the interior of the 3D reconstruction is achieved by using a mass histogram and the determined amplitudes. First, the orthographic projection of the point cloud is divided into individual segments of fixed size along a unit axis. Localized points within the boundary of a segment are added to a subset determining the distribution of points along that axis. Using both highest amplitudes in the histogram, the point cloud can be freed from outliers, as illustrated in figure 1 and shown in listing 1.

```
Step 1: Divide 2D coordinate plane XZ into
  segments S of fixed size along x-axis
Step 2: Project centered and aligned cloud on
  XZ coordinate plane
Step 3: For each S of XZ : count points
Step 4: Determine segment Sa and Sb
  with first and second maximum as amplitudes
Step 5: Cut off points before Sa and after Sb
Result: Trimmed cloud along x-axis
```

Listing 1: Outlier removal using the mass histogramm and both determined amplitudes along the x-axis. The same process is also done for y- and z-axis.

During the reconstruction process, objects that are relevant for comparison, such as windows or doors, may be occluded or have semi-transparent or partially intermediate regions, which makes it more difficult or even impossible to segment these objects later on for the purpose of detecting differences. If available, spatial information about such objects, like fitted units, can be obtained by the BIM (Building Information Modeling) data of a building or have to be measured manually. Another solution could be using a CNN, which identifies and locates even partwise occluded windows and doors to provide metadata of the located objects. Within our semi-automatic solution, the developed scanning software allows marking occluded regions during the reconstruction process, which can help to remove outliers within these regions and to free up occluded objects. The left side of figure 2 illustrates the reconstruction from the user's point of view including the regions marked for a later clean up. Based on the corner points of a marked region, a cuboid with twice the depth of the width of a corresponding region (negative and positive depth along the plane's normal) is used to remove points within its boundarys, which is illustrated on the right side of figure 2.

Regions marked are saved as metadata in a separate file related to the 3D model of the 3D reconstruction. The metadata contains a clear identification of the sliced object, as well as the object type, which provides information on whether the cut out object is a door or a window. Furthermore, the geometry type keeps information about what shape the user used to cut out the object, which he had previously selected from a range of predefined shapes (circle, rectangle, polygon). The anchor count contains the number of points used for the geometry type and the anchor points are stored as linear vector that contains the actual spatial data. In addition, the surface normal of the cut out geometry and the estimated unit normal are stored. Based on the anchor points and the unit normal, the dimensions of the cuboid are set up. Figure 3 contains a cut out from the metadata contained in the file, which represents the metadata related to the window within the right side of figure 2.

Figure 4 shows the 3D reconstruction in form of the raw point cloud (left) and the preprocessed point cloud (right), in which all points within the regions previously marked were removed, as well as a removal of outliers based on the mass histogram has been carried out (see figure 1). Also mentionable is the variable depth of cuboids using the determined width based on the planes created from the metadata for freeing occluded regions, which enables filtering doors or windows that are opened to the interior, as it can be seen in the right lower parts of figure 4.

3 DETECTING DIFFERENCES

To identify and locate differences between planning and actual data, segmentation and extraction of objects relevant for comparison is done individually for both

[1] Occipital Structure Sensor https://structure.io/

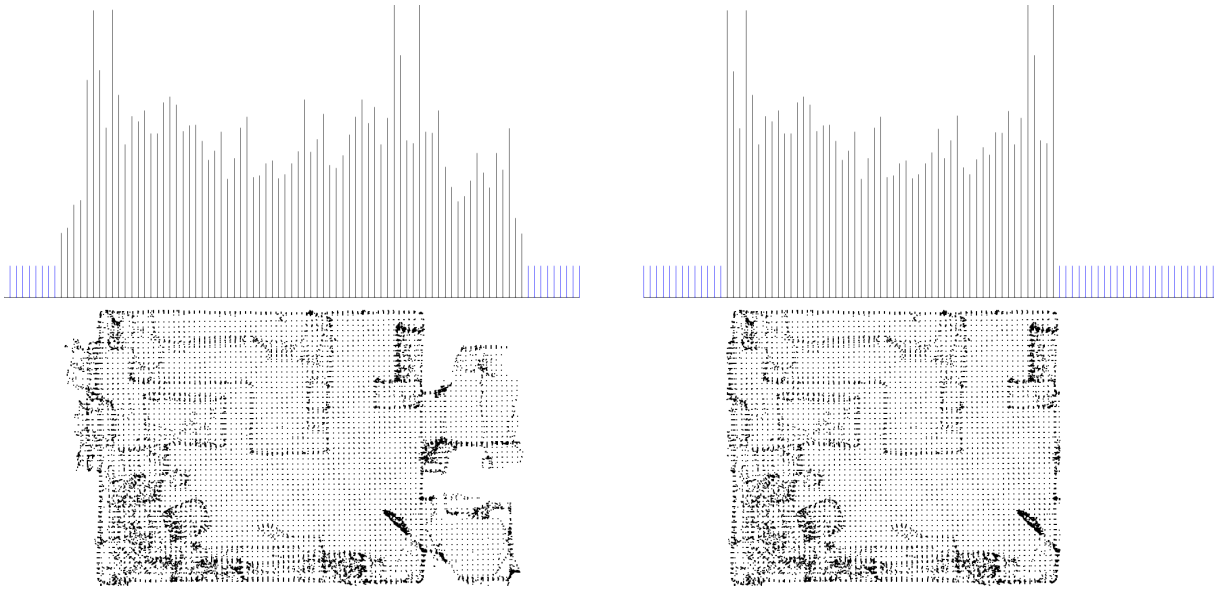


Figure 1: Left: mass histogram and corresponding point cloud. Right: point cloud cleaned of outliers based on determined amplitudes in the mass histogram along x-axis.

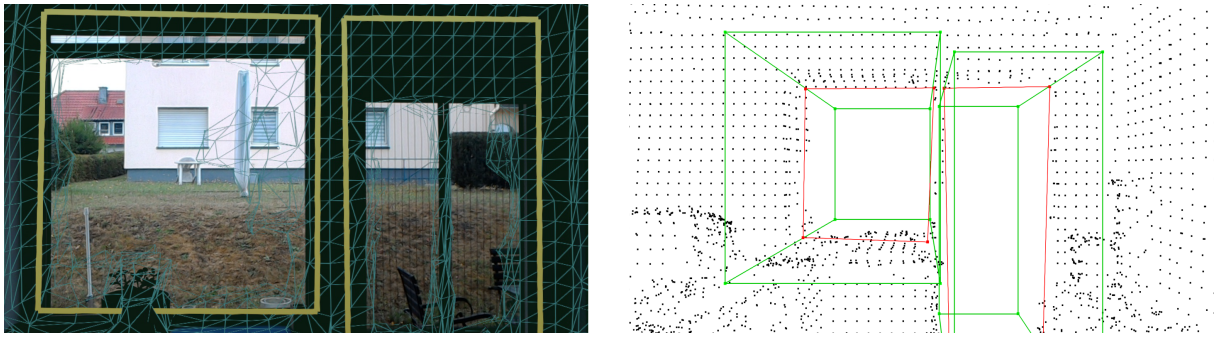


Figure 2: Creation of metadata during the 3D reconstruction process. The user marks regions that are interpreted as part of the interior due to misinterpreted depth data or occluded regions (left). The metadata created is used to remove outliers and points that have arisen due to misinterpreted depth data or that occlude objects relevant for a comparison (right).

```
# Sliced Object Data (.sod)
# Used as metadata for window or door objects in SFRA.
######## Content ########
SLICED_OBJECTS
SOCount         : 5
######## Slice0 ########
Id              : Slice0
ObjectType      : Window
GeometryType    : Rectangle
AnchorCount     : 4
AnchorPoints    : 0.3602401,0.9329683,-3.7518,-0.506594, ...
FaceNormal      : 0.04176028,-0.03948342,-0.9983472
UnitNormal      : 0,0,1
```

Figure 3: Example of the created metadata, which is optionally created during the reconstruction process and stored separately for the related 3D model of the reconstruction.

datasets. For the planning data segmentation, the 2D floor plan is converted into a planar point cloud and the wall segments are localized and extracted by use of a Euclidian Cluster Extraction (ECE)[2]. Each individual wall segment is connected to its both nearest neighbours and the located regions between two wall segments identify windows or doors as illustrated in figure 5. The left part of the figure shows the extracted ground segment. In the right part of the figure, walls, windows, and doors were segmented and extracted from the planar point cloud of the converted floor plan. Within the reconstructed point cloud, all wall faces are filtered and extracted using the amplitudes found in the mass histogram, as illustrated on the left in figure 6. Subsequently, openings are localized in the planar point cloud of the extracted wall side using α-shapes [TC98], as visualized on the right in figure 6. Figure 7 visualizes an overlay of all segmented and extracted objects relevant in the context of a difference detection in addition to the planar point cloud of the floor plan and the preprocessed point cloud of the reconstruction. Table 1 provides a comparison of the found differences between the planning data and its reconstruction concerning the width. As can be seen, differences between all extracted objects have been detected but the difference calculated for door 3 in table 1 is particularly noticeable compared to the other results. The reason for the high deviation

[2] Euclidean Cluster Extraction https://pcl.readthedocs.io/en/latest/cluster_extraction.html

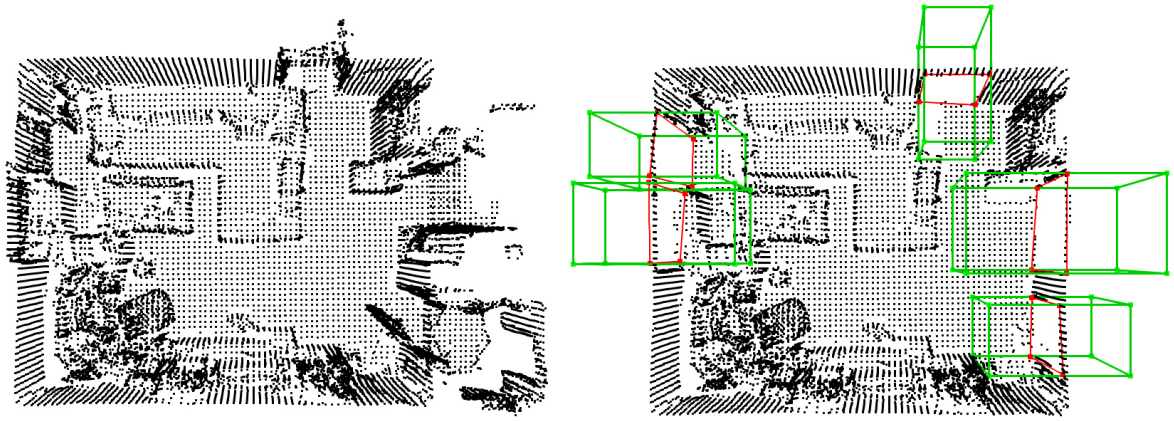


Figure 4: On the left side, the raw 3d reconstruction is visualized as point cloud. On the right, the preprocessed point cloud is visualized, using the mass histogramm to remove outliers and the metadata to clean up occluded regions.

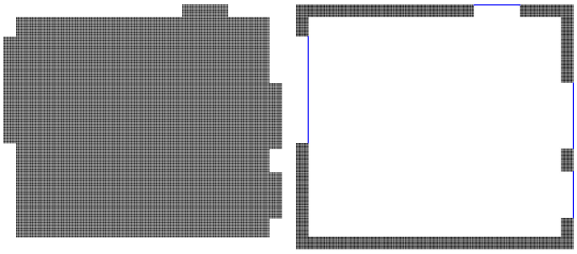


Figure 5: Planar point cloud of the floor plan split up into the extracted ground segment (left) and the individual wall segment point clouds (right) using an ECE. Each interconnection between two wall segments locates the start and end point of a window or door object within the 2D planning data.

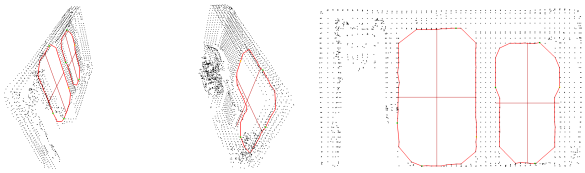


Figure 6: Localization of relevant objects for a comparison of differences using α-shapes in the planar point cloud of the extracted wall sides by the amplitudes of the mass histogram.

	Door 1	Window 1	Door 2	Door 3
Target	0.89m	2.04m	1.26m	0.89m
Actual	0.93m	2.01m	1.29m	1.04m
Difference	0.04m	0.03m	0.03m	0.15m
Compare	±4.30%	±1.49%	±2.32%	±14.42%

Table 1: Comparison of differences detected between planning and actual data in context of width.

is misinterpreted depth data that occurred during the scanning process, as can be seen from the mushroom-shaped upper part in the right door within the right side of figure 6. In addition, table 2 and table 3 are containing a comparison between the planning data, the real state of the building interior (ground truth) and the reconstruction.

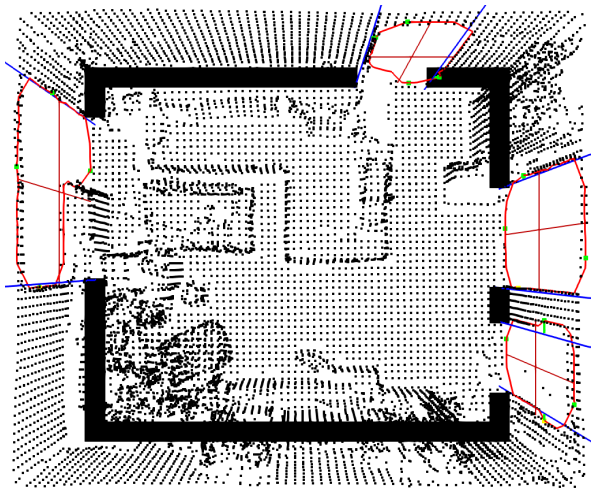


Figure 7: Overlay of the extracted regions from the planar point cloud of the floor plan by ECE (blue lines), the extracted wall segments from the floor plan point cloud (black rectangles), the preprocessed point cloud of the reconstruction as well as the extracted polygons from the segmented wall sides using α-shapes (red).

	Door 1	Window 1	Door 2	Door 3
Target	0.89m	2.04m	1.26m	0.89m
Real	0.91m	2.05m	1.25m	0.90m
Difference	0.02m	0.01m	0.01m	0.01m
Compare	±2.19%	±0.48%	±0.80%	±1.11%

Table 2: Comparison of differences between planning data and real state in context of width.

	Door 1	Window 1	Door 2	Door 3
Real	0.91m	2.05m	1.25m	0.90m
Actual	0.93m	2.01m	1.29m	1.04m
Difference	0.02m	0.04m	0.04m	0.14m
Compare	±2.15%	±1.99%	±3.10%	±13.46%

Table 3: Comparison of differences between real state and actual data in context of width.

4 UPDATING 3D PLANNING DATA

To be able to feed back detected differences into the 3D planning data, the corresponding points of a wall segment are localized for which a difference of the counterpart from the 3D reconstruction was found. Since differences are initially detected based on the determined deviations between doors and windows, these differences between two corresponding objects are used to manipulate the wall segments in the 3D planning data. If, for example, a discrepancy between a door from the planning data and the associated door from the spatial data of the reconstruction has been determined, points of the wall segments from the 3D planning data that are connected to this door are manipulated. For this purpose, all points of a wall segment that are related to a correction of the detected difference are first to be localized. This is made possible by using the adjacent properties of the individual wall segments and the connection points of two wall segments that are connected by the door or window in between. Starting from a connection point, points that are relevant for the manipulation can be localized by performing a collision detection for corner points that are located near to the location of the connection point with an intersection shape (e.g., circle or rectangle) on the planar point cloud of the planning data. Figure 8 illustrates the process.

As intersection shape a rectangular shape was used to determine the points that belong to a wall segment and are being part of the edge of this wall segment. The width and height for a rectangle is in correlation to the thickness of a wall segment and the detected difference to this location. For a better visualization, a fixed width and height was used to illustrate the localization of points relevant for a correction.

Relevant points for a correction of the determined differences were localized using the procedure described in the previous section. In the next step, the differences determined in section 3 can be fed back automatically and persistently into the 3D planning data. For this purpose, all points determined on one side of a wall segment are transformed according to the difference found. Figure 9 shows the correction of the spatial data determined based on the regions in figure 8 by an overlay of the 3D planning data before and after the correction of differences. Listing 2 shows the procedure in summary.

```
Step 1: Set up intersection shape located
  at connection point of window or door
Step 2: Localize vertices of original
  planning data within intersection shape
Step 3: Manipulate vertices based on
  determined deviations
Step 4: Save CAD file
Result: Updated CAD file
```

Listing 2: Correction of original CAD data based on detected differences.

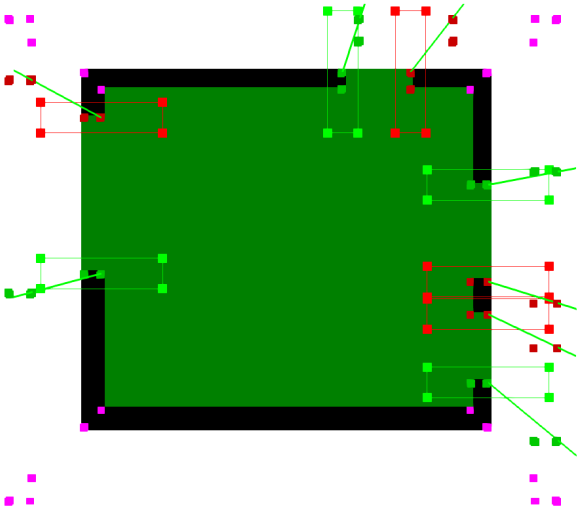


Figure 8: 2D planning data and the determined points of the 3D planning data for a correction of deviations. Points within an intersection shape belong to the wall segment that is connected to its neighboring wall segments by the window or door object in between and its starting point (green points within the green intersection shapes) and end point (red points within red intersection shapes). Purple points are not relevant for a correction of deviations.

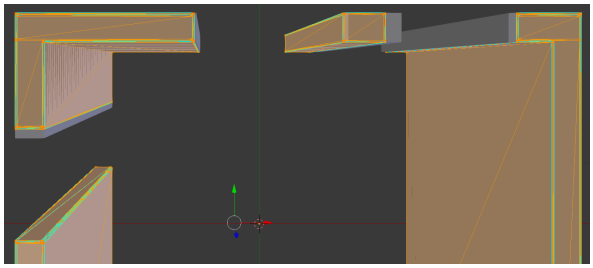


Figure 9: Overlay of the 3D planning data as 3D models before (grey) and after (yellow) the correction of differences. The view represents the 3D model of the planning data equivalent to the 2D and 3D planning data illustrated in figure 8.

5 CONCLUSION AND OUTLOOK

In this paper, we have presented a system, which allows an automated detection and visualization of differences between a 3D reconstruction of interiors and their corresponding planning data. Our proposed system is also capable of feeding back detected differences persistently into the 3D planning data. Furthermore, the preprocessing including the removal of outliers has been improved by using the metadata created during the reconstruction process with the Microsoft HoloLens to free occluded regions from points, which are not part of the object's structure.

For future work, we would firstly like to incorporate more complex types of geometries. Due to the system's actual limitation to detect and correct differences within manhattan world environments, next steps could focus on the possibility to handle more complex shaped rooms, which, for example, could be made possible by

using the floor plan point cloud for an outlier removal. In addition, further optimization and comparisons concerning the accuracy are possible. Secondly, we'd like to focus on visualization aspects, where a meta file format similar to the metadata created during the reconstruction process can be used that provides all neccessary data for a visualization of detected differences on the HoloLens and other Augmented or Mixed Reality devices. Moreover, such an AR-based difference visualization, maybe even annotated with further information about the corresponding problem, would greatly enhance on-site discussions during the building phase as well as later maintainance tasks.

REFERENCES

[Bos10] F. Bosché. Automated recognition of 3d cad model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. *Advanced Engineering Informatics*, 24:107–118, 01 2010.

[DFFW18] J. Doboš, C. Fan, S. Friston, and C. Wong. Screen space 3d diff: A fast and reliable method for real-time 3d differencing on the web. In *Proc. of the 23rd Intl. Conf. on 3D Web Technology*, Web3D '18, New York, USA, 2018. ACM.

[DGJ20] A. Dietze, P. Grimm, and Y. Jung. Visualization of differences between spatial measurements and 3d planning data. In *Proc. of the 25th Intl. Conf. on 3D Web Technology*, pages 1–5. ACM, 2020.

[GFPMS09] M. Golparvar-Fard, F. Pena-Mora, and S. Savarese. Application of d4ar - a 4-dimensional augmented reality model for automating construction progress monitoring data collection, processing and communication. *Journal of Information Technology in Construction*, 14:129–153, 06 2009.

[GMR17] E. Grilli, F. Menna, and F. Remondino. A review of point clouds segmentation and classification algorithms. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W3:339–344, 2017.

[GSC+07] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and M. Seitz, S. Multi-view stereo for community photo collections. In *2007 IEEE 11th Intl. Conference on Computer Vision*, pages 1–8, 2007.

[KER+17] M. Klomann, M. Englert, A. Rehberger, A. Dietze, T. Geier, S. Rieger, P. Grimm, and Y. Jung. NetFlinCS: A hybrid cloud-based framework to allow context-based detection and surveillance. In *Proceedings VSMM '17*, pages 1–8. IEEE, 2017.

[LJS+20] Y. Liu, J. Jiang, J. Sun, L. Bai, and Q. Wang. A survey of depth estimation based on computer vision. In *2020 IEEE 5th Intl. Conference on Data Science in Cyberspace*, pages 135–141, 2020.

[NIH+11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[SWL+16] Y. Sun, C. Wang, J. Li, Z. Zhang, D. Zai, P. Huang, and C. Wen. Automated segmentation of lidar point clouds for building rooftop extraction. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1472–1475, 2016.

[TBBS14] S. Tuttas, A. Braun, A. Borrmann, and U. Stilla. Comparision of photogrammetric point clouds with bim building elements for construction progress monitoring. volume 1, pages 341–345, 8 2014.

[TBBS17] S. Tuttas, A. Braun, A. Borrmann, and U. Stilla. Acquisition and consecutive registration of photogrammetric point clouds for construction progress monitoring using a 4d bim. *PFG - Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 85(1):3–15, 2017.

[TC98] M. Teichmann and M. Capps. Surface reconstruction with anisotropic density-scaled alpha shapes. In *Proceedings Visualization '98*, pages 67–72, 1998.

[TSBB15] S. Tuttas, U. Stilla, A. Braun, and A. Borrmann. Validation of bim components by photogrammetric point clouds for construction site monitoring. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2, 2015.

[ZHK+14] S. Zollmann, C. Hoppe, S. Kluckner, C. Poglitsch, H. Bischof, and G. Reitmayr. Augmented reality for construction site monitoring and documentation. *Proceedings of the IEEE special issue on applications of AR environments*, 102:137–154, 02 2014.

[ZLY19] W. Zhou, J. Lu, and W. Yue. A new semantic segmentation method of point cloud based on pointnet and voxelnet. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 803–808, 2019.

Similarity and symmetry measures based on fuzzy descriptors of image objects' composition

Marcin Iwanowski and Marcin Grzabka
Institute of Control and Industrial Electronics,
Warsaw University of Technology
ul.Koszykowa 75
00-662 Warszawa POLAND
iwanowski@ee.pw.edu.pl

ABSTRACT

The paper describes a method for measuring the similarity and symmetry of an image annotated with bounding boxes indicating image objects. The latter representation became popular recently due to the rapid development of fast and efficient deep-learning-based object-detection methods. The proposed approach allows for comparing sets of bounding boxes to estimate the degree of similarity of their underlying images. It is based on the fuzzy approach that uses the fuzzy mutual position (FMP) matrix to describe spatial composition and relations between bounding boxes within an image. A method of computing the similarity of two images described by their FMP matrices is proposed and the algorithm of its computation. It outputs the single scalar value describing the degree of content-based image similarity. By modifying the method's parameters, instead of similarity, the reflectional symmetry of object composition may also be measured. The proposed approach allows for measuring differences in objects' composition of various intensities. It is also invariant to translation and scaling and – in case of symmetry detection – position and orientation of the symmetry axis. A couple of examples illustrate the method.

Keywords

image similarity, symmetry detection, object detection, fuzzy logic

1 INTRODUCTION

In the paper, the method of content-based image similarity and symmetry measure is proposed[1]. The subject of comparison is the composition of elements of the visual scene. It follows the way a human compares two images. For example, one may say that two images are close to another because on both, "there is a bicycle, a car, and a dog, the dog is below the bike, shifted to the left and a car is near the bike shifted above-left".

The images under study have annotated data where the image pixel array is supplemented by the set of bounding boxes encasing the image objects. Such type of data is a typical result of, e.g., the object detection process. The method is focusing on the analysis of the mutual position of those objects. It applies fuzzy logic to describe the position of image objects' bounding boxes. The fuzzy estimation is based on the *fuzzy mutual position (FMP)* matrix that contains the fuzzy 2-D fuzzy position descriptors of every object pair [Iwa21]. Such descriptor is computed for a given object in reference to all other ones. It consists of a fuzzy set that refers to spatial relation with other ones and the value of appropriate membership function.

Two *FMP* matrices, computed separately for all objects on both images, gather the information on the composition of two scenes. In the paper, a metric is proposed that considers the separately two aspects of the fuzzy relations between objects expressed by fuzzy 2-D descriptors, which is used to measure the similarity/symmetry of the composition of objects on both scenes. The first aspect refers to non-directional properties of the position, i.e., how far is one object from another, is inside or outside, and is called *locus descriptor*. The second aspect refers to the orientation and is called *orientation descriptor*. The comparison of images is performed by matching the composition of objects' pairs on both images and measuring the pairwise similarity using *matching matrices*. The accumulated pairwise similarity is finally normalized to get the output similarity of complete images. Thanks to the separation of the orientation aspect of the 2-D descriptor, it is possible to find the symmetric orientations for each type of 2-D descriptor. Consequently, one may evaluate the reflectional symmetry within the composition of image objects.

The principal contribution of the research described in this paper is a method, allowing for applying the *FMP* matrix to estimating the level of similarity and symme-

[1] The authors conducted this research within the frames of the grant entitled "Powiedz mi, co widzisz" ("Tell me, what you see") of the POB SzIR of the Warsaw University of Technology.

tries of the composition of visual scenes. The similarity and symmetry of images are considered here not – as most of the known method does – as direct similarity of the image content but in terms of the composition of objects detected on both images using, e.g., popular deep-learning-bases object detection algorithms.

The paper is organized as follows. Section 2 contains the short survey of related papers. In section 3, the process of obtaining the *FMP* matrix is recalled. Section 4 is the core one, including describing the proposed approach to comparing images and estimating their degree of symmetry. Section 5 is devoted to the presentation and discussion on example results. Finally, section 6 concludes the paper.

2 RELATED WORKS

Image comparison methods are used in many fields of image processing and analysis. They perform comparisons at various levels, starting from the lowest, where they are used to compute accumulated pixel-wise differences of images. At the highest level, image comparison methods are used for content-based image retrieval. More sophisticated image comparison methods work by building descriptors from image features at different levels and comparing them with various metrics and algorithms. Low-level features may consist of color, texture, shape, or spatial location information. They can be extracted both locally or globally, at the image level. These features may be compared using distance in color space, histogram similarity measures, Minkowski, cosine, and Earth mover's distance distances, or non-parametric chi-square test [Rod00; Liu07]. Some approaches like SSIM produce image similarity scores by comparing luminance, contrast, and structural features [Wan04; Wan03]. At a higher level, region segmentation algorithms, like k-means, JSEG [Den01], may utilize low-level features to extract and match similar areas in the images.

Detecting symmetry in the image is a vital part of image understanding and can be used, e.g., for object detection and data compression. Symmetry can be divided into bilateral symmetry, rotational symmetry, and curve symmetry [Xia07]. In [Sco03] a two-stage algorithm has been proposed that locates visually salient features of the image by computing local energy function and then applies a Gaussian filter to extract the symmetry axis. An approach based on gradient orientation histogram and Fourier transform was proposed in [Sun99]. The method described in [Loy06] makes use of rotation invariant feature descriptors. The eigenvector decomposition of the covariance matrix to find bilateral symmetry hyperplanes was used in [OMa96].

Thanks to the rapid development of the deep-learning methods [LeC15], various techniques for detecting and identifying the content of the digital image have been proposed. One of the most popular, primarily due to relatively high computational efficiency, is the object detection methods [Gir15; Red16; Ren17] that result in a handy set of bounding boxes encasing the image objects. The object detection methods are fast and – in many cases – able to process images in real-time. Their output may be further processed to get more sophisticated information on the higher-level properties of the visual scene.

Describing the position of objects located within the image is another field of extensive study. Thanks to its ability to deal with the uncertainty that always accompanies a visual scene's interpretation, fuzzy logic has been often used as a principal tool in this field. In [Zha93; Miy94] different fuzzy methods of scene representation, recognition, and description of an image with detected semantically consistent image regions. The latter paper introduced the histogram of orientations that was further investigated in many papers [Mat01; Kel00]. Detailed surveys of this field have been done in [Sme00; Coh01; Blo05].

3 EXTRACTING THE FUZZY MUTUAL POSITIONS MATRIX

The notion of *fuzzy mutual position* (FMP) matrix that has recently been proposed in [Iwa21] describe the content of the visual scene consisting of objects annotated by their bounding boxes. It is a data structure consisting of fuzzy membership function values for all pairs of image objects. Membership function values refer to fuzzy 2D position descriptors defined as fuzzy sets associated with particular spatial relations between two objects, e.g.," [one object] is located inside to the right [of the second one]". The details are this concept are recalled in following subsections.

3.1 Fuzzification of bounding box corners' coordinates

Let focus at the beginning on the two-object case, let $B = \{(x_B, y_B), (x'_B, y'_B)\}$ be a given bounding box defined by the position of its upper-left (x_B, y_B) and lower-right (x'_B, y'_B) corner. Let $R = \{(x_R, y_R), (x'_R, y'_R)\}$ be the second bounding box – the *reference* one. At first, the position of B will be expressed relative to R using the fuzzy position descriptor of its corner coordinates.

The relative position of any given image point $(\mathbf{x}, \mathbf{y})$ in reference to R is defined as:

$$\mathbf{x} = 2 \cdot \frac{x - x_R}{x'_R - x_R} - 1 \; ; \; \mathbf{y} = 2 \cdot \frac{y - y_R}{y'_R - y_R} - 1. \qquad (1)$$

Using the above equation, the coordinates of corners of the bounding box B are transformed into relative ones:

$$B_R = \{(\mathbf{x}_{B\to R}, \mathbf{y}_{B\to R}), (\mathbf{x}'_{B\to R}, \mathbf{y}'_{B\to R})\}, \qquad (2)$$

where $\mathbf{x}_{B\to R}$ stands for the relative coordinate of the given corner of the bounding box B with bounding box R used as the reference one.

The relative position $(\mathbf{x},\mathbf{y})$ is next fuzzyfied [Zad65] separately for each axis into one of 5 fuzzy sets associated with the relative position of a point in relation to the reference bounding box R. They are described using trapezoidal membership functions. The following fuzzy sets are taken into account: *inside (i)*, *edge (e)*, *close (c)*, *near (n)*, and *far (f)*. The fuzzy values describe the distance zone counted starting from the centroid of R. In order to fully describe the position, in addition, the direction in which the point is located is coded using one of four values: *left (l)*, *right (r)*, *above (a)*, and *below (b)*. Fuzzification is performed for each axis separately, so finally there are two sets of fuzzy descriptors of corners of B, describing position along the x-axis: $D_{0x} = \{fl, nl, cl, el, il, ir, er, cr, nr, fr\}$ and along y-axis: $D_{0y} = \{fa, na, ca, ea, ia, ib, eb, cb, nb, fb\}$. As a result of the fuzzification process, each corner of the bounding box B relative to R along each of axes is described using one or two linguistic variables referring to point position descriptors associated with fuzzy sets and appropriate membership function values. These variables are *fuzzy descriptors of a point (corner)* relative to reference object R.

3.2 Determining the 2-D fuzzy bounding box position

In the second step, starting from the corners' fuzzy descriptors, the next level descriptors are computed – fuzzy *descriptors of edges* of B relative to R. They are further referred to as *1-D descriptors* as they reflect a bounding box's positions along single axes of the image coordinate system. Each bounding box is defined by four groups of corner descriptors (two for x-coordinates, two for y-coordinates, each of which consists of 1 or 2 descriptors), from which two groups of edge ones are derived. Depending on the position of an edge relative to the equivalent edge of the reference bounding box, the 1-D descriptor belongs to the following groups:

1. **outside position** – edge of B is located outside the edge of R without any intersection. There are three such descriptors, depending on how far B is from R: *far* (*FA*), *near* (*NE*), *close* (*CL*). In addition, the list of outside positions contain one additional case, when the edge of R is included in the edge of B: descriptor *longer* (*LO*)

2. **crossing position** – edge of B intersects the edge of R: possibly – *touching* (*TO*) and actually – *crossing* (*CR*),

	fl	*nl*	*cl*	*el*	*il*	*ir*	*er*	*cr*
fl	*FA/L*	*NE/L*	*CL/L*	*TO/L*	*CR/L*	*CR/L*	*CR/L*	*LO/H*
nl	—	*NE/L*	*CL/L*	*TO/L*	*CR/L*	*CR/L*	*CR/L*	*LO/H*
cl	—	—	*CL/L*	*TO/L*	*CR/L*	*CR/L*	*CR/L*	*LO/H*
el	—	—	—	*TO/L*	*IN/L*	*IN/L*	*SA/H*	*CR/R*
il	—	—	—	—	*IN/L*	*SH/H*	*IN/R*	*CR/R*
ir	—	—	—	—	—	*IN/R*	*IN/R*	*CR/R*
er	—	—	—	—	—	—	*TO/R*	*TO/R*

Table 1: A part of $\mathbf{FAM}_1$ for estimating the 1-D fuzzy position descriptors, for horizontal axis (reduced due to space limitations, for complete 10x10 version see [Iwa21]).

3. **inside position** – edge of B is entirely included in the edge of R: close to one of boundaries – *inside* (*IN*) or around the center of R – *shorter*(*SH*).

The above descriptors are used along with orientation indicators that reflect the direction of a given property (one of 9 above listed). There are 4 indicators that follows the principal directions: *left* (*L*)–*right* (*R*), for the x-axis (set of all 1-D descriptors along this axis will be denoted as D_{1x}) and *above* (*A*)–*below* (*B*) for the y-axis (D_{1y}). There exist, however, some cases when exact direction cannot be determined. In these cases, the simplified indicators are used: *horizontal* (*H*) or *vertical* (*V*). They are accompanying descriptors *SH, SA, LO*.

The 1-D descriptors are computed based on fuzzyfied coordinates of bounding box corners – upper left and lower right (eq. 2), which also depict the position of edges of the bounding box. They are ordered: $x_{B\to R} \leq x'_{B\to R}$ and $y_{B\to R} \leq y'_{B\to R}$, which implies that not all combinations of fuzzy descriptors of corners have to be considered (e.g. the left edge can only be located on the left side of the right edge – cannot be located on the right side of the latter). Since the number of corner descriptors equal $|D_{0x}| = |D_{0y}| = 10$, the total number of possible combinations equals $\sum_{i=1}^{10} i = 55$. Based on those combinations, the values of particular 1-D (edge) descriptors are computed using the *fuzzy associative matrix* (**FAM**), which allows for finding the appropriate 1-D descriptor for each pair of edge descriptors. In this case, it is a 10x10 matrix with 55 relevant values.

Relations between two point descriptors of the horizontal direction (x-axis) in the bounding box B ($\mathbf{x}_{B\to R}$ and $\mathbf{x}'_{B\to R}$) and 1-D position descriptor of the edge stretched between $\mathbf{x}_{B\to R}$ and $\mathbf{x}'_{B\to R}$ in relation to the horizontal edge of R are shown partially in Table 1. It exhibits the fuzzy associative matrix $\mathbf{FAM}_1$, rows of which refer to the fuzzy position of $\mathbf{x}$, columns to the fuzzy position of $\mathbf{x}'$. Elements of this matrix contain the fuzzy 1-D descriptor of the edge. Similar matrix for vertical axis is obtained by replacing orientation indicators: $l \leftrightarrow a$, $r \leftrightarrow b$, $L \leftrightarrow A$, $R \leftrightarrow B$, $H \leftrightarrow V$.

The $\mathbf{FAM}_1$ is used to compute membership functions to fuzzy sets related to particular 1-D descriptors. This computation is performed using Mamdani fuzzy reasoning (min-max principle):

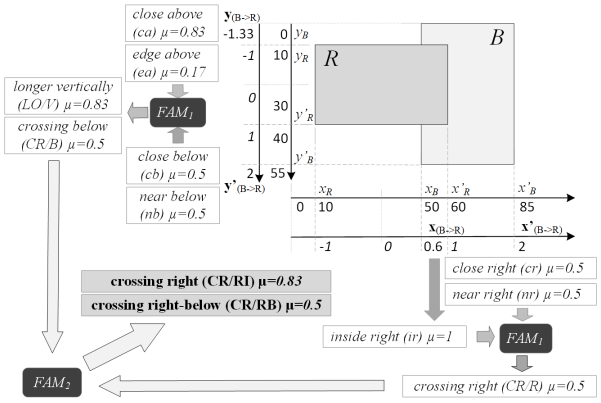


Figure 1: Illustration of the process of generating the 2-D fuzzy descriptors of example bounding box B relative to the reference R consisting of tranformation of coordinates, fuzzification of bounding-boxes' corners, 1st reasoning resulting in estimated fuzzy 1-D position of edges and 2nd reasoning resulting in final 2-D descriptors.

$$\mu_d(\mathbf{p},\mathbf{p}') = \max\left\{\min(\mu_{d_1}(\mathbf{p}),\mu_{d_2}(\mathbf{p}')\right\}, \qquad (3)$$

where $\max\{\}$ is computed for all d_1, d_2 such that $\mathbf{FAM}_1(d_1,d_2) = d$. In the case of x-axis $\mathbf{p} \equiv \mathbf{x}_{B\to R}$, $\mathbf{p}' \equiv \mathbf{x}'_{B\to R}$, $d_1, d_2 \in D_{0x}$, and $d \in D_{1x}$. In the case of y-axis $\mathbf{p} \equiv \mathbf{y}_{B\to R}$, $\mathbf{p}' \equiv \mathbf{y}'_{B\to R}$, $d_1, d_2 \in D_{0y}$, and $d \in D_{1y}$.

In the third and last stage of processing, the final 2-D set of descriptors of B relative to R is computed starting from two 1-D descriptors using another fuzzy associative matrix $\mathbf{FAM}_2$. The 2-D descriptor contains the information on the complete bounding-box 2-D position B relative to R. 2-D descriptors express two types of spatial properties of the object B relative to R: locus-based and orientation-based. *Locus-based* properties indicate where an object is located relative to the reference one and/or position without considering the direction. Some of them follow the concept of particular 1-D descriptors: *far* (FA), *near* (NE), *close* (CL), *touching* (TO), *crossing* (CR), *inside* (IN). The next two also follow the 1-D descriptors but do not express any directional properties: *same* (SA), and *larger* (LG). The latter is equivalent to 1-D descriptor *longer* (*LO*), but its name fits better to the 2-D position. The last descriptor – *split* (SP) refers to the case when the given object splits the reference one into two parts, which do not have a 1-D equivalent.

The *orientation-based* descriptors indicate a direction in which the given locus-based descriptor property is fulfilled. There is a total number of 11 orientation-based descriptors. Eight of them indicate the primary directions: *left* (LE), *left-above* (LA), *above* (AB), *right-above* (RA), *right* (RI), *right-below* (RB), *below* (BE) and *left-below* (LB) and are used together with locus-based descriptors: FA, NE, CL, TO, CR, and IN.

	CL/L	*TO/L*	*CR/L*	*IN/L*	*SH/H*	*SA/H*	*LO/H*
FA/A	FA/LA	FA/AB	FA/AB	FA/AB	FA/AB	FA/AB	FA/AB
NE/A	NE/LA	NE/LA	NE/AB	NE/AB	NE/AB	NE/AB	NE/AB
CL/A	CL/LA	CL/LA	CL/AB	CL/AB	CL/AB	CL/AB	CL/AB
TO/A	CL/LA	TO/LA	TO/LA	TO/AB	TO/AB	TO/AB	TO/AB
CR/A	CL/LE	TO/LA	CR/LA	CR/AB	CR/AB	CR/AB	CR/AB
IN/A	CL/LE	TO/LE	CR/LE	IN/LA	IN/AB	IN/AB	SP/AB
SH/V	CL/LE	TO/LE	CR/LE	IN/LE	IN/CE	SP/HO	SP/HO
SA/V	CL/LE	TO/LE	CR/LE	IN/LE	SP/VE	SA/CE	LG/HO
LO/V	CL/LE	TO/LE	CR/LE	SP/LE	SP/VE	LG/VE	LG/CE

Table 2: A part of $\mathbf{FAM_2}$ for estimating the 2-D fuzzy position descriptors (reduced due to space limitations, for complete 15x15 version see [Iwa21]).

Moreover, descriptors LE, RI, AB, and BE are used with the SP locus-descriptor to indicate splitting the reference object close to the particular edge. The first of the remaining three descriptors reflect the configuration when given bounding box have the same position of the centroid and same proportions: *centered* (CE) and is used together with locus-descriptors IN, SA, and LG. The last two direction descriptors *horizontal* (HO) and *vertical* (VE) are used only in combination with SP locus-descriptor to express the case of splitting of the reference bounding boxes in the middle of its either height or width. Considering all the above, the total number of 2-D descriptors – combinations of locus-based and orientation-based – equals $|D_2| = 57$. Considering the fact that $|D_{1x}| = |D_{1y}| = 15$, the size of $\mathbf{FAM}_2$ is 15x15 – it contain all combinations of 1-D descriptors along x- and y-axis. A part of $\mathbf{FAM}_2$ matrix is shown in Table 2.

Simmarily to the 1-D case, the 2-D descriptor is obtained using the Mamdani reasoning, this time using $\mathbf{FAM}_2$ matrix. For the bounding box B defined by the Eq. 2, the membership function value of a 2-D descriptor d is equal to:

$$\mu_d(B_R) = \max\{\min(\mu_{d_x}(\mathbf{x}_{B\to R},\mathbf{x}'_{B\to R}),\mu_{d_y}(\mathbf{y}_{B\to R},\mathbf{y}'_{B\to R})\}, \qquad (4)$$

where $\max\{\}$ is computed for all $d_x \in D_{1x}$ and $d_y \in D_{1y}$ such that $\mathbf{FAM}_2(d_x,d_y) = d$. The fuzzy relation d belongs to the set of all 2-D fuzzy relations D_2. The complete process of obtaining the 2-D fuzzy descriptors for an example object is shown in Fig. 1

With the usage of two **FAM** matrices, which are the loo-up-table ones, the calculations of fuzzy descriptors are fast. Nevertheless, one may speed it up even more by joining both matrices into a single 4-dimensional data structure, indexed by fuzzified corner coordinates (two along the x-axis and two along the y-axis).

3.3 Scene description using FMP matrix

Considerations in the previous section concern the case of relations between *two* objects. To get the description of the complete scene that consists of multiple objects, a matrix of one-to-one relations has been introduced, called the *fuzzy mutual position* (*FMP*) matrix. It consists of relative position descriptors for all combinations

of bounding boxes. Let $\mathbf{S} = \left\{B^{(1)}, B^{(2)}, ..., B^{(n)}\right\}$ be the visual scene – the set of n bounding boxes of the image. The FMP matrix is defined in $n \times n \times |D_2|$ as:

$$FMP_{c,r,d}(\mathbf{S}) = \mu_d\left(B^{(c)}_{B^{(r)}}\right) \quad 1 \leq c, r \leq n; d \in D_2, \quad (5)$$

where c and r stands for objects' indexes: current and reference, respectively, and d stands for the fuzzy 2-D descriptor[2]. The c-th row consists of 2-D fuzzy descriptors of c-th object computed relative to all other objects considered as reference ones. The element of index (c,r) contains a vector of descriptors of $B^{(c)}$ relative to $B^{(r)}$, and will be further referred to as $FMP_{c,r}(\mathbf{S})$.

4 SIMILARITY AND SYMMETRY DETECTION

The FMP matrix describes the visual scene using the object's fuzzy descriptors of objects concerning other ones. One can say that two objects are in a similar position if their fuzzy 2-D descriptors are compliant. Let $B^{(i)}$ and $B^{(j)}$, be two object present on the first image (i, j are their indexes). They are also present on the second image (to be compared with the first one); let us call them $B'^{(i)}$ and $B'^{(j)}$. They may be located on both images in different places (their bounding boxes may have completely different image coordinates). We can then say that their compositions are similar if the fuzzy 2-D descriptors on both images (stored in the FMP matrices) are close one to another i.e. $B^{(i)}_{B^{(j)}} \approx B'^{(i)}_{B'^{(j)}}$ and $B^{(j)}_{B^{(i)}} \approx B'^{(j)}_{B'^{(i)}}$. If we can draw the same conclusion from all other pairs i, j (or, at least, for the vast majority of them), then the composition of objects on both images may be considered similar. The ratio of matched bounding boxes to all bounding boxes present within images may be proportional to the overall composition similarity.

The above considerations may also be used to measure the symmetry of the image objects' composition. However, this time, the 2-D descriptor should be considered as composed two of two parts: locus-descriptor and orientation descriptor. The locus descriptor reflects an object's position relative to the reference one in terms of the fuzzy estimation of the relative distance to the reference object. It tells us whether the object is far, near, border-crossing, inside, etc. These properties are invariant to symmetry. There is, however, also the second part of the 2-D descriptor – the orientation descriptor. This one, in turn, strongly depends on image symmetry because it describes the direction in which one may find the given object looking from the reference one. In the symmetrical composition of object bounding boxes, these descriptors should be compared to their symmetrical counterparts. For example, we can say that the composition of some $B^{(i)}$ and $B^{(j)}$ is symmetric along x-axis, if $B^{(i)}$ is *close* to $B^{(j)}$ on both images but on the first it is located on the *left*, while on the second, on the *right*-hand side.

The detailed description of the proposed approach to detecting similarity and symmetry is in the following subsections.

4.1 Similarity detection

Let $\mathbf{S}$ be the first scene (set of bounding boxes of the first image), where the total number of bounding boxes equals n_1. Let $\mathbf{S}'$ be the second scene – the set of n_2 bounding boxes of the second image. Let, $id(B)$ be a function that returns the bounding box's ID-number. ID-numbers are unique numbers that identify the bounding boxes. We will use them to establish correspondences between objects. Each bounding box B on the first image has its counterpart B' on the second image iff $id(B) = id(B')$. Let assume that some number $n_0 > 0$ of bounding boxes from $\mathbf{S}$ and $\mathbf{S}'$ are matched, i.e., have the same ID number[3] The total number of different objects on both images equals thus to $n = n_1 + n_2 - n_0$. In the first step, objects present on both images are extracted to unified sets $\mathbf{C}$ and $\mathbf{C}'$ that, comparing to the original sets, are trimmed to the same length n_0 and ordered so that the bounding boxes of the same index have the same ID.

In order to estimate the total similarity of two images, in the beginning, the pairwise similarity between objects that are present on both images is evaluated by considering the appropriate elements of the mutual position matrices of both images. We thus reduce the problem of comparing two scenes to the process of investigating pairs of relations between sets of objects on two images. The FMP matrix contains values of membership functions to all possible fuzzy 2-D descriptors fuzzy sets. For a given pair of objects, only a few (usually one or two) values are greater than 0. These descriptors indicate the possible relations between objects. Because membership functions' values indicate the strengths of those relations, the membership function values are thresholded, so the weakest relations are rejected. To compare a pair of relations between bounding boxes, one compares fuzzy descriptors of membership function values greater than a given threshold t. In order, in turn, to compare descriptors, each of two

[2] Due to the fact that, in general, fuzzy position descriptor is not symmetric i.e. $B^{(i)}_{B^{(j)}} \neq B^{(j)}_{B^{(i)}}$, FMP is not symmetric neither.

[3] In case $n_0 = 0$ we will not be able to estimate the similarity at all – two images would be completely different, consisting of disjoint sets of objects. To estimate the similarity, at least two objects with fuzzy 2-D descriptors of their mutual relation must be present on both images with pairwise the same ID's.

	FA	NE	CL	TO	CR	IN	LG	SP	SA
FA	1	vh	vl	0	0	0	0	0	0
NE	vh	1	vh	vl	0	0	0	0	0
CL	vl	vh	1	vh	vl	0	0	0	0
TO	0	vl	vh	1	vh	vl	0	0	0
CR	0	0	vl	vh	1	vh	0	0	0
IN	0	0	0	vl	vh	1	0	vl	vl
LG	0	0	0	0	0	0	1	vl	vl
SP	0	0	0	0	0	vl	vl	1	vl
SA	0	0	0	0	0	vl	vl	vl	1

Table 3: Matching matrix for locus descriptors MM_{loc}, intermediary values are equal to vh = 0.5 and vl = 0.25.

	LE	LA	AB	RA	RI	RB	BE	LB	CE	HO	VE
1, LE	1	vh	0	0	0	0	0	h	0	vl	0
2, LA	hi	1	vh	0	0	0	0	0	0	vl	vl
3, AB	0	vh	1	vh	0	0	0	0	0	0	vl
4, RA	0	0	vh	1	vh	0	0	0	0	vl	vl
5, RI	0	0	0	vh	1	vh	0	0	0	vl	0
6, RB	0	0	0	0	vh	1	vh	0	0	vl	vl
7, BE	0	0	0	0	0	vh	1	vh	0	0	vl
8, LB	hi	0	0	0	0	0	vl	1	0	vl	vl
9, CE	0	0	0	0	0	0	0	0	1	vl	vl
10, HO	vl	vl	0	vl	vl	vl	0	vl	vl	1	0
11, VE	0	vl	vl	vl	0	vl	vl	vl	vl	0	1

Table 4: Matching matrix for orientation descriptors MM_{or}, intermediary values are equal to vh = 0.5 and vl = 0.25.

parts of the descriptors, locus and orientation, are investigated separately. For each part, one has to consider three cases. The first case refers to the equality of descriptor, i.e., the locus descriptor is equal to the same value on both images, e.g., "far". It is a clear case – the distance is in the same range on both images, the locus-based similarity is complete. The second case refers to the situation when descriptor values are not equal but semantically close one to another, e.g., its value on the first image equals 'far' while on the second one – to "near". In this case, a particular (lower) degree of similarity can also be assigned. Finally, there exists the third case. It refers to values that are semantically far from another (e.g. 'inside' vs. 'far'). In such a case, the similarity equals 0.

The same is true for orientation descriptors. Here one also should differentiate similarity degrees for cases of full (e.g. 'right' or both images), partial (e.g. 'right' on the first image, and 'right-above' on the second), and null (e.g. 'right' vs. 'above') accordance of descriptor values.

To properly and efficiently process the possible combinations of various descriptors' values, the *matching matrices MM* are used. There are two such matrices, one per each descriptor part (locus and orientation). Rows and columns of those matrices refer to descriptors' values to be matched. Their elements are real numbers between 0 and 1 which are weighting coefficients applied to evaluate a given combination's weight. Matching matrix for locus descriptor MM_{loc} is presented in Table 3. In the further considerations notation $MM_{loc}(d,d')$ is referring to the element of locus matching matrix for locus parts of descriptors d and d'. Similarly, values in Table 4 are orientation matching values, further referred to as $MM_{or}(d,d')$.

The matching matrices are used as a kind of look-up-tables to compute the similarity between objects' relative positions on both images. The final similarity measure is computed as normalized *accumulated pairwise similarity*. The complete workflow of the method of its computing is presented as Algorithm 1.

Algorithm 1 – computation of accumulated pairwise similarity

input data: $\mathbf{S}$,$\mathbf{S}'$ sets of bb's of two images
input parameter: t threshold
output: s_{acc} accumulated pairwise similarity

```
1   compute C, C', n and n_0 based on S and S'
2   compute FMP(C) and FMP(C')
3   s_acc ← 0
4   for i ← 1,...,n_0:
5     for j ← 1,...,n_0:
6       s ← 0
7       for d ← 1,...,|D_2|:
8         for d' ← 1,...,|D_2|:
9           if FMP_{i,j,d}(C) > t ∧ FMP_{i,j,d'}(C') > t:
10            s_loc = MM_loc(d,d')
11            s_or = MM_or(d,d')
12            s = max(s, s_loc · s_or)
13      s_acc ← s_acc + s
```

Two external loops of the algorithm (lines 4-5 up to 13) iterate all pairs of objects. For each pair of objects, their mutual position on both images is taken from their FMP matrices to compute pairwise similarity measure – stored in variable s (initialized in line 6). These computations are performed in two internal loops (lines 7-8 up to 12). They iterate all combinations of 2-D descriptors of the current two objects to find all pairs of descriptors with membership function values greater than the given threshold t (line 9). Each such finding means that there exists a pair of meaningful descriptors for which the pairwise similarity is computed as a product (line 12) of two values being the elements of two matching matrices. The first one reflects the pairwise similarity between loci (line 10, Table 3), while the second – between orientations (line 11, Table 4). The final value of pairwise similarity is the highest value of all the above findings (line 12). The similarity values for all object pairs are summed up and finally returned as the accumulated pairwise similarity (stored in variable s_{acc} initialized in line 3 and updated in line 13).

The value of accumulated pairwise similarity, being the result of Algorithm 1, is used to finally compute the similarity measure of images (represented by sets of

their proper bounding boxes) $\mathbf{S}$ and $\mathbf{S}'$ using the following normalization:

$$sim(\mathbf{S},\mathbf{S}') = \frac{s_{acc}}{n \cdot n_0} \tag{6}$$

This expression represents the average value of pairwise similarity ($\frac{s_{acc}}{n_0^2}$) multiplied by a factor $\frac{n_0}{n}$ that is the ratio of matched objects. If all objects are matched, i.e., each object of one image has a counterpart on the second ($n = n_0$), this factor equals 1. The final similarity measure is an average pairwise similarity. However, if some objects are present on only one image, the similarity factor is reduced accordingly. It follows the common-sense observation that the higher, among the total number of image objects, is the number of objects visible exclusively on one image, the lower is the images' similarity.

4.2 Reflectional symmetry detection

Thanks to the separation of locus and orientation descriptors, the information on mutual relation directions can be analyzed independently on the locus information. In particular, one may observe very interesting results when analyzing the relation of the orientation descriptors and their relations to the composition of objects on both images being the subject of comparison.

Let us imagine two similar visual scenes such that one is mirrored about another. The composition of objects in terms of their loci (how far are objects one from another, inside or outside others, etc.) is the same – locus descriptors may have similar values. However, in orientation-descriptors, the relations on one image convert themselves to their counterparts on the other. The left-hand side moves to the right-hand side and vice-versa. Thus, the opposite ones replace all the orientation descriptors: *left* become *right* and reversely.

The above properties can be encoded in the matching matrix MM_{or}. To get its version that measures symmetry, some of its rows must be interchanged (permuted) so that the best match is not with the same orientation descriptor on the second image but with its symmetric reflection. In case of classic mirroring, symmetry along x-axis, best match should be for the *left* $\leftrightarrow$ *right* correspondence. In the case of y-axis symmetry, the replacement should refer to *above* $\leftrightarrow$ *below* correspondence and in the case of diagonal symmetry – to both.

Let $perm(MM,[a_1,a_2,...])$ be the operation that permutes rows of the MM matrix so that the vector indicates their order in resulting matrix is the second argument, where $a_1,a_2,...,$ are indexes of the original matrix: the first row of the permuted matrix is the a_1-th row of the original, the second row is the a_2-th, etc. The matching orientation matrices for three principal symmetries are the following (x-axis, y-axis, xy-axis, respectively):

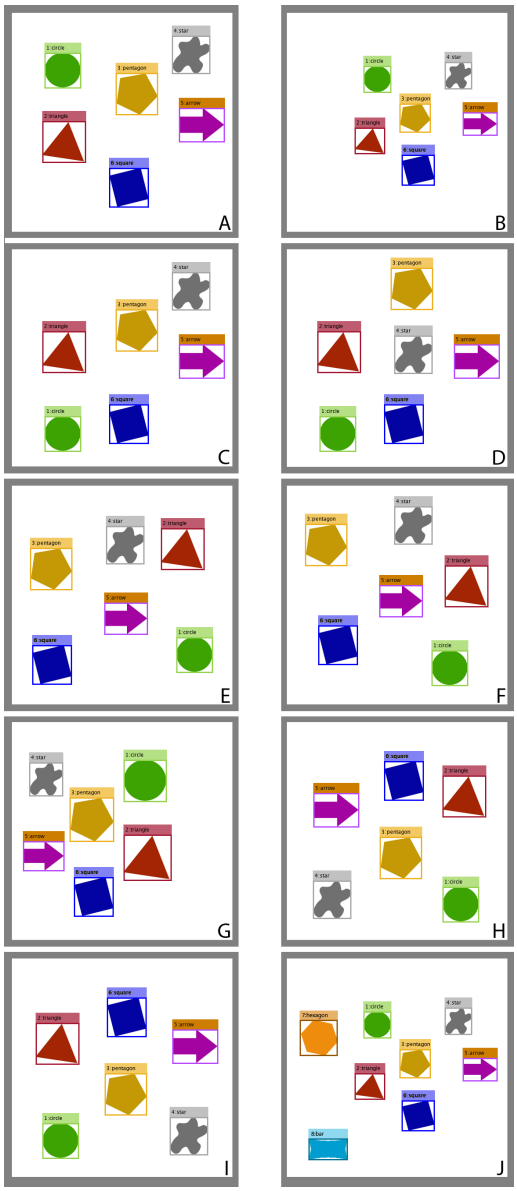


Figure 2: Set of 9 artificial test images with 6 object classes: *circle*, *square*, *triangle*, *pentagon*, *arrow*, and *star*.

$$\begin{aligned} MM_{or}^{x} &= perm(MM_{or},[5,4,3,2,1,8,7,6,9,10,11]) \\ MM_{or}^{y} &= perm(MM_{or},[1,8,7,6,5,4,3,2,9,10,11) \\ MM_{or}^{xy} &= perm(MM_{or},[5,6,7,8,1,2,3,4,9,11,10] \end{aligned} \tag{7}$$

By replacing, in the Algorithm 1, the original MM_{or} by one of the above variants, the measure obtained is proportional not to similarity of objects compositions, but to the given type of reflectional symmetry.

5 RESULTS

The method has been tested on both artificially created images and images being the results of the object detec-

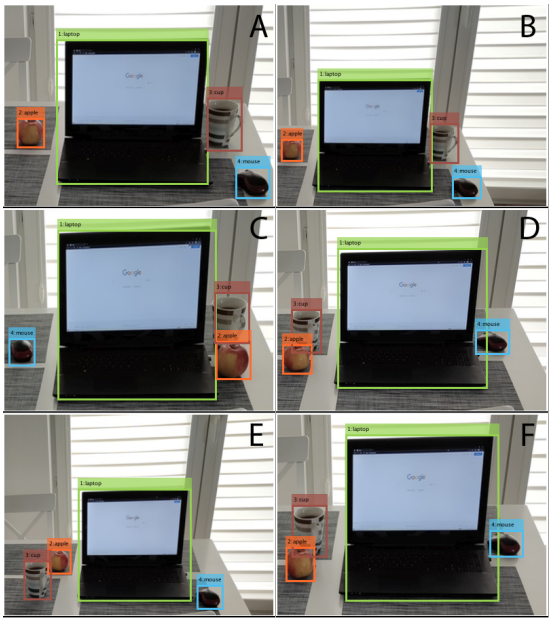


Figure 3: Set of 6 real images processed using object detector (4 object classes detected on each image: *laptop*, *apple*, *mouse*, *cup*).

A	B	C	D	E	F	G	H	I	J	
1.00	**0.97**	**0.76**	0.53	0.44	0.44	0.33	0.17	0.39	0.72	A
-	1.00	**0.76**	0.50	0.44	0.44	0.33	0.17	0.39	0.75	B
-	-	1.00	0.71	0.47	0.53	0.28	0.19	0.51	0.57	C
-	-	-	1.00	0.36	0.42	0.21	0.22	0.47	0.38	D
-	-	-	-	1.00	0.89	0.41	0.39	0.28	0.33	E
-	-	-	-	-	1.00	0.44	0.33	0.26	0.33	F
-	-	-	-	-	-	1.00	0.39	0.17	0.25	G
-	-	-	-	-	-	-	1.00	0.33	0.13	H
-	-	-	-	-	-	-	-	1.00	0.29	I
-	-	-	-	-	-	-	-	-	1.00	J

Table 5: Similarity measures of pairs of images from the test set 1 shown in Fig. 2.

A	B	C	D	E	F	
1.00	**1.00**	0.38	0.49	0.62	0.49	A
-	1.00	0.38	0.50	0.56	0.50	B
-	-	1.00	0.31	0.25	0.34	C
-	-	-	1.00	0.73	**1.00**	D
-	-	-	-	1.00	0.73	E
-	-	-	-	-	1.00	F

Table 6: Measures of similarity of image pairs (test set 2 shown in Fig. 3)

tion algorithm. The first set of scenes, shown in Fig. 2 consist of 10 images consisting of 6 or 8 objects in different mutual positions. The differences are of various kind: modifications of absolute position without changing their mutual positions considerably (see image A vs. B or E vs. F), two object replacement (see A vs. C, C vs. D), more replacements (e.g., B vs. D), x-symmetry (e.g., A vs. G), y-symmetry (A vs. I), different number of objects (B vs. J). The similarity and symmetry measures for all pairs of images are given in Tables 5, 7, 8, 9.

The second example scenes are real images presenting the same vision scene consisting of four objects: *laptop*, *apple*, *mouse*, *cup* automatically annotated using object detection method (see Fig. 3). Here also various configurations of objects are present. We can observe a variation of scale (see image A vs. B), the various position of objects (A and B vs. C, D, E), as well as mirror

A	B	C	D	E	F	G	H	I	J	
0.33	0.33	0.28	0.21	0.41	0.42	**0.85**	0.39	0.17	0.25	A
-	0.33	0.28	0.21	0.41	0.42	**0.89**	0.39	0.17	0.25	B
-	-	0.33	0.27	0.55	0.58	0.65	0.50	0.21	0.21	C
-	-	-	0.39	0.49	0.51	0.42	0.46	0.24	0.16	D
-	-	-	-	0.33	0.33	0.40	0.28	0.39	0.31	E
-	-	-	-	-	0.39	0.43	0.25	0.35	0.21	F
-	-	-	-	-	-	0.33	0.17	0.33	0.67	G
-	-	-	-	-	-	-	0.33	**1.00**	0.29	H
-	-	-	-	-	-	-	-	0.33	0.13	I
-	-	-	-	-	-	-	-	-	0.25	J

Table 7: Measures of symmetry (x-axis) of image pairs (test set 1 shown in Fig. 2).

A	B	C	D	E	F	G	H	I	J	
0.39	0.38	0.48	0.43	0.24	0.27	0.17	0.29	**0.88**	0.27	A
-	0.39	0.51	0.48	0.25	0.28	0.17	0.29	**0.89**	0.29	B
-	-	0.39	0.36	0.21	0.21	0.22	0.26	0.65	0.39	C
-	-	-	0.39	0.17	0.17	0.24	0.21	0.49	0.36	D
-	-	-	-	0.39	0.39	0.38	0.37	0.44	0.19	E
-	-	-	-	-	0.39	0.40	0.38	0.44	0.21	F
-	-	-	-	-	-	0.44	0.86	0.32	0.13	G
-	-	-	-	-	-	-	0.39	0.17	0.22	H
-	-	-	-	-	-	-	-	0.39	0.67	I
-	-	-	-	-	-	-	-	-	0.29	J

Table 8: Measures of symmetry (y-axis) of image pairs (test set 1 shown in Fig. 2).

A	B	C	D	E	F	G	H	I	J	
0.17	0.17	0.22	0.24	0.33	0.33	0.35	**0.85**	0.31	0.13	A
-	0.17	0.22	0.25	0.35	0.35	0.39	**0.89**	0.31	0.13	B
-	-	0.17	0.18	0.25	0.25	0.44	0.64	0.26	0.17	C
-	-	-	0.17	0.26	0.26	0.42	0.49	0.21	0.19	D
-	-	-	-	0.17	0.17	0.21	0.44	0.37	0.26	E
-	-	-	-	-	0.17	0.23	0.44	0.38	0.27	F
-	-	-	-	-	-	0.17	0.31	**0.80**	0.29	G
-	-	-	-	-	-	-	0.17	0.39	0.67	H
-	-	-	-	-	-	-	-	0.17	0.23	I
-	-	-	-	-	-	-	-	-	0.13	J

Table 9: Measures of symmetry (x- and y-axis) of image pairs (test set 1 shown in Fig. 2).

A	B	C	D	E	F	
0.38	0.38	0.55	0.34	0.30	0.34	A
-	0.38	0.52	0.34	0.30	0.34	B
-	-	0.38	**0.86**	0.67	0.83	C
-	-	-	0.25	0.25	0.28	D
-	-	-	-	0.25	0.25	E
-	-	-	-	-	0.31	F

Table 10: Measures of symmetry (x-axis) of image pairs (test set 2 shown in Fig. 3)

symmetry (x-axis, see C vs. F). The only symmetry observed here is mirroring, so the matrix presenting the x-axis symmetry is presented (see Table 10) along with the similarity measures array (see Table 6).

The discussion of the results is covered in following items pointing out properties of the proposed approach:

- **Translation and scale invariance** Thanks to the very first step of processing – conversion of image coordinates to relative ones, the method is scale and translation invariant. Both those transformation does not change the values of 2-D descriptors as long as the composition of objects within their group is the same. Examples confirming this property are the following[4]: 1A vs.1B Tab. 5 $sim = 0.97$; 2A vs.2B Tab. 6 $sim = 1.00$; 2D vs.2F Tab. 6 $sim = 1.00$.
- **Proportionality to the scale of changes**. The measure decreases when the number of objects

[4] To simplify explanation, in the references we will use shortcuts. For example, „1A vs.1B Tab. 5 $sim = 0.97$" means that image from the test scenes 1 (shown in Fig.3) „A" should be compared with an image shown on the position „B", the relevant measures are in Table 5, the similarity measure in this case is equal to 0.97. Images from the second set of scenes (shown in Fig.4) will be referred to as 2A, 2B, etc.

method	B,A	C,A	D,A	E,A	F,A	G,A	H,A	I,A	J,A
ours	0.97	0.76	0.53	0.44	0.44	0.33	0.17	0.39	0.72
MSE	0.08	0.04	0.07	0.12	0.1	0.11	0.11	0.12	0.08
PSNR	10.78	14.01	11.87	9.16	9.88	9.42	9.41	9.38	10.78
SSIM	0.73	0.88	0.78	0.62	0.66	0.65	0.63	0.63	0.73
M-SSIM	0.62	0.85	0.71	0.5	0.54	0.54	0.52	0.52	0.62

Table 11: Results of the proposed method ("ours") and standard similarity measures, similarity measures of image A with all other (test set 1 shown in Fig. 2)

method	B,A	C,A	D,A	E,A	F,A
ours	1	0.38	0.49	0.62	0.49
MSE	0.15	0.07	0.04	0.13	0.07
PSNR	8.36	11.77	14.05	8.88	11.56
SSIM	0.24	0.48	0.51	0.3	0.45
M-SSIM	0.16	0.47	0.56	0.2	0.39

Table 12: Results of the proposed method ("ours") and standard similarity measures, similarity measures of image A with all other (test set 2 shown in Fig. 3)

that changed their positions grows. The lower is thus the visual similarity (judged by a human observer), the lower is also the computed measure. Examples: 1A vs.1C (one replacement of objects) Tab. 5 $sim = 0.76$; 1A vs.1D (two replacements) Tab. 5 $sim = 0.53$; 1A vs.1H (completely different composition) Tab. 5 $sim = 0.17$; 2A vs. 2E (one shift) Tab. 6 $sim = 0.62$; 2A vs.2C (three shifts) Tab. 6 $sim = 0.38$.

- **Penalizing non-matched object** Thanks to the last step of processing, the normalization of accumulated pairwise similarity, the presence of objects on only one image reduces the similarity measure. Example: 1A vs.1J (two objects added, the remainder in the same composition) Tab. 5 $sim = 0.72$, to compare with 1A vs.1B.

- **Ability to detect symmetries** When choosing the appropriate matching matrices, the method is able to detect three reflectional symmetries. Examples, x-axis: 1A vs.1G Tab. 7 $sim = 0.85$; 1I vs.1H Tab. 7 $sim = 1$; 2C vs.2D Tab. 10 $sim = 0.86$; y-axis: 1A vs.1I Tab. 8 $sim = 0.88$; 1B vs.1I Tab. 8 $sim = 0.89$; xy-axis: 1A vs.1H Tab. 9 sim=0.85; 1B vs.1H Tab. 9 $sim = 0.89$; 1I vs.1G Tab. 9 $sim = 0.80$.

- **Axis of symmetry position's invariant symmetry measuring** Thanks to the relative way of describing the position of objects, the method is able to detect symmetries no matter where the symmetry axis is located. Example for x-axis: 1A vs.1G Tab. 7 $sim = 0.85$; 1B vs.1G Tab. 7 $sim = 0.89$.

The proposed approach was also experimentally compared to classic similarity measures: mean square error (MSE), peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [Zha93], and its multi-scale variant (M-SSIM) [Wan03]. Results are shown in Figs. 11 and 12. They show the values of measures computed with scene 1 and scene 2. In each case, the first scene (A) is compared with all others (B, C,...). Contrary to the proposed approach (first row), values of the remainder of measures do not follow the variations of the complexity of the composition of objects (bounding boxes). When looking for the visual similarity sensed by a human, one may introduce the following order of scenes starting from the most similar to scene A: for the set 1: B (scale), C (one replacement), J (two objects added), D (two replacements), E, F (four objects in the same mutual position), I (two groups of objects in similar mutual position), G, H (completely different composition) and for the scene 2: B (scale), E (one object shifted to the other side), D, F (in both one shifted to the other side, one shifted closed to the central one), C (two objects shifted). The proposed measure follows this order, while none of the classic ones does. What is more, considering the above ordering based on human perception, the ordering implied by classic measures seems to besomehow random.

6 CONCLUSIONS

The paper describes a novel method for measuring the similarity and symmetry of the annotated images' content consisting of objects defined by their bounding boxes. It allows comparing sets of bounding boxes to estimate the degree of similarity of their underlying images. It is based on the fuzzy approach that uses the fuzzy mutual positions matrix to describe spatial composition and relations between bounding boxes within a single image. In the paper, a method and algorithm for comparing such matrices computed for two images are proposed. It allows measuring the similarity of two images and outputs the single scalar value describing the degree of similarity. Modification of matching matrices used to establish correspondence between 2-D scene descriptors allows for applying the method to estimate the reflectional symmetries in the composition of the objects of the visual scenes.

The method has several valuable properties. It is translation and scale-invariant. The resulting measure is proportional to the degree of differences between images. It can detect symmetries no matter where the symmetry axis is located. It also has few parameters that allow its adjusting to particular purposes.

Moreover, it may be easily extended to get extra functionalities like, e.g., the ability to find the symmetric subsets of image objects. Last but not least, the method is fast and ready to use in real-time applications. It is due to the fact that it consists of simple operations: arithmetic operations, min/max, performed in nested loops but iterated with a relatively small number of times. It may be used in content-based image retrieval, semantic image analysis, and other computer vision fields.

REFERENCES

[Blo05] I. Bloch. "Fuzzy spatial relationships for image processing and interpretation: a review". In: *Image and Vision Computing* 23.2 (2005). Discrete Geometry for Computer Imagery, pp. 89–110. ISSN: 0262-8856.

[Coh01] Anthony Cohn and Shyamanta Hazarika. "Qualitative Spatial Representation and Reasoning: An Overview". In: *Fundam. Inform.* 46 (Apr. 2001), pp. 1–29.

[Den01] Y. Deng and B. S. Manjunath. "Unsupervised segmentation of color-texture regions in images and video". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.8 (2001), pp. 800–810.

[Gir15] R. Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[Iwa21] M. Iwanowski and M. Bartosiewicz. "Describing images using fuzzy mutual position matrix and saliency-based ordering of predicates". In: *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 2021.

[Kel00] J.M. Keller and X. Wang. "A Fuzzy Rule-Based Approach to Scene Description Involving Spatial Relationships". In: *Computer Vision and Image Understanding* 80.1 (2000), pp. 21–41. ISSN: 1077-3142.

[LeC15] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687.

[Liu07] Y. Liu et al. "A survey of content-based image retrieval with high-level semantics". In: *Pattern Recognition* 40.1 (2007), pp. 262–282. ISSN: 0031-3203.

[Loy06] G. Loy and J.-O. Eklundh. "Detecting Symmetry and Symmetric Constellations of Features". In: *Computer Vision – ECCV 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 508–521.

[Mat01] P. Matsakis et al. "Linguistic description of relative positions in images". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 31.4 (2001), pp. 573–588.

[Miy94] K. Miyajima and A. Ralescu. "Spatial organization in 2D images". In: *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*. 1994, 100–105 vol.1.

[OMa96] D. O'Mara and R. Owens. "Measuring bilateral symmetry in digital images". In: *Proceedings of Digital Processing Applications (TENCON '96)*. Vol. 1. 1996, 151–156 vol.1.

[Red16] J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[Ren17] S. Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149.

[Rod00] K. Rodden et al. *A Comparison of Measures for Visualising Image Similarity*. 2000.

[Sco03] R. Scognamillo et al. "A feature-based model of symmetry detection". In: *Proceedings. Biological sciences / The Royal Society* 270 (Sept. 2003), pp. 1727–33.

[Sme00] A. W. M. Smeulders et al. "Content-based image retrieval at the end of the early years". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.12 (2000), pp. 1349–1380.

[Sun99] C. Sun and D. Si. "Fast Reflectional Symmetry Detection Using Orientation Histograms". In: *Real-Time Imaging* 5.1 (1999), pp. 63–74. ISSN: 1077-2014.

[Wan03] Z. Wang, E.P. Simoncelli, and A.C. Bovik. "Multiscale structural similarity for image quality assessment". In: *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*. Vol. 2. 2003, 1398–1402 Vol.2.

[Wan04] Z. Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *Image Processing, IEEE Transactions on* 13 (May 2004), pp. 600–612.

[Xia07] Z. Xiao and J. Wu. "Analysis on Image Symmetry Detection Algorithms". In: *Proceedings - Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007*. Vol. 4. Jan. 2007, pp. 745–750.

[Zad65] L.A. Zadeh. "Fuzzy sets". In: *Information and Control* 8.3 (1965), pp. 338–353. ISSN: 0019-9958.

[Zha93] W. Zhang and M. Sugeno. "A fuzzy approach to scene understanding". In: *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*. 1993.

Multiple Object Tracking by Bounding Boxes Without Using Texture Information and Optical Flow

Ágnes Lipovits, László Czúni, Katalin Tömördi, Zsolt Vörösházi
Image Processing Laboratory
Faculty of Information Technology, University of Pannonia
Egyetem street 10.
Hungary (H-8200), Veszprém
lipovitsa@almos.uni-pannon.hu

ABSTRACT
Object tracking is a key task in many applications using video analytics. While there is a huge number of algorithms to track objects, there is still a need for new methods to solve the correspondence problem under certain circumstances. In our article, we assume a very typical but still open scenario: a still image object detector has already identified the objects to be tracked; thus, we have object labels, confidence values, and bounding boxes in each video frame captured at a low sampling rate. That is, optical flow methods difficult to be applied (also due to bad lighting conditions, cluttered or homogeneous areas and strong ego-motion), and moreover, many objects look similar (having the same category labels). Our proposed approach is based on the Hungarian method and incorporates the above information into the cost function evaluating the possible pairings of objects. To consider the uncertainty of the detector, the elements of the confusion matrix also contribute to the cost of pairs, as well as the probability of spatial translations based on prior observations. As a use case, we apply the algorithm to a data-set, where images were captured from onboard cameras and traffic signs were detected by RetinaNet. We analyze the performance with different parameter settings.

Keywords
Object tracking, object detection, Hungarian method, RetinaNet

1 INTRODUCTION

While the goal of all object tracking algorithms is to estimate the trajectory of moving objects, there are significant differences between their nature, considering their prerequisites and tolerance against the different visual condition. For example, while many algorithms heavily rely on the optical flow or velocity and find the corresponding areas using the analysis of image features of candidate regions (e.g. [Bewley2016], [Ma2019]), others try to involve texture-based object detection more deeply (for example [Ning2017] and [Wang2019]). When the camera is moving in a 3D space, and the image features of the relevant objects to be tracked are similar, multiple object tracking becomes very difficult and even can require efforts to neutralize camera motion [Czúni2012]. That is a reason that under such circumstances, the second approach seems much more appealing.

In our article, we process outdoor videos captured from onboard cameras, and different objects, mostly traffic signs, are to be detected and tracked. Besides the problems encountered from various factors, such as the cluttered environment, possible occlusions, scale variation, image noise, and low contrast, the temporal sampling rate may be too low in the case of high-speed ego-motion. Moreover, the same object classes often have multiple appearances; thus, optical flow methods (e.g. CAMShift, Kanade-Lucas-Tomasi, Horn-Schunck, etc.) can not be applied efficiently. Fig. 1 illustrates when the car is turning, and the optical flow (generated by [Farneback2003]) looks turbulent due to the rotational and translational motion of the camera, the complicated structure of the 3D scene, low contrast regions, and camera independent object motion.

We assume a very typical but still open processing pipeline scenario for object tracking: a still image object detector has already identified the objects to be tracked; thus, we have object labels, confidence values, and bounding boxes in each video frame captured at a given sampling rate. The task is to find the correspondence between the bounding boxes (or declare there is no such match).

Our proposed approach is based on the Hungarian method [Kuhn1955] and incorporates the above information into the cost function evaluating the possible pairings of objects. To consider the uncertainty of the detection, the elements of the confusion matrix (error matrix) of the detector also contribute to the cost of pairs, as well as the probability of spatial translations based on prior observations.

In the next section, we overview related papers, then

Figure 1: Optical flow computed by [Farneback2003] when the car is turning.

in Section 3 we describe the data-set and the object detection method. Section 4 contains the theoretic details of our proposal, while in Section 5 we analyze the performance with different parameter settings.

2 ABOUT TRACKING METHODS

There are many aspects to categorize the large number of approaches such as stereo or monocular, model-based or model-free, multiple target or single-target, casual or bi-directional, and long-term or short-term trackers.

Since there is a larger variety and number of tracking algorithms, we can not give a comprehensive overview in our short article. We focus on those found the most similar or the recent ones lacking optical flow calculations but using neural networks.

[Huang2008] describes a three-level hierarchical tracking method for multiple objects: at low-level, short tracks are generated for further analysis; at the middle-level, these tracks are further processed to form longer trajectories based on the Hungarian method; while at the highest level, a scene structure model (including three maps for entries, exits and scene occluders) is created. This high-level step implements scene knowledge-based reasoning to reduce trajectory fragmentation and prevent possible identity switches. The method only uses colour histograms, position and size, no other information from the detector itself.

[Henriques2011] also applies the Hungarian method but focuses on the ability to model multiple objects that are merged into a single measurement and track them as a group. The solution is based on a graph structure that encodes these multiple-match events. Since object identities are lost when objects merge, the problem of tracking individual objects across groups is posed as a standard optimal assignment problem.

A good recent example to combine different modalities in the tracking-by-detection domain is [Karunasekera2019], where a dissimilarity measure based on object motion, appearance (colour histogram), structure (Local Binary Pattern), and size was used; assignment is solved by the Hungarian method.

In contrast to [Karunasekera2019], in [Bewley2016] an efficient, while relatively simple and lightweight tracker, for multiple objects with constant velocity, was described using variations of Faster Region CNNs, and the plain old Kalman-filter and the Hungarian method. The main idea was to use the very efficient CNNs for the detection of objects, instead of using hand-crafted visual features of classical appearance-based trackers. Not surprisingly, it was found that the detection quality had a significant impact on tracking performance. Our proposal is also a simple tracker, but we incorporate the properties of detected objects (namely the confidence and the probability of confusions) into the solution of the correspondence problem. Moreover, since our tracker handles velocity in a probabilistic manner, linear or constant motion is not assumed.

There are approaches where both the spatial and temporal domains of tracking are fused by deep neural networks. For example [Ning2017] proposes a method to extend the deep neural network learning and analysis into the spatio-temporal domain by combining Yolo and LSTM networks. The spatially and temporally deep method applies regression and can effectively tackle problems of occlusions and motion blur. [Jiang2018] also applies Yolo and LSTMs but in a very different way. Each object has its own tracker, regarded as an agent, trained by utilizing deep reinforcement learning, where LSTM is used to predict parameters (motion and scale) of the observed object. Data association between the output of Yolo and the trackers is also done by an LSTM. The drawback of this technique is that besides Yolo the LSTMs should also be trained according to the environments. [Wang2019] introduces SiamMask performing both real-time visual object tracking and semi-supervised video object segmentation. Once trained, it relies on a single bounding box initialization and operates online, and can produce object segmentation masks and rotated bounding boxes at high-speed. Unfortunately, both methods can handle only a single object at a time. While most approaches follow a sequential strategy as detection then tracking, some new DNNs try to fuse the two steps. For example in [WangZ2019] a Feature Pyramid Network is used to find objects with their bounding boxes with a constraint that the distance between observations of the same identity in consecutive frames should be smaller than the distance between different identities. Unfortunately, this does not hold in many cases for our videos.

For readers with more interest to overview this field we propose to check earlier papers [Cannons2008],

[Smeulders2013] or [Fiaz2019] published more recently.
Our proposed solution will be a monocular, model-free, multiple target, causal, and short-term tracker. What is more important that we are not applying optical flow (only accumulated statistics about the motion of the camera), can utilize any object detection mechanism, which produces bounding boxes and confidence values, making our algorithm well-suited, as a post processing step, in many image processing pipelines.

3 THE BENCHMARK DATA-SET

Automotive applications require fast and accurate detection of street objects, traffic signs are among the key elements. While the detection and classification of traffic signs on some existing data-sets can reach good performance [Lim2017], there are many circumstances where results are still poor [Temel2019]. Also the number of classes is not really limited since there are many composite objects or unique designs which make existing methods easily fail.

3.1 The Hun158 and Hun169 Data-sets

The Hun158 benchmark data-set contains 158 different classes of Hungarian road traffic signs and some typical street objects (e.g. bus stops, dust-bins, etc.). It comprises pixel-wise segmented objects on 3440 image frames of size 1280×720 and 1920×1080 resolutions. The number of annotated objects is 13300, each has at least 20 appearances. The Hun169 data-set contains 20 videos of 37462 frames with 61274 annotated objects from 168 classes. In Hun169 objects are denoted by bounding boxes, videos were recorded at 15 and 30 FPS. There is no overlapping between Hun158 and Hun169.

3.2 Object Detection

On all the frames of Hun158 we computed the bounding boxes and trained the RetinaNet [Lin2017]. See Fig. 2 for the illustration of two subsequent frames with detected objects.

RetinaNet [Lin2017], as a popular dense detector network, is one of the best single-stage object detection model that has been proven to work efficiently with dense and small scale objects. The main new features of RetinaNet were the application of Feature Pyramid Networks and Focal Loss. By these enhancements it could reach the performance of previous single-stage detectors (e.g. Yolo, SSD) while it exceeds the accuracy of the existing state-of-the-art two-stage detectors

Figure 2: Typical results of the detection on two consecutive frames.

(R-CNN variants). Beside the Feature Pyramid Network it has two subnetworks: one for regression of the precise allocation of the bounding boxes, and the other one for classification (labeling). We used VGG19 as the backbone of RetinaNet.
Testing on the Hun169 we could reach about 0.805mAP (considering only objects of size larger than 10 pixels in any dimension). Most of the errors came from small objects and detections of such traffic signs which were missing from the trained classes (appearing as FP - false positives).
On Fig. 3 our large sized confusion (158×158) matrix (CM) with a magnified part can be seen to allow visualization of the performance of the detector. Each column of the matrix represents the instances in an actual class (for identifying the traffic road signs) while each row represents the instances in a predicted class by detectors. Black colors denote values zero or near to zero, while lighter colors in the main diagonal of matrix denote a large number of instances. The values are the averaged values of the aggregated matrix which was generated from 18 confusion matrices and derived from testing of the 18 videos.

3.3 Data-set for Tracking

For multi-object tracking purposes we used 20 videos from Hun169; Table 1 summarizes information about the training and testing parts (objects larger than 10 pixels were filtered out). During training of the tracking algorithm, statistical information was gathered about the size and position of the objects in the Ground Truth

(GT annotation) data and the probabilities of possible confusions of the detector ran antecedently.

Videos (@15 or 30 FPS)	training	testing
Number of videos	18	2
Names of videos HUN_	V01 - V18	V19, V20
Length of video (min:sec)	22:59	6:30
Number of frames	20703	11741
Number of objects	728	361
Number of bounding boxes	31455	20416

Table 1: Training and testing data-set for tracking.

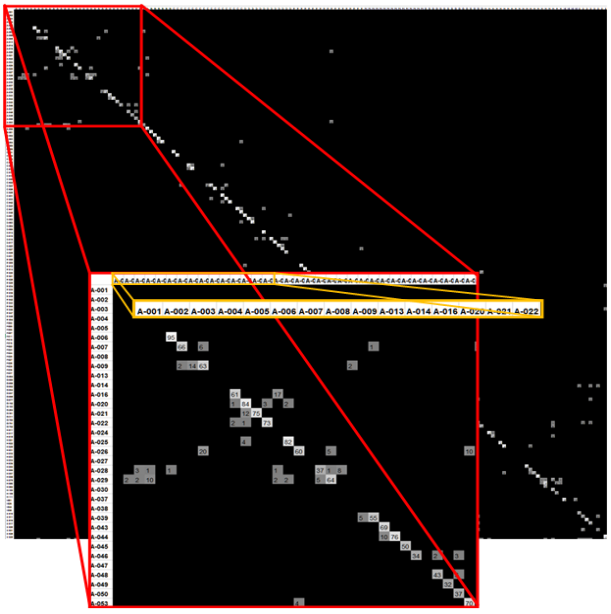


Figure 3: Illustration of the confusion matrix of the applied object detector at confidence threshold level 0.5 based on 18 videos of the Hun158 data-set.

4 TRACKING OF BOUNDING BOXES

We employ the Hungarian method to find the optimal correspondence between detected objects of two frames using the following cost function. Let us define the cost of pairing detections i and j from frame k and $k+1$:

$$\begin{aligned} C_{i,j} = \alpha(1 - CM(\mathscr{L}(d_{k,i}), \mathscr{L}(d_{k+1,j}))) \\ + \beta \frac{\phi_{\Delta A}(\Delta A)}{\max(\phi_{\Delta A})} + \gamma \frac{\phi_{\Delta y}(\Delta y)}{\max(\phi_{\Delta y})} \\ + \delta\,(1 - p_i p_j)\,, \end{aligned} \tag{1}$$

where $\mathscr{L}(d_{k,i})$ is the class label of the i^{th} detection on the k^{th} frame, CM is the confusion matrix (for illustration see Fig. 3), ΔA and Δy are the differences between bounding boxes' area and y coordinates in consecutive frames, $\phi_{\Delta A}$ and $\phi_{\Delta y}$ are the corresponding probability density function assuming normal distributions, and p_i is the confidence value of the i^{th} detection. All terms are between 0 and 1, grid-search is used to find their optimal settings, resulting in the best MOTA (Multiple Object Tracking Accuracy) [Stiefelhagen2006] value. The algorithm used for multiple object tracking is given in Algorithm 1.

Algorithm 1: Multiple Bounding Box Tracking algorithm

Input: $D = \{D_0, D_1, \cdots, D_{F-1}\}$
Initialize: $AT = \emptyset, FT = \emptyset$
for $f = 0$ *to* $F-1$ **do**
  while $at \in AT$ **do**
    $t = at_{last}$
    $d = Hun\_Method(AT_{last}, D_f, t), d \in D_f$
    if $d < Th$ **then**
      push d to at
      remove d from D_f
    else
      add at to FT
      remove at from AT
  while $d \in D_f$ **do**
    start new track list with d and insert into AT
while $at \in AT$ **do**
  add at to FT
Result: FT

The proposed algorithm requires a training phase to determine its main parameters, however, these parameters are easy to obtain (and are based on a larger set of image sequences as given in the next section). Since no motion flow is computed between the consecutive frames, we can rely purely on information given by the detector (coordinates of bounding boxes, labels, confusion matrix, and the detection's confidence values). The proposed algorithm has the following parameters:

- F: the number of frames in the sequence,
- D_f: list of detections on the frame f, $0 \leq f < F$,
- D: list of D_i, $0 \leq i < F$,
- AT: active tracks; $at \in AT$,
- at_{last}: last element of AT,
- FT: finished tracks,
- Th: threshold of the costs.

5 EXPERIMENTS

Demonstrating the usability of our algorithm, we used the benchmark data-set described in Section 3.3. Normalized histograms of the area changes and y-direction displacements of the 37163 bounding boxes of the 18 training videos (see Table 1) were computed to fit the normal distributions, as shown in Fig. 4, and in Table 2.

To measure the accuracy we use MOTA [Stiefelhagen2006], which is a widely used met-

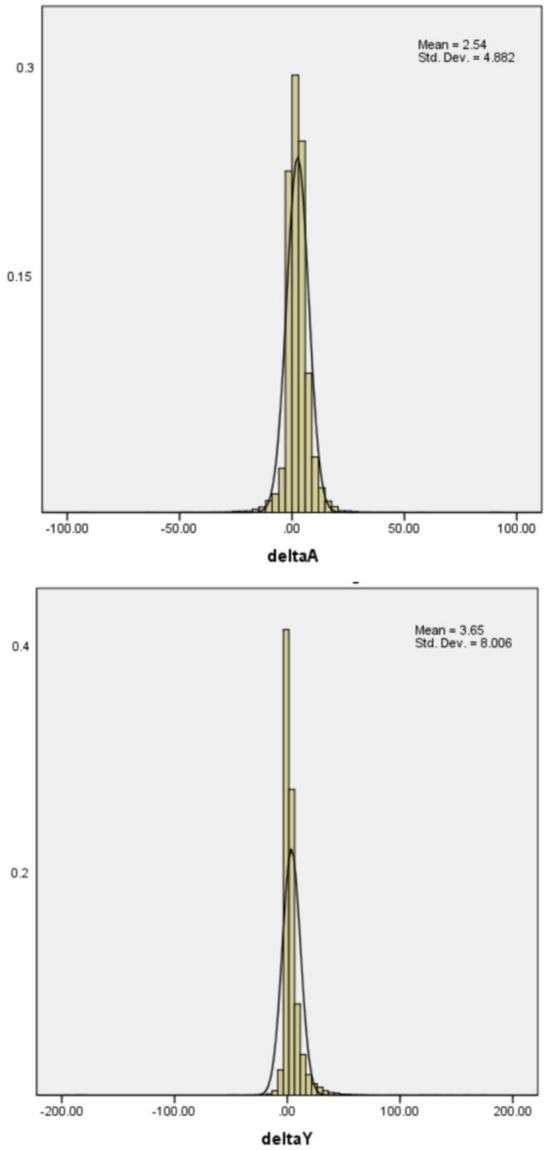


Figure 4: Normal distributions and their parameters fitted to the ΔA's and Δy's normalized histograms, measured on 18 training videos.

ric to evaluate the performance of multiple object tracking algorithms. It combines three sources of errors as FP (false positive), FN (false negative), and $IDSW$ (identity switch):

$$MOTA = 1 - \frac{\sum_k (FN_k + FP_k + IDSW_k)}{\sum_k GT_k}, \qquad (2)$$

where GT_k denotes the number of Ground Truth objects at frame k. The grid-based parameter search range is from 0 to 1 for all four parameters, using a step size of 0.2. The average MOTA value of the test videos was maximal at parameter values $\alpha = 1, \beta = 0.6, \gamma = 0.6$, and $\delta = 0.2$. The evaluation of tracking on videos shows that changes in parameters influence tracking outcomes, but the standard deviation is relatively small

	$\phi_{\Delta A}$	$\phi_{\Delta y}$
mean	2.5363	3.6472
standard deviation	4.88180	8.00638
max	0.2953	0.4168

Table 2: Parameters estimated on the 18 training sequences.

as seen in Table 3 above. The standard deviation is calculated at the parameters for the best MOTA values per video over the entire grid. Analyzing the diagrams in Fig. 5, we note that δ should be set to a relatively low value to reach high MOTA, thus the confidence value of detection should not be given much weight. The method is less sensitive to changes in α, β and γ.

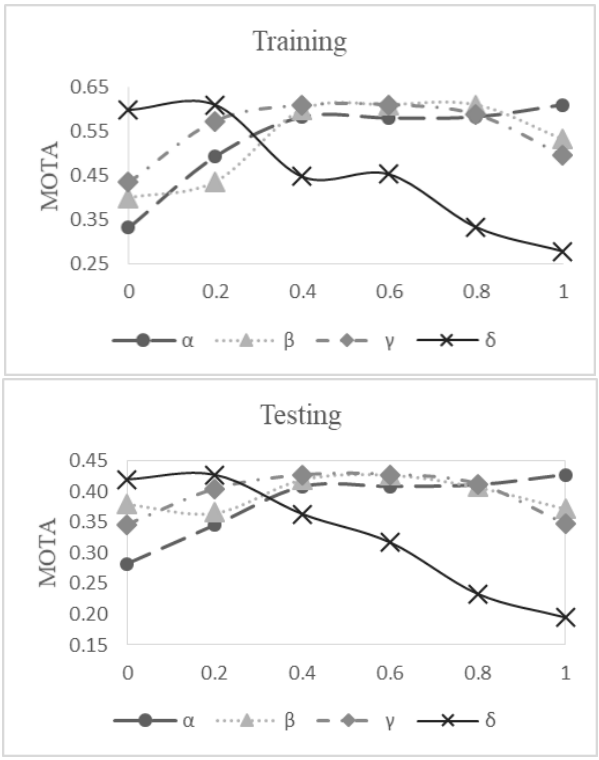


Figure 5: Parameter tests for MOTA with three fixed values. ($\alpha = 1, \beta = 0.6\,\gamma = 0.6, \delta = 0.2$)

6 CONCLUSIONS AND FUTURE WORK

In our article we described a tracking-by-detection method based on the outputs of an arbitrary object detector. Instead of using explicit textural features we rely on the confusion matrix of the detector to involve the possible similarity in appearance (which is very common for traffic signs). Motion is represented by the distribution of the vertical motion component of the detected bounding boxes accumulated in a training phase, thus no optical flow is needed, and there can be large changes in coordinates. This fits well to situations, where there is non-constant, often large

Video	Recall	Precision	F-measure	mAP	MOTA	Std. dev. (MOTA)
HUN_V01	0.8339	0.6658	0.7404	0.854	0.6728	0.0921
HUN_V02	0.6950	0.8487	0.7642	0.714	0.6595	0.1345
HUN_V03	0.7120	0.6758	0.6935	0.705	0.6754	0.0905
HUN_V04	0.8683	0.7694	0.8158	0.885	0.6133	0.1421
HUN_V05	0.7460	0.6396	0.6887	0.772	0.5956	0.0920
HUN_V06	0.9075	0.7762	0.8367	0.906	0.7223	0.0555
HUN_V07	0.7465	0.8419	0.7913	0.833	0.5440	0.0571
HUN_V08	0.7615	0.8022	0.7813	0.812	0.5922	0.1261
HUN_V09	0.8403	0.7368	0.7852	0.817	0.7248	0.1103
HUN_V10	0.9845	0.7737	0.8665	0.992	0.6155	0.0975
HUN_V11	0.4876	0.7269	0.5837	0.547	0.4201	0.0494
HUN_V12	0.7500	0.6970	0.7225	0.792	0.6624	0.1082
HUN_V13	0.5902	0.7451	0.6586	0.704	0.5561	0.0979
HUN_V14	0.7456	0.8200	0.7810	0.826	0.5481	0.1234
HUN_V15	0.7594	0.7988	0.7786	0.818	0.5465	0.0823
HUN_V16	0.6550	0.7042	0.6787	0.690	0.4671	0.1587
HUN_V17	0.9502	0.7744	0.8534	0.952	0.7778	0.1041
HUN_V18	0.8441	0.8169	0.8303	0.885	0.6032	0.1928
HUN_V19	0.4483	0.6274	0.5229	0.651	0.4143	0.1472
HUN_V20	0.4553	0.7078	0.5541	0.617	0.4404	0.0616

Table 3: The results of detection (at threshold 0.5) and tracking for 18 training and 2 test videos ($\alpha = 1$, $\beta = 0.6$, $\gamma = 0.6$, $\delta = 0.2$).

velocity and the camera has strong ego-motion (e.g. on-board car cameras).

We tested the proposed approach on a data-set of large number of object classes, sometimes with strong similarity. RetinaNet is a good candidate detector for such tasks. The complexity of tracking is very low, since the proposed cost function contains simple functions of pre-computed variables.

As future work we plan to compare it to other methods for speed and accuracy.

7 ACKNOWLEDGMENTS

Our special thanks to Dániel Zajzon for the object detection algorithms and to Nándor Szollát for data processing. We are grateful to the NVIDIA corporation for supporting our research with GPUs obtained by the NVIDIA GPU Grant Program. We acknowledge the financial support of the project EFOP-3.6.1-16-2016-00015 under the Széchenyi 2020 program, 2020-4.1.1-TKP2020 project under the Thematic Excellence Program, and grant GINOP-2.2.1-15-2017-00058.

8 REFERENCES

[Lim2017] Lim, K., Hong, Y., Choi, Y., Byun, H. (2017). Real-time traffic sign recognition based on a general purpose GPU and deep-learning. PLoS one, 12(3), e0173317.

[Temel2019] Temel, D., Alshawi, T., Chen, M. H., AlRegib, G. (2019). Challenging environments for traffic sign detection: Reliability assessment under inclement conditions. arXiv preprint arXiv:1902.06857.

[Lin2017] Lin, T. Y., Goyal, P., Girshick, R., He, K., Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

[VIA] VGG Image Annotator (VIA): https://www.robots.ox.ac.uk/ vgg/software/via/

[DarkLabel] Dark Label Image Annotation Tool: https://darkpgmr.tistory.com/16

[RouteShoot] RouteShoot georeferenced video recording application: https://www.wilsonpymmay.co.uk/

[Czúni2012] Czúni, L., Gál, M. (2012, September). Directional votes of optical flow projections for independent motion detection. In International Conference on Computer Vision and Graphics (pp. 329-336). Springer, Berlin, Heidelberg.

[Farneback2003] Farneback, Gunnar. "Two-frame motion estimation based on polynomial expansion." Scandinavian conference on Image analysis. Springer, Berlin, Heidelberg, (2003).

[Fiaz2019] Fiaz, M., Mahmood, A., Javed, S., Jung, S. K. (2019). Handcrafted and deep trackers: Recent visual object tracking approaches and trends.

ACM Computing Surveys (CSUR), 52(2), 1-44.

[Milan2016] Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831.

[Cannons2008] Cannons, K. (2008). A review of visual tracking. Dept. Comput. Sci. Eng., York Univ., Toronto, Canada, Tech. Rep. CSE-2008-07, 242.

[Smeulders2013] Smeulders, A. W., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., Shah, M. (2013). Visual tracking: An experimental survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(7), 1442-1468.

[Ning2017] Ning, G., Zhang, Z., Huang, C., Ren, X., Wang, H., Cai, C., He, Z. (2017, May). Spatially supervised recurrent convolutional neural networks for visual object tracking. In 2017 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-4). IEEE.

[Wang2019] Wang, Q., Zhang, L., Bertinetto, L., Hu, W., Torr, P. H. (2019). Fast online object tracking and segmentation: A unifying approach. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 1328-1338).

[Huang2008] Huang, C., Wu, B., Nevatia, R. (2008, October). Robust object tracking by hierarchical association of detection responses. In European Conference on Computer Vision (pp. 788-801). Springer, Berlin, Heidelberg.

[Bewley2016] Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B. (2016, September). Simple online and realtime tracking. In 2016 IEEE international conference on image processing (ICIP) (pp. 3464-3468). IEEE.

[Karunasekera2019] Karunasekera, H., Wang, H., Zhang, H. (2019). Multiple object tracking with attention to appearance, structure, motion and size. IEEE Access, 7, 104423-104434.

[Henriques2011] Henriques, J. F., Caseiro, R., Batista, J. (2011, November). Globally optimal solution to multi-object tracking with merged measurements. In 2011 International Conference on Computer Vision (pp. 2470-2477). IEEE.

[Weng2020] Weng, X., Wang, J., Held, D., Kitani, K. (2020). 3d multi-object tracking: A baseline and new evaluation metrics. arXiv preprint arXiv:1907.03961.

[Stiefelhagen2006] R. Stiefelhagen, K. Bernardin, R. Bowers, J. S. Garofolo, D. Mostefa, and P. Soundararajan. The CLEAR 2006 evaluation. In CLEAR, 2006.

[Kuhn1955] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval research logistics quarterly, 2(1-2), 83-97.

[Ma2019] Ma, Y. (2019). An object tracking algorithm based on optical flow and temporal-spatial context. Cluster Computing, 22(3), 5739-5747.

[Jiang2018] Jiang, M. X., Deng, C., Pan, Z. G., Wang, L. F., Sun, X. (2018). Multiobject tracking in videos based on LSTM and deep reinforcement learning. Complexity, 2018.

[WangZ2019] Wang, Z., Zheng, L., Liu, Y., Wang, S. (2019). Towards real-time multi-object tracking. arXiv preprint arXiv:1909.12605, 2(3), 4.

View Synthesis: LiDAR Camera versus Depth Estimation

Yupeng XIE, Sarah FACHADA, Daniele BONATTO, Mehrdad TERATANI, Gauthier LAFRUIT
Université Libre de Bruxelles
LISA department
Av. F.D. Roosevelt 50 CP165/57
Belgium, 1050 Brussels
Yupeng.Xie@ulb.be; Sarah.Fernandes.Pinto.Fachada@ulb.ac.be; Daniele.Bonatto@ulb.ac.be;
Mehrdad.Teratani@ulb.ac.be; Gauthier.Lafruit@ulb.ac.be

Abstract

Depth-Image-Based Rendering (DIBR) can synthesize a virtual view image from a set of multiview images and corresponding depth maps. However, this requires an accurate depth map estimation that incurs a high computational cost over several minutes per frame in DERS (MPEG-I's Depth Estimation Reference Software) even by using a high-class computer. LiDAR cameras can thus be an alternative solution to DERS in real-time DIBR applications. We compare the quality of a low-cost LiDAR camera, the Intel Realsense LiDAR L515 calibrated and configured adequately, with DERS using MPEG-I's Reference View Synthesizer (RVS). In IV-PSNR, the LiDAR camera reaches 32.2dB view synthesis quality with a 15cm camera baseline and 40.3dB with a 2cm baseline. Though DERS outperforms the LiDAR camera with 4.2dB, the latter provides a better quality-performance trade-off. However, visual inspection demonstrates that LiDAR's virtual views have even slightly higher quality than with DERS in most tested low-texture scene areas, except for object borders. Overall, we highly recommend using LiDAR cameras over advanced depth estimation methods (like DERS) in real-time DIBR applications. Nevertheless, this requires delicate calibration with multiple tools further exposed in the paper.

Keywords

View Synthesis, Depth Estimation, LiDAR, Camera Calibration, DERS, DIBR, RVS

1 INTRODUCTION

Depth-image-Based-Rendering (DIBR) technology [7] is widely used in end-to-end immersive autostereoscopy, promoting the continuous progress of 3D computer vision applications [13] [14]. It uses multiviews and their associated depth maps to synthesize a realistic virtual view. MPEG-I (The Moving Picture Expert Group Immersive) has specially introduced its DIBR-based Reference View Synthesizer (RVS) [11], which can support view synthesis in real-time with a large baseline. However, as DIBR-based, RVS performance is highly dependent on its input depth maps quality.

DERS [15] can estimate high accuracy depth maps but with a high computational cost due the complexity of its algorithm, which remains a significant challenge for the real-time application purpose. Additionally, low texture regions of the image significantly reduce the algorithm's performance.

LiDAR-based RGB-D cameras currently play a significant role in the research field of computer vision [17]. Their high accuracy of depth map acquiring with the low computational cost promotes the development of real-time 3D applications [5]. Hence, using an RGB-D camera, such as the Intel Realsense LiDAR L515 (hereafter LiDAR), can be considered an alternative solution to DERS in real-time view synthesis applications.

Nonetheless, LiDAR's depth maps are difficult to evaluate due to the depth sensor's limitations to capture non-reflective colors, absorbing materials, and objects with light deflecting shapes [9]. Moreover, camera calibration and depth registration are required because the captured depth map and its associated color image have different resolutions and misalignment due to different sensors' (RGB and Depth) positions.

This paper proposes a method to evaluate the LiDAR camera's performance. Instead of assessing its depth maps accuracy directly, we evaluate the quality of the virtual view synthesized by RVS with depth maps and their corresponding color images of the LiDAR, which is precisely calibrated. We compare the quality of virtual views, synthesized using the depth maps of LiDAR, and the estimated one by DERS, respectively. In this

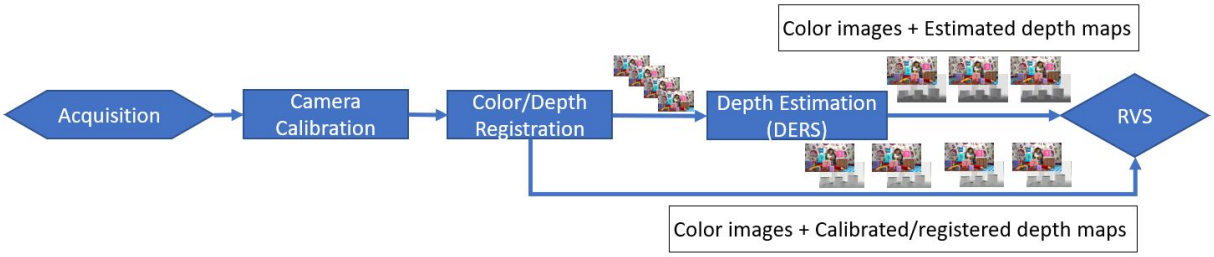


Figure 1: Processing pipeline

comparison, we used the IV-PSNR virtual reality quality assessment metrics to evaluate their performance gap. Based on the evaluation results, we has observed the substantial trade-off between view synthesis performance using RVS, given the depth map acquired by LiDAR and estimated depth map by DERS. The processing pipeline is illustrated in Figure 1, which details are explained in section 2 and 3.

2 OVERVIEW OF THE PROCESSING PIPELINE

This section explains the processing pipeline illustrated in Figure 1. We also give a brief introduction to DERS, RVS, and our assessment measure IV-PSNR. The motivation and detailed procedure for camera calibration and depth registration are provided separately in section 3.

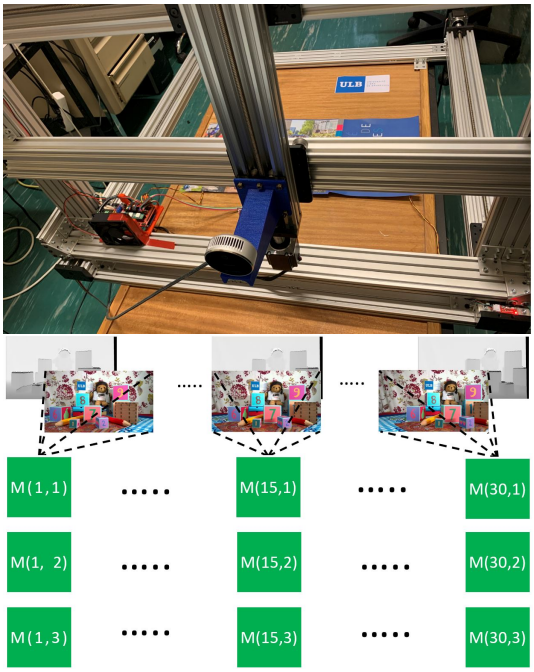


Figure 2: Acquisition system and 2D multiview configuration.

Acquisition

Before acquiring the test sequence for our evaluation of depth map quality by DERS and LiDAR for view synthesis, we have conducted a precise camera calibration, which is one of the contributions to this paper (section 3). Once the LiDAR was calibrated, we have mounted it on our acquisition robot to acquire a 2D dataset of 30x3 multiview color images and simultaneously registered their depth maps in real-time (Figure 2). The color images are used in DERS for estimation of depth maps and RVS for view synthesis.

Depth Estimation Reference Software (DERS)

DERS is one of the state of art high quality depth estimation software that have been promoted as reference software in MPEG-I. The latest version is 9.0 [12]. The main process of DERS is to estimate one depth map from a sparse setup of multiple reference color images. The algorithm includes two fundamentals steps.

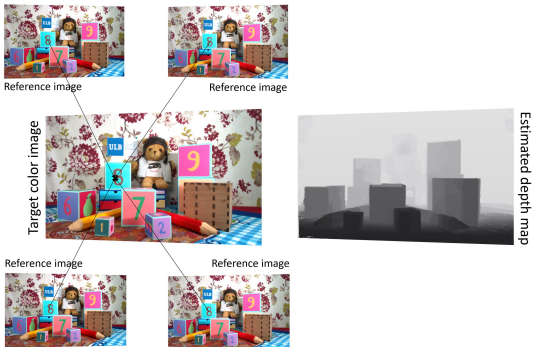


Figure 3: Estimation of depth map using five reference views.

(a) Calculate matching cost cube: DERS works in depth or disparity estimation mode. For the sake of simplicity, we consider the procedure to estimate disparities. The disparities can then be transformed into depth maps using [1]. Disparity estimation is performed using pair of images: The reference and the evaluation image. Both images are registered in a preprocessing step. Each pixel of the reference image corresponds to only

an unknown pixel in the evaluation image. DERS performs a Sum of Absolute Differences (SAD) between a patch around the pixel in the reference image and all the patches centered on the corresponding epipolar line in the evaluation image to find matching candidates.

This procedure gives rise for each pixel of the reference image to a cost function on all the possible disparities along the epipolar line. Each cost function per pixel is then stored in a cost cube with dimensions width $\times$ height $\times$ cost. With the cost value varying between the minimum disparity and the maximum disparity. This procedure is performed between all images and the reference image, and the resulting cost volumes are merged and stored in a matching cost cube.

(b) Graph cut global optimisation: Selecting the best cost for each pixel in the cost cube results in noisy depth maps. Therefore, DERS performs a global optimization technique known as Graph-cut [3] [4] to obtain the cost cube's optimal values. Furthermore, a smoothing map is used to increase or decrease the cost of a cut between the image's pixels to consider if two pixels are on the same object. This increases the optimization quality by forcing close-by pixels to have similar depth values.

Figure 3 demonstrates one example of the DERS depth map, which is estimated by using five reference (including the target one in the center) color images. Subjectively, the depth map's quality is clean and sharp without outliers. However, it is not accurate to conclude before evaluating the quality of the virtual view synthesized by RVS using the estimated depth map.

Reference View Synthesizer (RVS)

RVS and it's real time extension RaVIS [2] is a DIBR-based software developed in the context of MPEG-I standardization activities. It has been designed to take any number of reference images, with corresponding depth maps, in any configuration and renders outputs by interpolation and extrapolation. Using several reference images makes it more resistant to DIBR artifacts such as ghosting (due to poor calibration) and disocclusions (due to missing information in the input images).

IV-PSNR

For evaluating quality metrics of RVS output, we have used IV-PSNR [6] which is a PSNR-based quality metric, which defined by equation (1) and (2). IV-PSNR takes YUV [16] (Y defines the luma component and two chrominance components U-blue projection and V-red projection) file as its input. On YUV images, the IV-PSNR is the weighted mean on each component:, where MAX is the maximum possible pixel value. Similarly to the MSE, $IVMSE$ is the mean of the squared error IVE for each pixel p of the virtual view, where D is a correction term taking into account the global color difference between the virtual and reference image.

$$IVPSNR_{yuv} = 10 \times log\left(\frac{MAX^2}{IVMSE}\right) \tag{1}$$

$$IVMSE = \sum_{y=0}^{H}\sum_{x=0}^{W} Min_{p_{R(x,y)}\in\Omega}\frac{(p_V(x,y) - p_R(x,y) + D)}{WH} \tag{2}$$

Unlike classic PNSR, IV-PSNR considers a patch of adjacent pixel quality metric evaluation instead of each single pixel. In this respect, it less considers the corresponding pixel shift in the objects' edges and is insensitive to the global color's difference between the ground truth and virtual view, which is suitable for immersive video quality metric evaluation. Please refer to the reference [6] for more details.

3 LIDAR CALIBRATION AND REGISTRATION

This section presents one of this paper's main contributions to explain how to use LiDAR correctly. Multiple camera calibration is requisite for adequately using the LiDAR camera to capture test sequence or view synthesis. The depth maps need to be aligned to their corresponding color image since they are captured separately from two sensors Figure 4.

The performance of such registration relies on the sensor's intrinsic and extrinsic parameters accuracy. However, the default parameters provided by the Intel SDK are too coarse to conform RVS's particular requirement. Hence, we have used Kalibr [8] [10] to calibrate the LiDAR camera for getting more robust parameters of sensors.

Calibration: Kalibr is a conventional calibration software that supports multiple camera calibration with a non-global field of view (do not restrict entire calibration target captured in each sensor). Its implementation is relatively straightforward and outputs a detailed calibration statement to help users understand the calibration accuracy.

Figure 4: Intel L515 camera layout

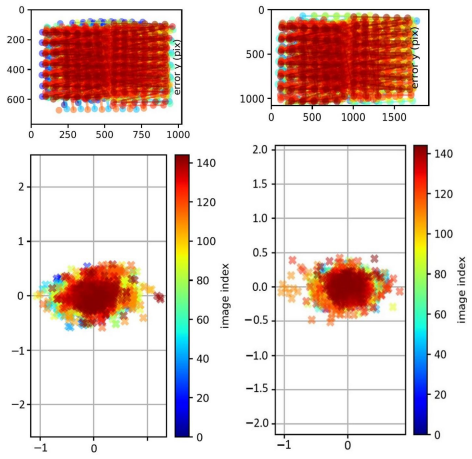


Figure 5: Calibration accuracy: (a) Distribution of detected pixel of the depth sensor and the interval of its related reprojection error.(b) Distribution of detected pixel of the color sensor and the interval of its related reprojection error.

We use a calibration pattern that is attached on a flat glass. This is important to achieve the high accuracy of calibration. According to the Kalibr's output Figure 5 (a and b), the LiDAR's depth sensor reprojection error is reduced to under ± 0.5 pixel for the color sensor and ± 1 pixel for the depth sensor. Compared with the uncalibrated registered depth maps (hereafter LiDAR-UCRD), the calibrated registered depth maps (hereafter LiDAR-CRD) have significantly impacted the view synthesis's quality (detailed discussion are in section 4).

Depth Map Registration: We have performed the depth registration process based on all calibration parameters. Nevertheless, it still has inevitable imperfections (Figure 6) even though camera calibration was sufficiently precise.

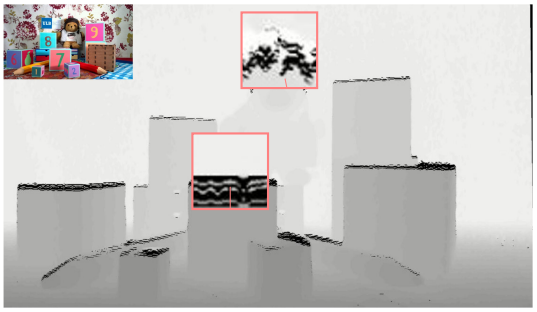


Figure 6: LiDAR registered depth map. (**a**) Error area above the cube. (**b**) Error area at the top of bear's head

In Figure 6 (**a**), the black pixels are the typical artifacts in registered depths. Depth sensor has a lower resolution than the color sensor, which cause some missing pixels in calibrated and registered LiDAR (LiDAR-CRD). Moreover, the color depth is located on top of the depth sensor. When projecting back depth map corresponding 3D points to the color sensor, some pixels information are missing since these specific pixels occluded in the depth sensor view.

In Figure 6 (**b**), the blank depth area in front of the bear's hat comes from the material's absorption of the light. LiDAR does not receive any reflected of a light ray in this area, which leads to invalid depth information.

Having LiDAR calibrated and registered, we can use its depth map to synthesize virtual views by RVS.

4 EXPERIMENTS

We have used two different RVS (Figure 7 and Table 1). Figure 7(a) demonstrates using the four reference images plus corresponding depth maps from 4 corners (i.e., $M_r(1,1)$, $M_r(30,1)$, $M_r(1,3)$, $M_r(30,3)$) of the dataset to synthesize 15 different intermediate virtual views (i.e., $M_v(1,2)$ to $M_v(15,2)$) by RVS with a large baseline-15cm. As illustrated in Figure 7(b),we used

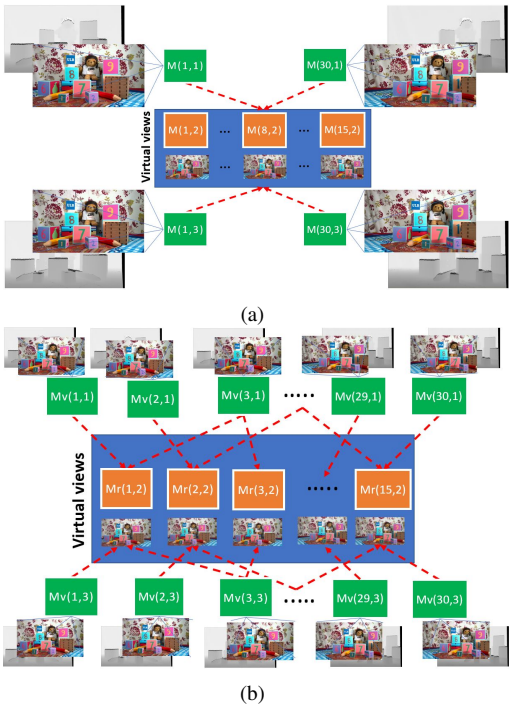


Figure 7: Experimental configuration using RVS

Table 1: Experimental configuration using RVS

RVS setup	DERS/LiDAR Camera
No. of input (image+depth)	4
Color image resolution	1920x1080
Depth resolution	1920x1080
Large baseline	15cm
Small baseline	2cm

RVS to synthesize the same virtual views but with a small baseline of 2cm with the inputs selected orderly from four adjacent reference images plus corresponding depth maps. This process accordingly was repeated with different depth maps DERS, LiDAR-UCRD, and LiDAR-CRD. In our evaluation, we have combined objective evaluation by IV-PSNR and subjective evaluation by examining the virtual views' quality.

Objective Evaluation

LiDAR-UCRD vs. LiDAR-CRD: Figure 8 (a) and (b) show the performance evaluation of the virtual view quality with LiDAR CRD and LiDAR UCRD in IV-PSNR, based on different baseline setup to RVS (Figure 7). The 15 virtual views ($M_{(1->15,2)}$) performance with LiDAR-CRD (blue line) is slightly higher than with LiDAR-UCRD (red line). Nevertheless, these gaps are nearly negligible since IV-PSNR less sensible to pixel shifting, and this can not show the quality improvement by precise camera calibration. Therefore, we have used subjective evaluation to approve the benefits of improving virtual view quality with LiDAR-CRD, demonstrated in the following subsection (Subjective Evaluation).

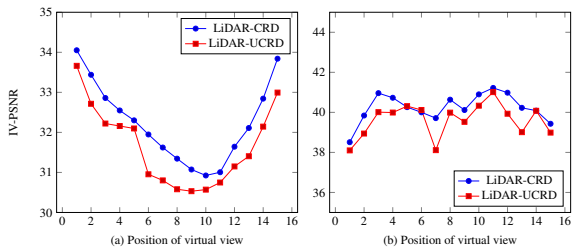


Figure 8: IV-PSNR (virtual view), LiDAR-UCRD vs. LiDAR-CRD. (a) Virtual views $M_{((1->15,2)}$ (15cm BL); (b) Virtual views $M_{(1->15,2)}$ (2cm BL)

LiDAR-CRD vs. DERS: According to Figure 9 (a) and (b) the virtual view synthesis using LiDAR CRD $M_{(1->15,2)}$ maintain at least 32.2dB with a large 15cm baseline or 40dB with a small 2cm baseline in IV-PSNR

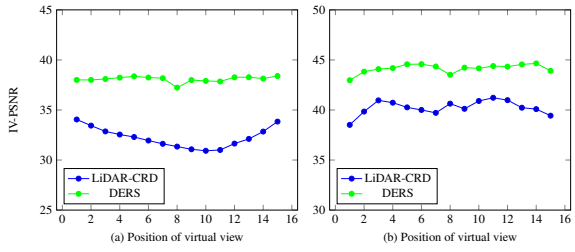


Figure 9: IV-PSNR (virtual view), LiDAR-CRD vs. DERS. (a): Virtual views $M_{(1->15,2)}$ (15cm BL); (b): Virtual views $M_{(1->15,2)}$ (2cm BL)

are acceptable values. Compared with the DERS-based (green line) virtual views, the latter outperforms about 4dB.

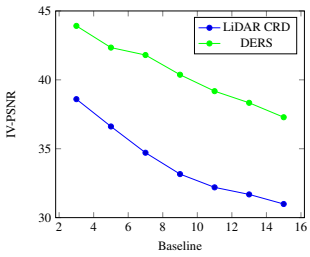


Figure 10: IV-PSNR (virtual view), LiDAR-CRD vs. DERS: For viewpoint M(8,2), when the baseline distance among reference views varies

Figure 10 shows the virtual views' with DERS and LiDAR-CRD given a viewpoint at M(8,2), while the baseline distance for the references images are varied. For one dedicated intermediate virtual view $M_{(8,2)}$, DERS-based virtual view has a constant superiority in IV-PSNR than LiDAR-based virtual view.

Subjective Evaluation

Referring to the objective measures presented above, we cannot fully conclude precisely. To have a better understanding of the advantages and disadvantages of LiDAR vs. DERS, in the following, we compare them subjectively and report their computational performances.

Figure 11(1th row) demonstrates one of ground-truth (original image) and outputs of RVS based on three different depth maps of DERS, LiDAR UCRD, LiDAR CRD. We have used the error map Figure 11(3rd row) to better demonstrate the differences between each virtual view and the ground truth. In the error map, brighter pixel color means a more significant error; and vice versa.

LiDAR-UCRD vs. LiDAR-CRD: Fig, 11(c), (d) and (4th row) show that the virtual view quality with LiDAR CRD significantly better than with LiDAR-UCRD. Thanks to the precise calibration and the accurate depth registration, the virtual view has fewer pixels shifting in the objects.

LiDAR-CRD vs. DERS: The error map of Figure 11(3rd row) shows the main reason for higher IV-PSNR in DERS compared to LiDAR-CRD. LiDAR camera suffers from the typical weakness in boundary scanning. The LiDAR CRD-based view virtual has some objects with border shrinking according to Figure 11((b), (d), in the first two columns of 4th row). Following three reasons can explain these differences: **1) The excessive incident Angle of the laser**: Too wide intersection angle between the incident and reflected

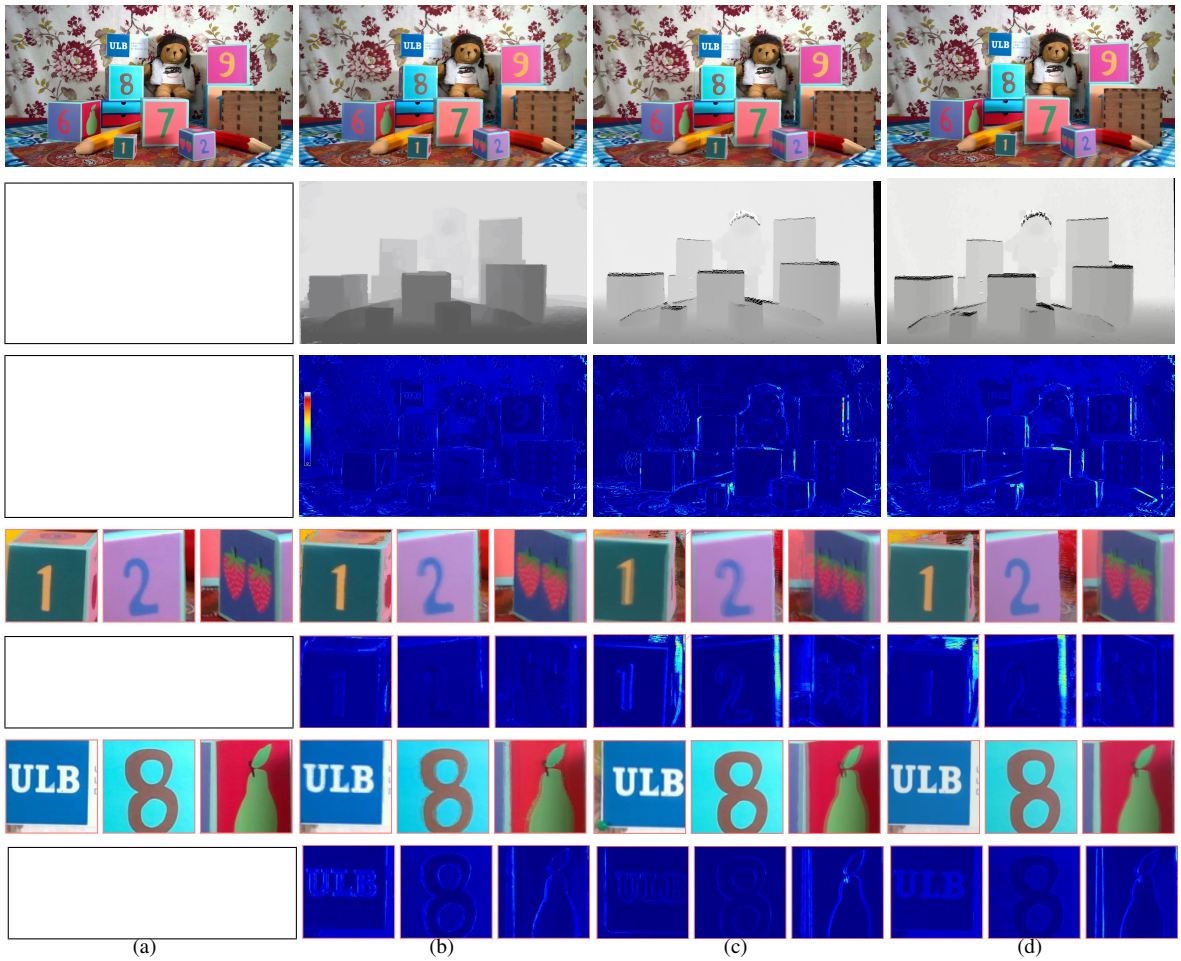


Figure 11: Subjective evaluation: (a) Ground-Truth, for column (b), (c) and (d), the first-row is DERS-based virtual view, second-row is LiDAR-UCRD-based virtual view, third-row is LiDAR-UCRD-based virtual view and the other rows are magnified regions and corresponding error maps.

ray that hampers LiDAR camera to get receiving echoes properly. **2) Registered depth map occlusion**: The occlusion problem of depth registration, which is vertically shifted in this LiDAR. **3) Laser interference**: The superfluousness of depth scanning among complex structural scenes, echoes interfere with each other at the high multi-reflection area. This interference brings some errors into depth information measurement.

However, DERS also has its vulnerability. Its performance is usually worse in low texture areas because its cost matching step relies on changing pixel value intensity. Low texture areas make cost matching difficult that prominently affects DERS output accuracy. Due to this flaw, the DERS-based view has some objects with pixels shifting. In contrast, LiDAR cameras do not suffer from getting wrong depth information in low texture areas as they benefit from its active acquisition attribute. The differences shown in Figure 11((b), (d), 5th row)

Last but no least, we compare the computation performance required for acquiring depth by LiDAR and estimating by DERS. DERS requires extremely high computational resources than the LiDAR according to the Table 2. The run time easily reaches around 4minutes to estimate one depth map with a high-class PC. Therefore, it is generally not possible to use DERS for a real-time DIBR system purpose.

Table 2: Comparison of the computation costs between DERS and LiDAR

PC Configuration	i9-10900X, 64GB Ram	
	DERS	LIDAR
CPU usage	90%	6% $\sim$ 30%
Ram usage	12GB	250Mb
Processing time	4.1 mins	Real time (30fps)

5 CONCLUSION

Using the accurately calibrated and precisely registered depth maps of the Intel Realsense LiDAR camera can output sufficiently good virtual views by RVS. When

compared with the DERS-based virtual views, the latter outperforms the former objectively in IV-PSNR. We have subjectively observed that DERS only outperforms LiDAR in border areas. LiDAR camera's depth maps have better performance than DERS in low texture with no border area. Both DERS and the LiDAR camera's depth maps have a similar performance in high texture with no border area. Therefore, overall, the LiDAR camera's depth maps showed a substantial trade-off to DERS in virtual view quality subjectively. Moreover, DERS requires remarkably high computational resources, and its processing run time is relatively long, up to several minutes per depth map estimation. In contrast, using the LiDAR camera, without any computational cost, can achieve an acceptable trade-off in subjective quality in comparison with DERS. Therefore, we recommend the RGB-D camera such as LiDAR for real-time DIBR application purposes.

6 ACKNOWLEDGMENTS

This paper has been supported by the EU project HoviTron (Holographic Vision for Immersive Tele-Robotic OperatioN), Call identifier: H2020-ICT-2019-3, Grant Agreement: 951989. Sarah Fachada is a research fellow at FNRS (Fonds de la recherche scientifique). Authors appreciate the support of Prof. Eduardo Juarez, in cross checking the achieved results by Mr. Jaime Sancho, at UPM, Spain.

REFERENCES

[1] S.R. Barry and O. Sacks. *Fixing My Gaze: A Scientist's Journey Into Seeing in Three Dimensions*. Basic Books, 2009. ISBN: 9780786744749.

[2] Daniele Bonatto, Sarah Fachada, and Gauthier Lafruit. In: *RaViS: Real-time accelerated View Synthesizer for immersive video 6DoF VR*. Vol. 2020. Jan. 2020, pp. 382–1. DOI: 10.2352/ISSN.2470-1173.2020.13.ERVR-381.

[3] Y. Boykov, O. Veksler, and R. Zabih. In: *Fast approximate energy minimization via graph cuts*. Vol. 23. 11. 2001, pp. 1222–1239. DOI: 10.1109/34.969114.

[4] Y. Y. Boykov and M. -. Jolly. "Interactive graph cuts for optimal boundary region segmentation of objects in N-D images". In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 1. 2001, 105–112 vol.1. DOI: 10.1109/ICCV.2001.937505.

[5] Yan-Pei Cao, Leif Kobbelt, and Shi-Min Hu. "Real-time High-accuracy Three-Dimensional Reconstruction with Consumer RGB-D Cameras". In: *ACM Transactions on Graphics* 37 (Sept. 2018), pp. 1–16. DOI: 10.1145/3182157.

[6] Adrian Dziembowski. In: *Software manual of IVPSNR for Immersive Video*. ISO/IEC JTC 1/SC 29/WG 04 N0013, (Oct. 2020).

[7] Christoph Fehn. "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV". In: *Stereoscopic Displays and Virtual Reality Systems XI*. Ed. by Mark T. Bolas et al. Vol. 5291. International Society for Optics and Photonics. SPIE, 2004, pp. 93–104. DOI: 10.1117/12.524762.

[8] Paul Furgale, Joern Rehder, and Roland Siegwart. In: *Unified temporal and spatial calibration for multi-sensor systems*. Nov. 2013, pp. 1280–1286. DOI: 10.1109/IROS.2013.6696514.

[9] Ying He et al. "Depth Errors Analysis and Correction for Time-of-Flight (ToF) Cameras". In: *Sensors* 17 (Jan. 2017), p. 92. DOI: 10.3390/s17010092.

[10] Jörn Rehder et al. In: *Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes*. 2016, pp. 4304–4311.

[11] A. Schenkel S. Fachada D. Bonatto and G. Lafruit. "Depth image based view synthesis with multiple reference views for virtual reality". In: *2018 - 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*. 2018, pp. 1–4. DOI: 10.1109/3DTV.2018.8478484.

[12] S.Rogge et al. In: *MPEG-I Depth Estimation Reference Software*. 2019, pp. 1–6. DOI: 10.1109/IC3D48390.2019.8975995.

[13] M. Tanimoto et al. In: *Free-Viewpoint TV*. Vol. 28. 1. 2011, pp. 67–76. DOI: 10.1109/MSP.2010.939077.

[14] M. P. Tehrani et al. In: *Free-viewpoint image synthesis using superpixel segmentation*. Vol. 6. June 2017. DOI: 10.1017/ATSIP.2017.5.

[15] Krzysztof Wegner and Olgierd Stankiewicz. In: *DERS Software Manual*. ISO/IEC JTC1/SC29/WG11 M34302, (07. 2014).

[16] Ning Xu and Yeong-Taeg Kim. "Luminance preserving color conversion for 24-bit RGB displays". In: *2009 IEEE 13th International Symposium on Consumer Electronics*. 2009, pp. 271–275. DOI: 10.1109/ISCE.2009.5156864.

[17] Michael Zollhöfer et al. "State of the Art on 3D Reconstruction with RGB-D Cameras". In: *Computer Graphics Forum* 37 (May 2018), pp. 625–652. DOI: 10.1111/cgf.13386.

Lift: An Educational Interactive Stochastic Ray Tracing Framework with AI-Accelerated Denoiser

Gonçalo Soares
Instituto Superior Técnico/Inesc-ID
Universidade de Lisboa
Lisboa, Portugal
goncalofdsoares@tecnico.ulisboa.pt

João Madeiras Pereira
Instituto Superior Técnico/Inesc-ID
Universidade de Lisboa
Lisboa, Portugal
jap@inesc-id.pt

ABSTRACT

Real-time physically based rendering has long been looked at as the holy grail in Computer Graphics. With the introduction of Nvidia RTX-enabled GPUs family, light transport simulations under real-time constraint started to look like a reality. This paper presents Lift, an educational framework written in C++ that explores the RTX hardware pipeline by using the low-level Vulkan API and its Ray Tracing extension, recently made available by Khronos Group. Furthermore, to accomplish low variance rendered images, we integrated the AI-based denoiser available from the Nvidia´s OptiX framework. Lift's development arose primarily in the context of the graduate 3D Programming course taught at Instituto Superior Técnico and Master Theses focused on Real-Time Ray Tracing and provides the foundations for laboratory assignments and projects development. The platform aims to make easier students to learn and to develop, by programming the shaders of the RT pipeline, their physically-based rendering approaches and to compare them with the built-in progressive unidirectional and bidirectional path tracers. The GUI allows a user to specify camera settings and navigation speed, to select the input scene as well as the rendering method, to define the number of samples per pixel and the path length as well as to denoise the generated image either every frame or just the final frame. Statistics related with the timings, image resolution and total number of accumulated samples are provided too. Such platform will teach that nowadays physically-accurate images can be rendered in real-time under different lighting conditions and how well a denoiser can reconstruct images rendered with just one sample per pixel.

Keywords

Educational Ray Tracing framework, Nvidia RTX , Vulkan, Path Tracing, AI-accelerated Denoiser

1 INTRODUCTION

1.1 Motivation

For a long time, Real-Time Ray Tracing has been considered a far to reach dream where physically-based rendering would be calculated fast enough so that we would be able to interact with the scene and perceive the changes in real-time. Now that Nvidia RTX technology (Burgess, 2020) is available, the dream has become feasible more than ever before. Typical production renderers deal with offline image synthesis by using a large number of samples per pixel to render the best-looking image possible. Under time constraint, even the fastest ray tracers can only trace few rays per pixel at 1080p and 30Hz. While this number increases every few years, the trend is partially countered by the move towards higher resolution displays and higher refresh rates. It therefore seems likely that a realistic sampling budget for real-time applications will remain on the order of a low number of short paths per pixel which implies that the usage of Monte Carlo integration of indirect illumination leads to images containing very high levels of variance. To bridge this gap, recent research was able to design real-time reconstruction filters to remove the noise (denoise) from extremely low sample count images, thus allowing to synthesize noise-free images at real-time refresh rates (30Hz). These state-of-the-art denoisers often resort to machine learning methods like the works of (Chaitanya et al., 2017) and (Vicini et al., 2019). Over the years, multiple ray-tracing algorithms have been proposed to physically simulate the light transport and the interactions with materials of a scene under different lighting conditions. Now that it is possible to make these simulations at rates never seen before, it seems relevant to develop an educational framework that helps Master students to implement such algorithms by exploiting the RTX technology and by adding a denoising step, and then perform

a comparison between them to assess both the performance and the perceptual image quality.

1.2 Objectives

This paper describes the implementation of the Lift system which aims to provide an educational platform that makes easier students to learn and to program new light transport algorithms into a Vulkan RT-based working pipeline. Lift additionally provides a set of settings and statistics in order to facilitate the comparison of the students' programs with the built-in Monte Carlo algorithms, namely Path Tracing (PT) and Bidirectional Path Tracing (BDPT). The implementation of these two algorithms was leveraged on the graphics card RTX capabilities through the use of the Ray Tracing extension for the low-level Vulkan API (Daniel Koch Tobias Hector and Werness, 2020), recently made available by Khronos Group.

Furthermore, we integrated an AI-accelerated denoiser to be used together with both PT and BDPT algorithms.This denoiser is available in Optix (Parker et al., 2010), a programmable general-purpose ray tracing framework. The purpose was to create a platform that allows not only to achieve, in real-time, high-quality images with just one sample per pixel but also to check whether it is worth investing in more complex raytracing algorithms, like the BDPT algorithm, when compared to the unidirectional path tracer with denoiser.

Since OptiX's denoiser was our choice, we could use this API also to develop the internal RT working pipeline. However, one of the requisites of the 3D Programming Master course at Instituto Superior Técnico (IST) is to teach the Vulkan low level API and since its RTX extension was made available this year we decided to use Vulkan in the Lift framework. The development of the prototype was technically challenging because we had to build the Vulkan interoperability with the OptiX framework.

Summarizing, Lift is an educational framework that will help advanced students to learn and to tackle the following state-of-the-art questions:

- Can Ray Tracing algorithms render in real-time physically-accurate images, under different lighting conditions?
- How well a denoiser behaves in reconstructing images rendered with one sample per pixel?
- Is it worth investing on complex light transport algorithms over the simple path tracer with denoising?

In order to check if Lift was robust enough to address properly the above questions, we gathered data from multiple scenes rendered with both algorithms, PT and BDPT, with image denoising feature enabled and disabled. Then, we compared both performance and image quality results. This evaluation is described in the paper too.

Figure 1: Lift: the graphical interface

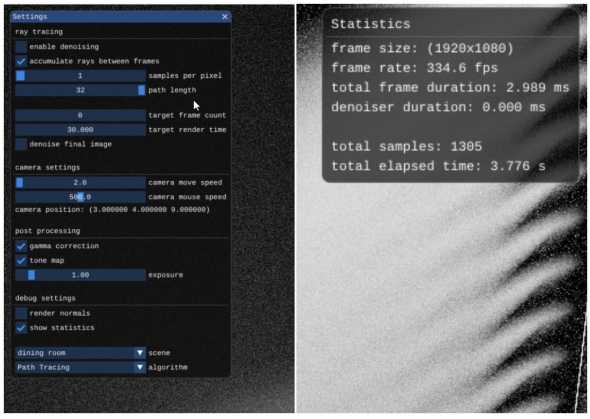


Figure 2: Tool's GUI zoom in

2 LIFT IMPLEMENTATION

2.1 Overview

The prototype was designed around a rendering architecture with progressive refinement that allows the choice of one of the two light transport techniques, the selection of the input scene and the enabling of a feature to denoise the generated image either every frame or just the final frame. Concerning the final frame of the progressive rendering, the platform's GUI allows the user to set either a target number of accumulated samples or a desired rendering's elapsed time. Statistics related with the timings, image resolution and total number of accumulated samples are provided too. Figures 1 and 2 show the tool's graphical interface with the following configuration settings: path tracer selected, denoiser disabled, 1 sample per pixel (spp), 32 bounces for the maximum path length and the rendering target defined as 30 seconds elapsed time.

The denoising step is performed by the graphics card CUDA cores exposed via the Optix framework. OptiX provides a function that takes as input noisy images generated by light transport algorithms with a small

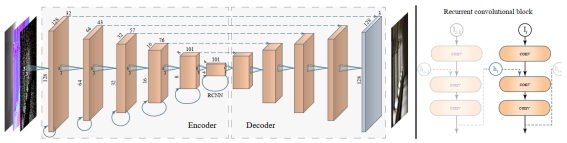


Figure 3: Architecture of the recurrent auto-encoder used in the OptiX denoiser (Chaitanya et al., 2017)

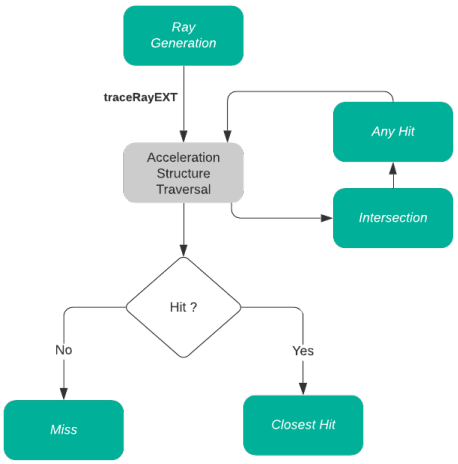


Figure 4: Vulkan's Ray Tracing Pipeline

number of samples and displays images with much higher quality, resembling images generated with many more samples.The denoiser is based on the work of (Chaitanya et al., 2017) which uses a variation of a deep convolutional network, namely the Recurrent Neural Network represented in Figure 3. The network architecture includes distinct encoder and decoder stages that operate on decreasing and increasing spatial resolutions, respectively. A recurrent block is placed at every encoding stage. Each recurrent block consists of three convolution layers with a 3 × 3-pixel spatial support. One layer processes the input features from the previous layer of the encoder. It then concatenates the results with the features from the previous hidden state, and passes it through two remaining convolution layers. The result becomes both the new hidden state and the output of the recurrent block.

2.2 Vulkan ray tracing pipeline

In order to render images by using Vulkan's Ray Tracing Pipeline, firstly it is necessary to setup up the way the Acceleration Structures (AS) are laid out in the Device Memory. Secondly, these structures should be loaded with the geometry of the scene. Finally, it would be also needed to perform their update configuration in order to support dynamic scenes which is not our case. The pipeline behaviour depends on the GLSL coding of a set of shaders: Ray Generation, Intersection, Hit, and Miss shaders. The relationship between these shaders (green blocks) is depicted in Figure 4.

Ray Generation shaders are the entry point of the ray tracing process: they are executed for each pixel on a multi-threaded 2D grid. This shader typically initializes a ray starting at the location of the camera and in a direction given by evaluating the camera lens model at the pixel location. It will then cast the rays into the scene. Other shaders below it will process further events, and return their result to the Ray Generation shader through the ray payload record. Finally, it will write the algorithm's output to memory, e.g., the screen or a texture. Two Ray Generation shaders were coded regarding both PT and BDPT algorithms. These two shaders are detailed in the next subsection.

Intersection shaders are used to implement ray-primitive intersection algorithms, other than ray-triangle intersections which have built-in support.This provides flexibility by allowing scenes with custom primitives like hair or spheres. An Intersection shader was programmed to support the classical optimized sphere-ray intersection algorithm (Akenine-Moller et al., 2018).

There are two types of Hit shaders in Vulkan: one is optional and is named Any Hit, the other is called Closest Hit. An Any Hit shader is run every time any intersection is found. The Closest Hit shader is called only for the closest intersection. A typical usage for an Any Hit shader is related with alpha testing: if the ray hits a transparent point then the intersection will be discarded and the traversal continues with that ray.

We didn't use the Any Hit shaders and our Closest Hit shaders are just responsible for creating the ray payload record which stores both the geometry and material information of the intersected primitive. This record is then sent back to the Ray Generation shader.

Miss shaders are called when a ray fails to intersect with any geometry in the scene. These shaders typically return the scene's background color or a texel from an environment map.

2.3 Ray Generation shaders

Two Ray Generation shaders were GLSL coded regarding both PT and BDPT algorithms and are described below.

2.3.1 Path Tracing

With Path Tracing the color of a pixel is computed by the averaged sum of all the paths starting from that pixel and eventually hit an emissive object. The number of paths per pixel is defined by the number of samples per pixel (spp). Progressive refinement allows the user to start rendering and to watch the image as its quality improves and stop the process once satisfied. To accomplish progressive refinement on the displayed image frame, each current pixel color is accumulated with

the contribution from previous frames and then submitted to post-processing effects, like exposure, tone mapping, and gamma correction before to be stored in a texture. The PT Ray Generation shader can be represented by the following pseudo-code:

```
void main() {
  vec3 pixel_color = vec3(0);
  for(int s = 0; s < number_of_samples; s++)
  {
    origin = pixel + random_offset();
    pixel_color += traceEyePath(origin,direction);
  }
  pixel_color /= number_of_samples;
  accumulateWithPreviousFrames(pixel_color);
  applyPostProcessing(pixel_color);
  storePixel(pixel_color);
}
```

The contribution of a path is calculated by the function ***traceEyePath***, which starts a path based on both origin and direction data inputs. Importance sampling, Next Event Estimation (NEE) and Russian Roulette techniques were deployed in its implementation. At the closest intersection, it's firstly checked if the material is emissive resulting, in that case, the termination of the path and the return of the emissive color. Otherwise, the Bidirectional Scattering Distribution Function (BSDF) of the material is importance sampled in order to calculate the direction of the path's next bounce. Then, the NEE is implemented by shooting a shadow ray from the current intersection towards an importance sampled point in a luminary or emissive object. In terms of luminaries, area lights are supported in our prototype. If there is no intersection, it means that the hit point is directly illuminated and therefore a contribution is added to that path. This contribution is weighted by the reciprocal of the probability density function (pdf) of the chosen luminary importance sampling operation. Finally, further extending of the current path is determined by the Russian Roulette Termination method. Russian roulette allows us to randomly stop computing terms in a sum as long as we reweight the terms that are not terminated. This implies the definition of a termination probability q based on the path throughput weight.Thus, the contributions of the surviving bounces will be increased by a factor $1/q$ in order to compensate the terminated bounces. The code of the function ***traceEyePath*** is listed below:

```
vec3 traceEyePath(vec3 origin, vec3 direction) {
  vec3 color = vec3(0);
  for(int bounce=0;bounce < path_length;bounce++){
    traceRay(scene, origin, direction);
    if (Hit is Emissive)
        return color * emissive;
    sampleBSDF();
    if (Ray missed) return env_color;
    //Next event estimation
    bool in_shadow = traceShadowRay();
    if (not in_shadow)
        color += material_clr / pdf;
    russian_roulette_termination();
  }
  return color;
}
```

2.3.2 *Bidirectional Path Tracing*

When compared to the Path Tracing algorithm, BDPT traces paths from the luminaries in addition to paths from the camera and connects the paths with shadow rays. The BDPT Ray Generation shader is similar to the PT one. The main difference consists of tracing, in first place, the light paths from the luminaries and performing a random walk into the scene. At each intersection, it is recorded the vertex position, the material as well as the pdf associated with the sampling of the material's BSDF function to determine the direction of the light path's next bounce. Next step consists of tracing the camera paths. And finally, at each vertex of a camera path, we connect it to all the vertices of a light path by using shadow rays: if no object is occluding the two vertices, the weight of this edge is properly computed with Multiple Importance Sampling (MIS) technique by using the balance heuristic estimator from (Veach and Guibas, 1995), and divided by the corresponding pdf. The pseudo-code for the BDPT Ray Generation shader is listed below.

```
void main() {
  vec3 pixel_color = vec3(0);
  for(int s = 0; s < number_of_samples; s++)
  {
    origin = pixel + random_offset();
    buildLightPath();
    pixel_color += traceEyePath(origin,direction);
  }
  pixel_color /= number_of_samples;
  accumulateWithPreviousFrames(pixel_color);
  applyPostProcessing(pixel_color);
  storePixel(pixel_color);
}
```

The function ***buildLightpath*** builds the light paths from the luminaries which will be used in the ***traceEyePath*** function to combine with the eye paths, as listed below:

```
vec3 traceEyePath(vec3 origin, vec3 direction) {
  vec3 color = vec3(0);
  for(int bounce=0;bounce < path_length;bounce++){
    traceRay(scene, origin, direction);
    if (Hit is Emissive)
        return color * emissive;
    sampleBSDF();
    if (Ray missed) return env_color;
```

```
    for (int l = 0; l < LIGHTPATHLENGTH; l++) {
     bool in_shadow=traceShadowRay(light_paths[l]);
     if (not in_shadow)
       color += material_clr / pdf / MIS weight;
    }
    russian_roulette_termination();
  }
  return color;
}
```

2.4 OptiX - Vulkan interoperability

The denoiser uses Cuda and rendering is done with Vulkan. The OptiX – Vulkan interoperability was built through image sharing buffers with semaphore synchronization.

The OptiX denoiser is using Cuda, so the rendered image will need to be shared between Vulkan and Cuda. The denoiser actually requires linear images, which are not available to Vulkan. So instead of directly sharing the images, we create buffers which are shared between Vulkan and Cuda and we copy the images to the buffers, converting them to linear images.

A semaphore strategy was used to tackle the problem of synchronization between denoising and rendering tasks. We use a Vulkan timeline semaphore to signal when the ray traced image is rendered and transferred to the buffer and a Cuda wait semaphore to hold the execution of the denoiser on the GPU. We add the inverse process at the end of the denoiser with a Cuda semaphore signaling the image is denoised and on Vulkan a wait semaphore to copy the buffer back to an image, tonemap it and display.

3 EVALUATION AND RESULTS

3.1 Testing Methodology

The evaluation of Lift tool was performed by using a set of scenes with different types of BSDFs in different lighting conditions in order to encompass a diversity of optical effects simulation like specular and glossy reflections/refractions, color bleeding or caustics reflections. The used scenes are illustrated in Figures 5, 6, 7 and 8 which were rendered in our prototype.

The *Teapot Cornell Box* is a simple scene with few primitives, featuring the classic Cornell Box with a teapot and showcasing different types of materials such as diffuse, glass, and colored metal.This way, optical effects like color bleeding, glossy reflection and refraction as well as caustics can be tested. The *Covered Dragon* scene includes the same classic Cornell Box too but we added a rectangular area to cover the ceiling luminary in order to make it difficult to reach. To raise the scene's geometric complexity, the known Standford Dragon, with approximately 5,500,000 triangles and

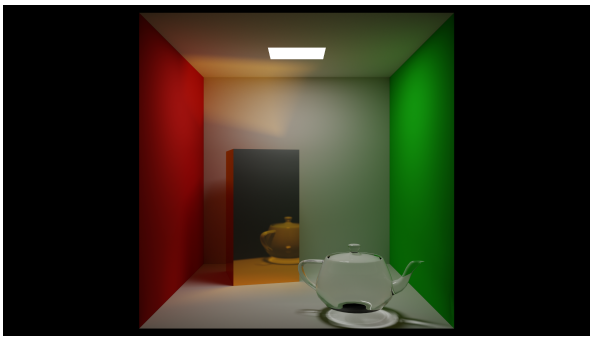


Figure 5: Teapot Cornell Box

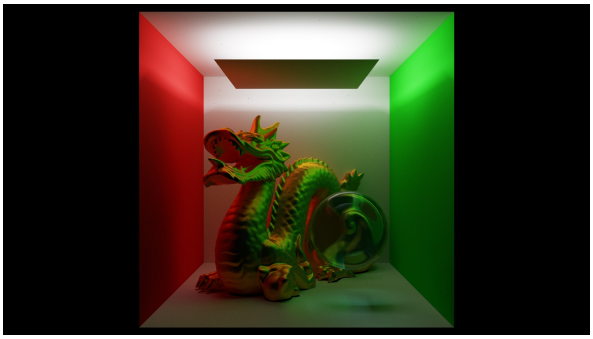


Figure 6: Covered Dragon

Figure 7: Breakfast Room (Bitterli, 2016)

Figure 8: Japanese Classroom (Bitterli, 2016)

a sphere were added. The *Breakfast Room* (Bitterli, 2016) scene features multiple different geometries and includes a lightsource placed outside the room which is illuminated through a window with blinders. Finally, we chose the *Japanese Classroom* (Bitterli, 2016), a commonly used scene to benchmark ray tracing applications and denoisers.

Both qualitative and quantitative metrics were used to benchmark the PT and BDPT algorithms. Regarding the qualitative metric to make the final image quality assessment, Structural Similarity Index Metric (SSIM) (Zhou Wang et al., 2004) is the preferred method for Monte Carlo rendered images (Whittle et al., 2017). We used a tool that computes (dis)similarity (DSSIM) between two or more PNG images using the SSIM algorithm at multiple weighed resolutions (Kornelski, 2020). The returned value is $1/SSIM - 1$, where 0 value means identical image, and > 0 (unbounded) represents the amount of difference.

The rendering experiments in both ray tracing algorithms were conducted with progressive refinement by tracing into the scene, in each frame, a single random walk per pixel (1 spp) with a maximum path length of 32 bounces. The ground truth images used in the perceptual image quality assessment (Figures 5, 6, 7 and 8) were generated with the above settings and additionally with the target rendering´s elapsed time set to 90 minutes.

The specifications of the testing computer machine were a AMD Ryzen 7 2700 8-Core CPU with a base clock frequency of 3.2GHz and 16GB of DDR4 RAM, and an NVIDIA GeForce RTX 2070 GPU with 8GB of GDDR6.

3.2 RESULTS

3.2.1 Performance

The performance of both ray tracing algorithms, PT and BDPT, was measured. The chart depicted in Figure 9 exhibits the average frames per second (fps) reached by each algorithm taking into account the time to render a frame. Consult Table 1 for more figures at different resolutions. At 1920x1080 resolution, Path tracing easily accomplished real-time rates in every benchmark scene. As far as it concerns to the BDPT, despite of its algorithmic complexity and consequently lower performance figures, it was also capable to sustain real-time framerates on the testing scenes. The worst performance scenario occurred with the *Japanese Classroom* scene where BDPT only performed at 19 fps.

The Table 2 exhibits the time spent on denoising a single frame. We verified that the duration of the denoising operation is independent of both the scene and the ray tracing algorithm being only affected by the image resolution. Mostly important, real-time framerates

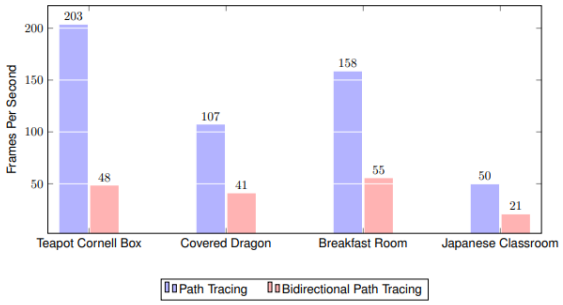


Figure 9: Average Frame Rate on 1920x1080 Resolution

	Frame Duration [ms]	
	PT	BDPT
Teapot Cornell Box 1280x720	2.3	9.6
Covered Dragon 1280x720	4.6	11.3
Breakfast Room 1280x720	2.9	8.5
Japanese Classroom 1280x720	9.3	22.2
Teapot Cornell Box 1920x1080	4.8	20.8
Covered Dragon 1920x1080	9.5	24.4
Breakfast Room 1920x1080	6.3	18.6
Japanese Classroom 1920x1080	20.2	52.4

Table 1: Frame Rendering times

were always accomplished by using the denoiser with PT algorithm and sometimes with the BDPT approach. For instance, consulting Table 1, the worst performance of PT algorithm occurred with the *Japanese Classroom* scene where the rendering of a *1920x1080* frame resolution took 20.2 milliseconds. By enabling the denoiser every frame we got 29 fps which is consistent with the duration of 14.5 milliseconds for the denoising step.

The downloaded supplemental material includes a video sequence with an animation of the movement of the camera in several scenes.

Image Resolution	1280x720	1920x1080
Denoiser Duration [ms]	7.5	14.5

Table 2: Denoiser working times

It is worth mentioning that we were able to run in Lift platform both light transport algorithms exceeding real-time rates at 1920x1080 resolution, as long as we kept a single random walk per pixel every frame. For instance, in the *Japanese Classroom* scene, by running PT algorithm with 16 spp and keeping 32 as the max path length, we achieved only 0.5 fps.

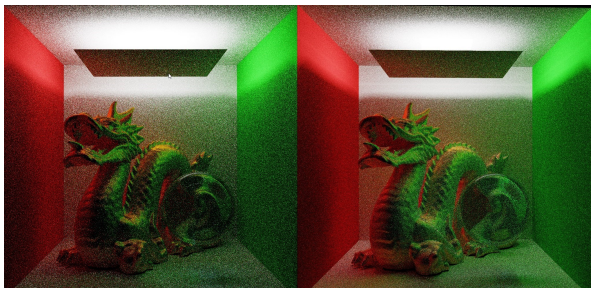


Figure 10: "Covered Dragon" image rendering after 5 seconds - Left: PT with 601 accumulated frames; Right: BDPT with 210 accumulated frames

3.2.2 *Perceptual Image Quality Assessment*

Visual inspection and convergence rates analysis were the main strategies to benchmark the perceptual quality. The progressive refinement feature allows the user to watch the image rendering as its quality improves and converges to the corresponding ground truth image. Thus, the temporal evolution of the DSSIM metric defines the convergence rate.

Firstly, we used visual inspection to compare both BDPT and PT algorithms without denoising. As expected, due to its algorithmic complexity, for the same rendering elapsed time, we got higher noisy BDPT rendered images than with PT for all scenes except for the Covered Dragon scene. As explained before, indirect illumination predominates in this scene since the luminary is covered. Thus, BDPT excels due to the combination of eye paths with light paths. Figure 10 depicts the rendered image after 5 seconds for both algorithms. We can notice that BDPT even with a lower number of accumulated samples per pixel (210 versus 601) performed better than PT algorithm.

Secondly, we analyzed how the application of the reconstruction (denoising) step affects the image quality for both light transport algorithms. This class of experiments was performed by setting the target rendering´s elapsed time to 16 milliseconds. The Figure 11 exhibits the final image frame after rendering the *Japanese Classroom* scene with PT algorithm during 16ms. This image rendering implied the accumulation of 8 frames, meaning that every pixel just received a contribution of 8 paths with a maximum of 32 bounces length each. Due to this low sample count, a significant amount of variance can be perceived in the image. Then, we repeated the experiment but now with the BDPT algorithm and the result is represented in Figure 12. It's noticeable the higher variance in the image due to the fact that, with 16ms elapsed rendering time, only 4 samples were accumulated for every pixel. Lastly, we ran again the PT algorithm but now with the denoiser set to remove the noise just in the final frame. This image frame is illustrated in Figure 13 and the very low level of variance is immediately evident when compared with the previously performed experiments. As already mentioned before, enabling the denoising step to clean the final frame implied to extend the elapsed rendering time of 14.5 milliseconds. Figure 14 reveals a side by side comparison of these images.

Figure 11: "Japanese Classroom" image rendering with PT algorithm after 16 milliseconds: 8 accumulated samples per pixel

Figure 12: "Japanese Classroom" image rendering with BDPT algorithm after 16 milliseconds: 4 accumulated samples per pixel

Figure 13: "Japanese Classroom" image rendering with PT w/Denoiser after 30.5 milliseconds - 8 accumulated samples per pixel

And finally we analyzed the convergence rates. The charts depicted in Figures 15, 16 and 17 exhibit the temporal evolution of the DSSIM metric for the *Covered Dragon*, *Breakfast Room* and *Japanese Classroom* benchmark scenes, i.e. how the rendered image quality progressively increases by accumulating samples per pixel. To conduct these experiments, we set 16384 for

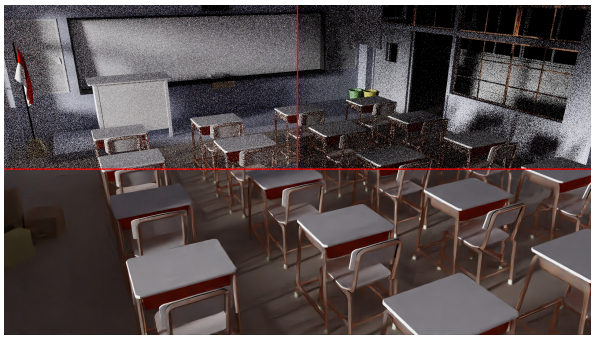


Figure 14: "Japanese Classroom" - Side by Side comparison rendered in 16 milliseconds. Path Tracing (Top-Left), Bidirectional Path Tracing (Top-Right), Path Tracing Denoised (Bottom)

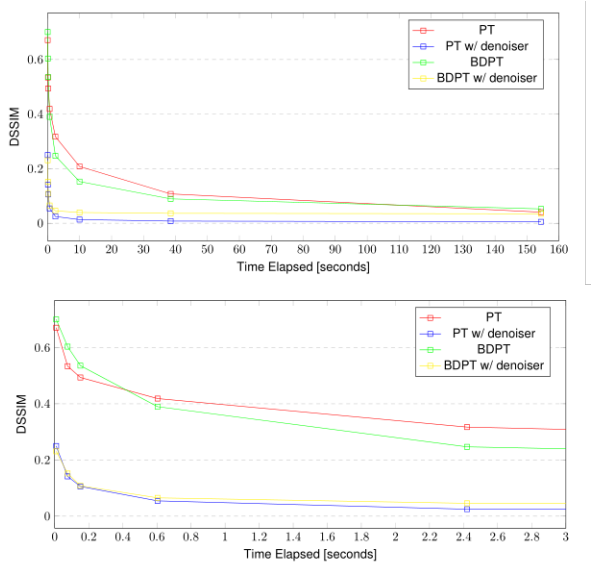


Figure 15: "Covered Dragon" - Top: DSSIM temporal evolution when accumulating 16384 frames; Bottom: Zoom in for the first 3s

the total number of accumulated frames as the rendering target.

As first observation, regardless of whether denoising operation is enabled, both light transport approaches converged to the ground truth images. This is confirmed by the decreasing trajectory of DSSIM metric over time until it gets close to 0.

A second observation, with the denoiser disabled, concerns the worst performance of BDPT algorithm compared to the unidirectional path tracer except for the Covered Dragon scene which confirmed the previous assessment done in Figure 10 with visual inspection. Looking at Figure 15 we verify that BDPT performs slightly better that PT algorithm.

As final observation, these charts clearly demonstrate that, with the denoiser enabled, the unidirectional path tracer, in all scenes, converged much faster to the corresponding reference images. Still with the denoiser on,

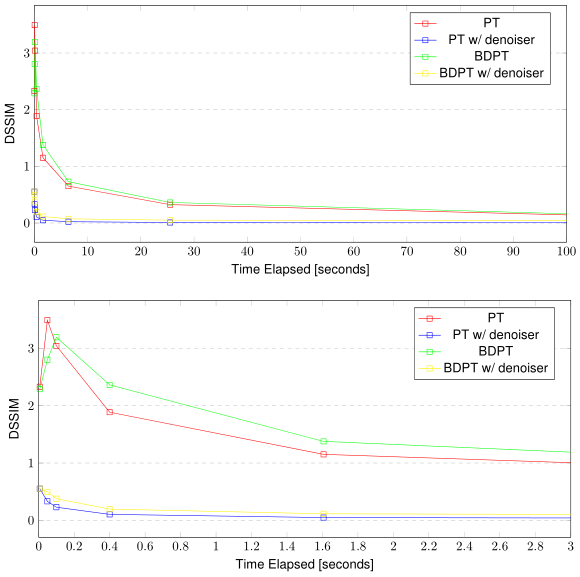


Figure 16: "Breakfast Room" - Top: DSSIM temporal evolution when accumulating 16384 frames; Bottom: Zoom in for the first 3s

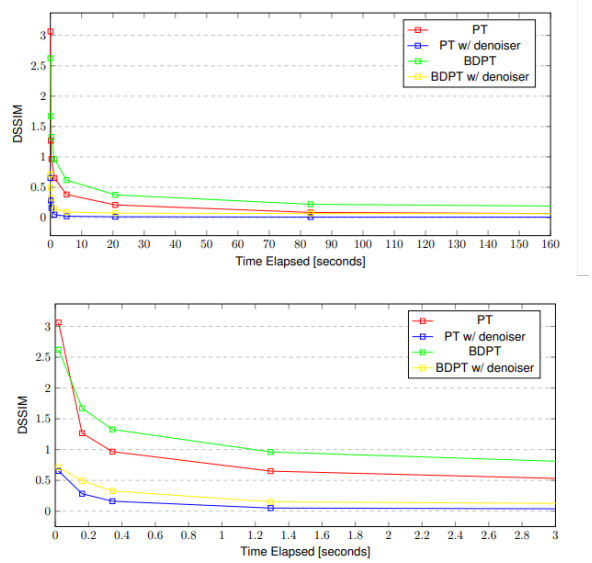


Figure 17: "Japanese Classroom" - Top: DSSIM temporal evolution when accumulating 16384 frames; Bottom: Zoom in for the first 3s

we verified that BDPT, in general, performed equally or slightly worse than PT technique.

Regarding the usage of the AI-accelerated OptiX denoiser in animation, we can notice in the video sequence, included in the supplemental material, the existence of flicker artifacts when the denoising operation occurs while the camera is moving. Several experiments lead us to conclude that this denoiser can be used for animation. However it still requires high sample counts for good results. With low sample counts, low frequency (blurry) noise can be visible in animation

frames, even if it not becomes immediately apparent in still images.

Thus, the evaluation of the two light transport algorithms in both metrics, performance and image quality, seems to indicate that, with this type of benchmark scenes, it is not worth investing in the more complex BDPT approach, because we are able to get better or similar results by using the unidirectional path tracer with denoiser.

4 TEACHING WITH LIFT

The need to develop Lift, an educational RT platform with an AI-Accelerated Denoiser, arose primarily in the context of the 3D Programming course taught at Instituto Superior Técnico (IST) and Master Theses focused on physically-based rendering. Developing a decent code base for the student projects or theses with Vulkan that could be quickly picked up was really a problem; students typically come from slightly different undergraduate studies and one could not expect the same level of competence in a programming language like C++, let alone delve into the details of the currently available ray tracing Vulkan API.

4.1 3D Programming Course

In our graduate course, 3D Programming, we supplement the theoretical lectures with 12 weekly one and half-hour laboratory sessions. Eight lab sessions are devoted to the Ray Tracing topic and include the use of Lift platform to solve the required exercises. The last four sessions regard the use of Unity3D for application development.

In the respective 8 lab classes, the students use Lift pipeline to develop two assignments which will be evaluated. In the first assignment the students should implement the Distribution ray tracer with support for glossy reflections/refractions, motion-blur and depth-of-field effects. In the second one, the students extend their ray-tracer from the previous assignment to get a Path Tracer and integrate it with the denoiser provided by Lift. Then, by using a set of benchmark scenes as well as their own scenes, the students should perform a comparison of their solutions with the built-in PT and BDPT, to assess both the performance and the perceptual image quality. In the end, the students will learn that nowadays physically-accurate images can be rendered in real-time under different lighting conditions and how well a denoiser can reconstruct images rendered with just one sample per pixel.

4.2 Master Theses

The preliminary results accomplished with Lift platform seem to favour PT with AI-accelerated Denoiser over BDPT as the best approach to obtain the highest image quality in the shortest amount of time. However, other complex ray tracing algorithms may still be able to outperform the path tracer under challenging lighting scenarios. With the Lift tool, students doing their Theses around physically-based rendering have all the programming foundations to fully answer to the question if is it worth investing on more complex algorithms, namely to perform further testing with other candidates like photon mapping-based techniques. Vertex Connection and Merging (VCM) and Unified Path Sampling (UPS) are effectively identical techniques independently developed by (Georgiev et al., 2012) and (Hachisuka et al., 2012), respectively. VCM/UPS combines bidirectional path tracing and progressive photon mapping, and is particularly advantageous for specular-diffuse paths and specular-diffuse-specular paths (i.e. caustics and specular reflections of caustics). We have currently four master students implementing over Lift platform the UPBP (unified points, beams, and paths) algorithm by (Křivánek et al., 2014) which is a generalization of VCM to volumes. It is particularly suitable for rendering volume caustics and reflections of volume caustics. Their method uses both point and beam lookups, and combines the results with Multiple Importance Sampling (MIS).

The Metropolis Light Transport-based approaches, originally proposed by (Veach and Guibas, 1997), would initially be off our radar because, in general terms, they can be seen as an extension of the bidirectional path tracer: instead of constantly creating new paths from scratch, paths are mutated into new ones. Anyway, two students are currently finalizing the evaluation of a MLT implementation over Lift and the preliminary results seem to indicate a slightly better behaviour when compared with BDPT.

5 CONCLUSIONS

In order to allow Master students, in the context of the 3D Programming course or their Theses, to address the current Real Time Ray Tracing questions formulated in section 1.2, the main objective was to build successfully an educational RT platform with an AI-Accelerated Denoiser. With Lift, we have shown to be able to accomplish interactive low-variance stochastic ray tracing. By using our prototype, the students will learn that nowadays physically-accurate images can be rendered in real-time under different lighting conditions and how well a denoiser can reconstruct images rendered with just one sample per pixel. In fact, with Lift they are able to run both light transport algorithms, PT and BDPT, in different lighting conditions exceeding real-time rates at 1920x1080 resolution, as long as we kept a single random walk per pixel every frame. We verified that the denoiser penalized the algorithms' performance by introducing a fixed reduction factor but it did not pre-

vent to achieve real-time framerates. Anyhow, this issue was largely compensated by the huge image quality improvements provided by the reconstruction filter. The OptiX denoiser revealed to own a remarkable behavior in reconstructing images rendered with just one sample per pixel. Its main drawback was the observation of flicker artifacts when the denoising operation occurs while the camera is moving. This means, that for animations, the filter is not spatio-temporal stable and our future roadmap includes its replacement by (Vicini et al., 2019) approach.

Resources: The source code repository on github is available at (Soares, 2020). The downloaded supplemental material includes a video sequence with an animation of the movement of the camera in several scenes.

6 ACKNOWLEDGEMENTS

The work reported in this article was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under project UIDB/50021/2020. This work has also been kindly supported by NVIDIA.

REFERENCES

Akenine-Moller, T., Haines, E., and Hoffman, N. (2018). *Real-time rendering, Fourth edition*. AK Peters/CRC Press.

Bitterli, B. (2016). Rendering resources. https://benedikt-bitterli.me/resources/.

Burgess, J. (2020). Rtx on—the nvidia turing gpu. *IEEE Micro*, 40(2):36–44.

Chaitanya, C. R. A., Kaplanyan, A. S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., and Aila, T. (2017). Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)*, 36(4):98: 1 – 12.

Daniel Koch Tobias Hector, J. B. and Werness, E. (2020). Ray tracing in vulkan. https://www.khronos.org/blog/ray-tracing-in-vulkan.

Georgiev, I., Krivánek, J., Davidovic, T., and Slusallek, P. (2012). Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6):192–1.

Hachisuka, T., Pantaleoni, J., and Jensen, H. W. (2012). A path space extension for robust light transport simulation. *ACM Trans. Graph.*, 31(6).

Kornelski (2020). Image similarity comparison simulating human perception. https://github.com/kornelski/dssim.

Křivánek, J., Georgiev, I., Hachisuka, T., Vévoda, P., Šik, M., Nowrouzezahrai, D., and Jarosz, W. (2014). Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph.*, 33(4).

Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., and Stich, M. (2010). Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics*.

Soares, G. (2020). Vulkan path tracer with optix denoiser integration. https://github.com/goncalofds/lift.

Veach, E. and Guibas, L. J. (1995). Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings SIGGRAPH'95*, pages 419–428. ACM.

Veach, E. and Guibas, L. J. (1997). Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, page 65–76, USA. ACM Press/Addison-Wesley Publishing Co.

Vicini, D., Adler, D., Novák, J., Rousselle, F., and Burley, B. (2019). Denoising deep monte carlo renderings. *Computer Graphics Forum*, 38.

Whittle, J., Jones, M., and Mantiuk, R. (2017). Analysis of reported error in monte carlo rendered images. *The Visual Computer*, 33:1–9.

Zhou Wang, Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Learning Cell Nuclei Segmentation Using Labels Generated with Classical Image Analysis Methods

Damian J. Matuszewski
Department of Information Technology, Uppsala University
Sweden, 75105 Uppsala
damian.matuszewski@it.uu.se

Petter Ranefall
Department of Information Technology, Uppsala University, and SciLifeLab BioImage Informatics Facility
Sweden, 75105 Uppsala
petter.ranefall@it.uu.se

ABSTRACT

Creating manual annotations in a large number of images is a tedious bottleneck that limits deep learning use in many applications. Here, we present a study in which we used the output of a classical image analysis pipeline as labels when training a convolutional neural network (CNN). This may not only reduce the time experts spend annotating images but it may also lead to an improvement of results when compared to the output from the classical pipeline used in training. In our application, i.e., cell nuclei segmentation, we generated the annotations using CellProfiler (a tool for developing classical image analysis pipelines for biomedical applications) and trained on them a U-Net-based CNN model. The best model achieved a 0.96 dice-coefficient of the segmented Nuclei and a 0.84 object-wise Jaccard index which was better than the classical method used for generating the annotations by 0.02 and 0.34, respectively. Our experimental results show that in this application, not only such training is feasible but also that the deep learning segmentations are a clear improvement compared to the output from the classical pipeline used for generating the annotations.

Keywords

Deep learning, U-Net, CellProfiler, Data annotation, Microscopy

1. INTRODUCTION

Neural network-based image segmentation and classification have made huge progress in the last decade. However, the ground truth creation is still a major bottleneck for training deep learning models in many applications [Lec15a, Xin17a, Gup19a]. Here, we propose a simple and efficient way to automatically generate training labels for the segmentation of cell nuclei – with minimal manual interaction. Of course, some data still has to be manually annotated to evaluate the automatic label generation and the network's performance. However, our approach limits this only to the images in the test set as the network is trained entirely on the automatically generated labels. Moreover, our deep learning (DL) model has shown to substantially improve the results compared to the classical image analysis pipeline that was used for generating its training labels.

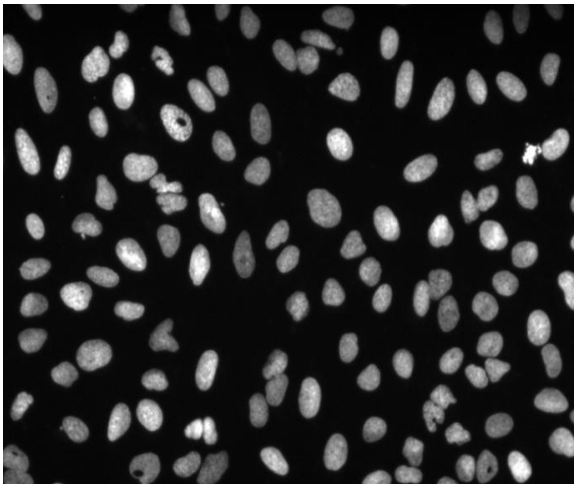


Fig. 1. Raw input image.

In Sect. 2, we describe the proposed method and neural network design, as well as the dataset that has been used for evaluation. In Sect. 3, we describe the metrics used for evaluation and present the results of the experiment.

2. METHOD

2.1 Dataset

As the application for demonstrating the proposed method, we have selected the task of cell nuclei segmentation. We used the BBBC039v1 dataset, available from the Broad Bioimage Benchmark Collection [Cai19a, Ljo12a]. This dataset has a manually created ground truth (GT), here only used for evaluating the results and not for training. The images are 690x520 pixels, and the nuclei typically have a diameter between 10 and 50 pixels. An example of an image from this dataset is shown in Fig. 1. For practical reasons, each image was tiled into four overlapping tiles of size 512x512 pixels. The tiles from the same image were always kept in the same

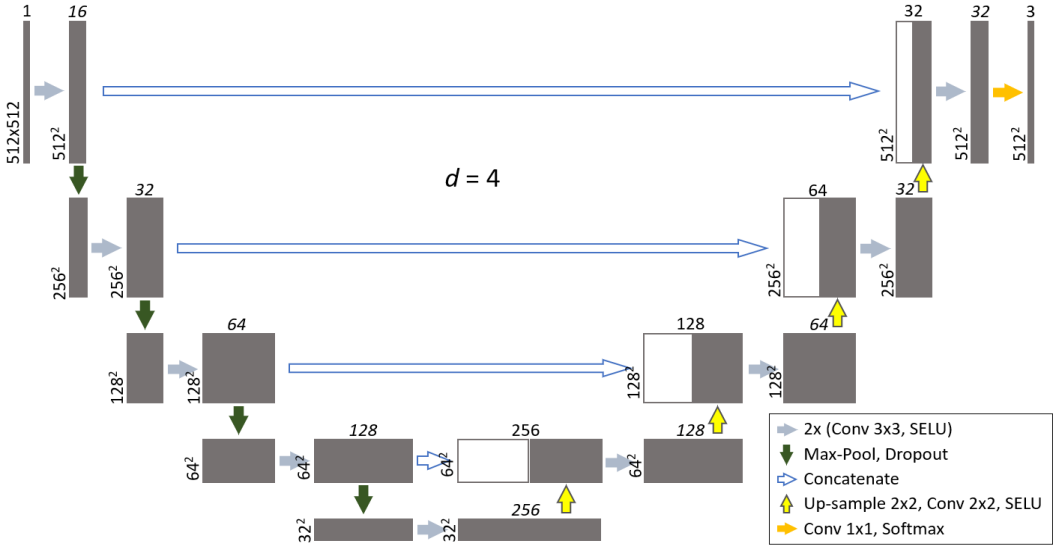


Fig. 2. The U-Net-based deep learning architecture.

data partition: training, validation, or test data subset, respectively. The initial dataset had 200 images with a recommended fixed split of 100-50-50, which resulted in 400 image tiles for training, 200 for validation, and 200 for the test. Before feeding the image tiles to the network, we first augmented them with all combinations of flipping and multiple 90 degrees rotations (this resulted in 8 image tiles from 1; it was done only with the training and validation sets and not the test set) and then normalized them by subtracting the corresponding mean intensity value from each augmented image tile and dividing it by the standard deviation, which is a standard procedure in deep learning.

2.2 Automatic generation of training labels

We used segmentation results of classical image analysis methods as the labels for training our neural network model. The segmentations were created using the CellProfiler [Mcq18a] module IdentifyPrimary-Objects with the threshold method 'Minimum cross-entropy' and the default Threshold smoothing scale=1.3488 to detect the objects [Sez04a], 'Shape' as the Method to distinguish clumped objects, and 'Intensity' as the Method to draw dividing lines. The object separation in CellProfiler is done using the Watershed algorithm [Vin91a]. The difference between 'Shape' and 'Intensity' when distinguishing clumped objects is that 'Shape' uses the local maxima of the distance transform [Bor86a] of the binary objects as seeds, whereas 'Intensity' uses the local maxima of the input image intensities as seeds. The separation lines are drawn by the Watershed algorithm for the previously defined seeds, based on the distance transform for 'Shape' or the input image intensities for 'Intensity'.

We also tried the three other segmentation configurations: Shape/Shape, Intensity/Shape, and Intensity/Intensity. However, the selected Shape/Intensity configuration gave the best results by a great margin. In all configurations, we used the same 10 to 50 pixels interval for the allowed object size (diameter). Finally, we tried the consensus between different segmentations: a combination of the three best segmentation configurations obtained by masking out (so that they are ignored during the training of the neural network) those cell clusters that did not get the same number of objects after the separation. The main idea behind it was to purify the training data by excluding difficult cases that were most likely erroneously labeled. However, this did not improve the results with respect to using only the best of the segmentation settings (Shape/Intensity), as shown in Section 3.

2.3 Neural network design

We used a modified U-Net [Ron15a], a fully convolutional neural network presented in Fig. 2. Following [Mat19a], we reduced the number of feature maps in each layer with respect to the original U-Net and increased it in the last convolutional block. We kept the same depth of the architecture as the original U-Net, i.e., 4. However, we added dropout layers [Sri14a] after each max-pooling with increasing rates: 0.2, 0.25, 0.3, and 0.35, respectively. We used zero padding and selu [Kla17a] as the activation function in all convolutional layers except the last one in which we used softmax. We also used L2 weights regularization with l=0.0001 in all convolutional layers except the output layer. We optimized the models using Adam [Kin14a] with the default hyperparameters except for the learning rate which was set to 0.00001. To make the training reproducible and to reduce the stochastic factors in our comparisons of training with different labels, we used the same

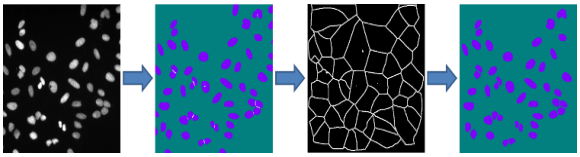


Fig. 3. Post-processing. From left to right: 1) raw input image, 2) segmentation results, 3) the cell separation lines are dilated and merged with the background; the resulting image is then skeletonized and dilated, 4) new separation lines are subtracted from the nuclei segmentation results; the objects touching the image edges are removed.

network architecture and hyper-parameters in all models. Moreover, we fixed the random initialization seeds of the network weights (we used Glorot normal initialization [Glo10a]) in all models as well. As a consequence, all models started with the same values in the corresponding network weights.

We trained the network to assign each pixel in the input image to one of the three classes: background (BG), nucleus (N), or nuclei separation lines (SL). We used a custom loss function λ_D: the class-weighted sum of the log Dice coefficient losses [Mat19a] defined by:

$$\lambda_D(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0.2 \log \delta_{BG} + 0.4 \log \delta_N + 0.4 \log \delta_{SL}), \quad (1)$$

$$\delta_C = \frac{2 \sum y_i \hat{y_i} + 1}{\sum y_i + \sum \hat{y_i} + 1}, \quad (2)$$

where y is the ground truth, $\hat{y}$ is the segmentation of class C, and δ_C is the pixel-wise Dice coefficient for that class. We trained each model for 50 epochs and selected the version with the smallest loss on the validation set annotated in the same way as the corresponding training set.

2.4 Post-processing

To ensure that the resulting division lines separate the clumped nuclei, we dilated them with a 3x3 pixel square structuring element, merged them with the background, and then skeletonized, followed by dilating the skeleton by 1 pixel. Finally, we subtracted the new divisions from the nuclei mask. Nuclei touching the edges of the images were also excluded, both in the ground truth and in the segmentations. See Fig. 3.

3. RESULTS

3.1 Metrics

As the metrics for comparing our results, we used two types of accuracy: 1) the pixel-wise Dice coefficient for the nuclei and separation lines before the post-processing; and 2) the object-wise Jaccard index J of the segmented nuclei after the post-processing:

$$J = \frac{|M|}{|G| + |R| - |M|}, \quad (3)$$

where $|G|$ is the number of nuclei in the ground truth (GT), $|R|$ is the number of nuclei in the result, and $|M|$ is the number of 1-to-1-matched nuclei between the results and the GT. There is a results-GT nucleus match if a segmented nucleus has a spatial overlap with exactly one nucleus in the GT, and that corresponding GT nucleus overlaps with exactly one nucleus in the segmentation results. The overlaps were computed between eroded (with a 3x3 pixels square structuring element) masks to avoid confusion of objects near the separation lines. The advantage of this metric is that it penalizes the wrong number of objects, but tolerates minor pixel differences. On the other hand, the Dice score coefficient penalizes wrong object boundaries but does not evaluate the number of segmented objects. Some applications require accurate nuclei counting while others need their precise boundary segmentation. As the proposed approach is not bound to any particular application, we report and compare our results with both metrics.

3.2 Results

Tables 1 and 2 present the results of the proposed methods evaluated on the test set with the GT (the manual annotations of the dataset). We can clearly see that adding the DL improved the results compared to the classical segmentations that had been used as input for its training. The DL results when trained on the best of the classical segmentation configurations (Shape/Intensity) are not far behind the results when training the same network (and with the same hyper-parameters) on the manual GT. Therefore, we can conclude that training on the output of a classical segmentation method is a good way to avoid the tedious work of creating manual labels for training and validation sets. However, the test set still has to be fully annotated to allow a proper assessment of the DL performance.

We can also observe that, contrary to our initial expectations, the segmentation consensus gave worse results in training DL than the best segmentation configuration alone. One possible explanation is that a noticeable part of the training data was masked out, and thus, the network had to learn how to perform segmentation on a smaller number of cells. A smaller, less representative training dataset usually leads to poorer generalization; and that is what we believe happened here. This may lead to the conclusion that it is better to include noisy or imprecise annotations rather than removing them from the training dataset. However, this hypothesis has to be tested in a much larger study with multiple datasets, various types, and levels of data corruption, and hence, it is outside the scope of this paper.

3.3 Processing Time

The initial segmentation and the post-processing were done in CellProfiler 3.1.9 on a laptop computer (64 GB RAM, 2.90 GHz Intel® Core™ i7-6920HQ CPU,

Windows 10 64 bit). The initial segmentation took about 5s/image and the post-processing took about 9s/image. These values are if run non-parallel, and only including the actual image processing – not saving, etc. However, in practice, this is automatically parallelized in CellProfiler. The neural network training and inference were done on a stationary computer (32 GB RAM, 3.30 GHz Intel® Core™ i9-7900X CPU, NVIDIA GeForce RTX 2060 SUPER 8 GB GPU, Windows 10 64 bit). Training time: ~ 43ms/update step, 140s/epoch (3200 image tiles), inference time: ~ 12ms/image tile, 2s/test set (200 image tiles).

4. CONCLUSION

We have demonstrated how classical image analysis methods can be combined with deep learning not only to reduce the tedious work of annotating images but also to improve the results offered by the classical pipelines. Our results show that it is possible to train a successful neural network without using any manually annotated ground truth; by using the output of a classical segmentation pipeline as training labels instead. Moreover, this approach gave a clear performance improvement compared to the classical methods that were used in generating the training data.

5. ACKNOWLEDGEMENTS

DJM received funding from the Swedish e-science initiative eSSENCE. PR is funded by the Swedish Science for Life Laboratory (SciLifeLab). The authors have no further relevant financial or non-financial interests to disclose.

Method	Nuclei	Separations
DL trained on GT	*0.964*	*0.492*
DL trained on Shape/Intensity	**0.958**	**0.355**
DL trained on Consensus	0.948	0.257
Classical Shape/Intensity	0.939	0.29

Table 1. Pixel-wise Dice coefficient before post-processing.

Method	Jaccard index
DL trained on GT	*0.892*
DL trained on Shape/Intensity	**0.838**
DL trained on Consensus	0.805
Classical Shape/Intensity	0.504
Classical Intensity/Shape	0.481
Classical Shape/Shape	0.464
Classical Intensity/Intensity	0.251

Table 2. Object-wise Jaccard index after post-processing.

6. REFERENCES

[Bor86a] Borgefors, G. (1986), Distance transformations in digital images, Comput.vision, Graphics, Image Processing, 34: pp. 334–371.

[Cai19a] Caicedo, J. C., et al. (2019). Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images. Cytometry Part A, 95(9): pp. 952–965.

[Glo10a] Glorot, X., Bengio, Y. (2010) Understanding the difficulty of training deep feedforward neural networks. In Proc. of the 13th International Conference on Artificial Intelligence and Statistics: pp. 249–256.

[Gup19a] Gupta, A., et al. (2019) Deep learning in image cytometry: a review. Cytometry Part A, 95(4): pp. 366–380.

[Kin14a] Kingma, D.P., Ba, J.L. (2014) Adam: A method for stochastic optimization. In Proc. of the International Conference on Learning Representations (ICLR), arXiv:1412.6980

[Kla17a] Klambauer, G., et al. (2017) Self-normalizing neural networks. Advances in Neural Information Processing Systems: pp. 971–980.

[Lec15a] LeCun, Y., et al. (2015) Deep learning. Nature, 521(7553): pp. 436–444.

[Ljo12a] Ljosa, V., et al. (2012). Annotated high-throughput microscopy image sets for validation. Nature Methods, 9, 637.

[Mat19a] Matuszewski, D. J., Sintorn, I.-M. (2019) Reducing the U-Net size for practical scenarios: Virus recognition in electron microscopy images. Computer Methods and Programs in Biomedicine, 178: pp. 21–39.

[Mcq18a] McQuin C., et al. (2018). CellProfiler 3.0: Next-generation image processing for biology. PLoS Biology, 16(7): e2005970.

[Ron15a] Ronneberger, O., et al. (2015) U-net: Convolutional networks for biomedical image segmentation. In Proc. of the International Conference on Medical Image Computing and Computer-Assisted Intervention: pp. 234–241.

[Sez04a] Sezgin, M., Sankur, B. (2004), Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging, 13(1): pp. 146–165.

[Sri14a] Srivastava, N., et al. (2014) Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1): pp.1929–1958.

[Vin91a] Vincent, L., Soile, P (1991)., Watersheds in digital spaces: An efficient algorithm based on immersion simulations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(6): 583598.

[Xin17a] Xing, F., et al. (2017) Deep learning in microscopy image analysis: A survey. IEEE Transactions on Neural Networks and Learning Systems, 29(10): pp. 4550–4.

Comparison of Lossless Image Formats

David Barina

Centre of Excellence IT4Innovations
Faculty of Information Technology
Brno University of Technology
Bozetechova 1/2, Brno
Czech Republic
ibarina@fit.vutbr.cz

Abstract

In recent years, a bag with image and video compression formats has been torn. However, most of them are focused on lossy compression and only marginally support the lossless mode. In this paper, I will focus on lossless formats and the critical question: "Which one is the most efficient?" It turned out that FLIF is currently the most efficient format for lossless image compression. This finding is in contrast to that FLIF developers stopped its development in favor of JPEG XL.

Keywords

Lossless Image Compression, PNG, JPEG-LS, JPEG 2000, JPEG XR, WebP, WebP 2, H.265, FLIF, AVIF, JPEG XL

1 INTRODUCTION

This paper compares the compression performance of state-of-the-art formats for lossless image compression. It deals with the essential formats from old PNG (1992) to modern JPEG XL (2020). The comparison is performed on three datasets. The first one represents high-resolution photos. The second one consists of artificial images (illustrations). And the third one is made up of scanned pages of books. The motivation for our comparison is to answer the question of which format is most efficient in terms of bits per pixel.

For comparison, we use tools that find the most suitable parameterization of the given compression formats. As far as we know, this makes us different from most published comparisons. Under these conditions, it turns out that the most efficient format for all three datasets is FLIF, closely followed by JPEG XL. This situation is specifically quite interesting because the FLIF development has stopped as JPEG XL supersedes it.

This paper consists of four sections. The Introduction section opens this paper. The Background section discusses the formats and datasets used. The Method and Results section presents our measurements. Finally, the Conclusions section closes the paper.

2 BACKGROUND

This section describes the individual lossless image formats in our comparison. We dealt with the following formats: PNG, JPEG-LS, JPEG 2000, JPEG XR, WebP, WebP 2, H.265, FLIF, AVIF, and JPEG XL.

Lossless algorithms [7] usually work on the principle of pixel prediction, the error of which is subject to entropic coding, or a dictionary compression method followed by an entropic encoder. Some formats use a transformation instead of an explicit predictor to decorrelate pixels. Then we talk about residue coding instead of prediction error. However, the principle is the same. The entropic encoder can be a simple (Golomb-Rice encoder) as well as a very specialized arithmetic encoder coupled with context modeling. The paragraphs below describe the individual lossless compression methods in detail.

PNG (1995) [11, 12] is a purely lossless format initially intended to replace the GIF format. It is based on a predictor followed by the dictionary compression method DEFLATE (a combination of LZ77 and Huffman coding), which is usually implemented by the zlib library. Several tools can test different predictors and different parameterizations of the DEFLATE method. This will achieve an even better compression ratio than the zlib at the cost of several orders of magnitude slower compression. The zopflipng tool is used in our comparison with the parameters `-iterations=500 -filters=01234mepb`.

JPEG-LS (1998) [17] is a lossless image format created as a replacement for the lossless JPEG mode, which is inefficient. This compression method consists of a predictor and a subsequent context entropic coder (Golomb-Rice coder). The coder can also encode a sequence of

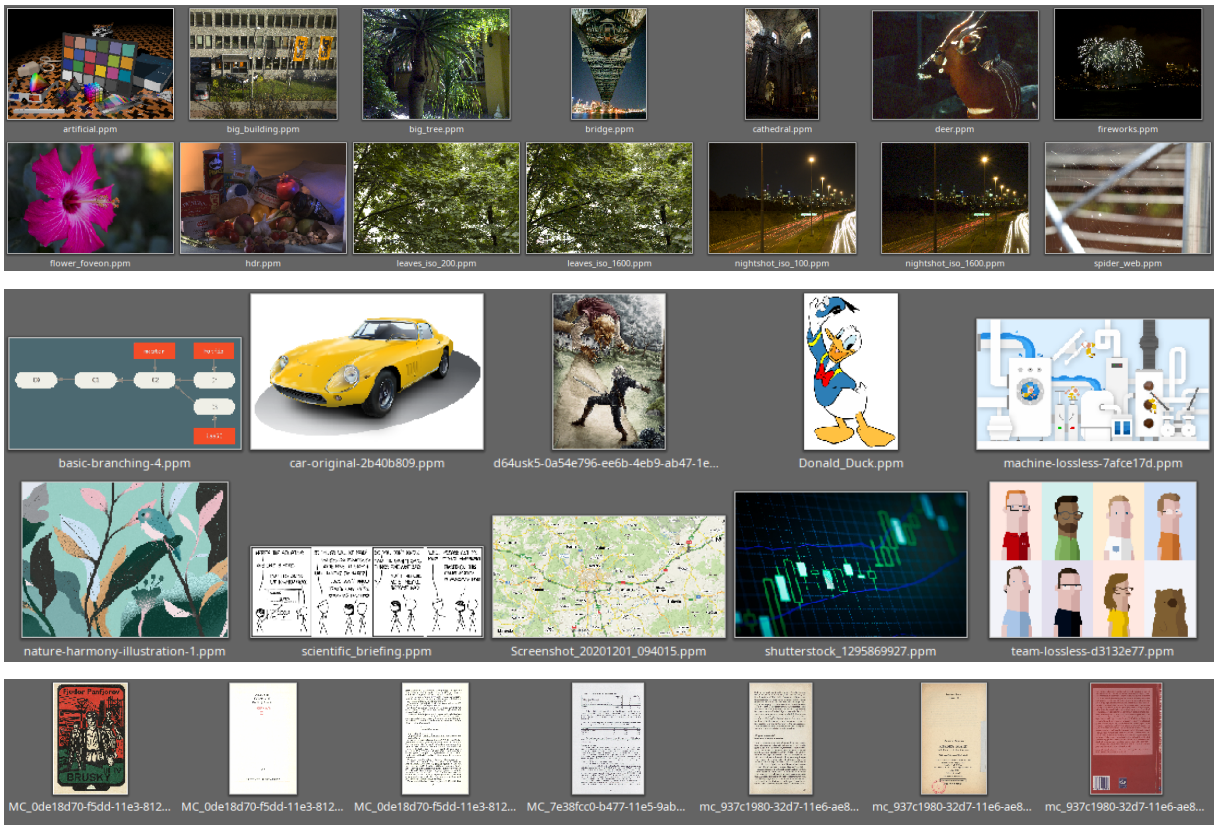


Figure 1: Three datasets used in our comparison. From top: Photos (14 images), Illustrations (10 images), and Books (7 images).

the same pixels by a variant of the RLE method. We use the libjpeg 1.58 library from Thomas Richter for compression with parameters `-ls 0 -c`.

JPEG 2000 [16] is a format based on a discrete wavelet transform [8] followed by a context arithmetic coder. This makes it different from most other lossless compression methods, which are based on prediction error coding. The format allows the choice of lossless or lossy compression paths. Compression is incomparably faster than PNG. There are several implementations of this format, of which we consider the Kakadu commercial library to be the best. Kakadu version 8.0.5 is used for compression with parameters `Cblk={64,64} Stiles={8192,8192} Creversible=yes Clayers=1 Clevels=5`.

JPEG XR (HD Photo) [4] is a 2009 format developed by Microsoft. The format offers a lossless compression mode based on hierarchical discrete cosine transform (DCT) and Huffman coding. As with JPEG 2000, the processing steps are the same for both lossless and lossy coding. Reference software 1.41 with default parameters is used for compression.

WebP [5] is a 2010 compression format developed by Google. The format allows the choice of lossy or lossless compression. The lossless variant is based on a predictor followed by a combination of LZ77 and Huffman coding. The cwebp reference tool with the `-lossless` parameter is used for compression. WebP 2 is the successor of the WebP image format, currently in development. The cwp2 reference tool with the `-effort 9 -q 100` parameters is used in our comparison.

H.265 (also HEVC) [15] is a 2013 video compression format that supports lossless compression. This format is again based on spatial prediction (without the use of DCT). The format is a classic representative of hybrid video compression. HEIC and BPG image formats are based on their Intra profile. We use the x265 library version 3.4 for compression through the FFmpeg framework with parameters `-c:v libx265 -preset placebo -x265-params lossless=1`.

FLIF [13, 14] is a lossless image format from 2015, strongly inspired by the FFV1 format. The format is based on a predictor and a sophisticated arithmetic encoder MANIAC. As with PNG, the compression ratio can be improved by searching for various parameters. We use the flifcrush tool for this. FLIF development has stopped since FLIF is superseded by JPEG XL.

AVIF is based on the AV1 [6] video format from 2018. Also, this format is a classic representative of hybrid video compression. The format offers both lossy and lossless compression. The format uses a predictive-transform coding scheme for lossless mode, where the

prediction comes from intra-frame reference pixels. The residuals undergo a 2D transform, and then they are entropy coded using arithmetic coding. In comparison, the libavif version 0.8.3 library with the `-l -c aom` parameters (libaom 2.0.0) was used.

JPEG XL (2020) [1, 2] is a new image format that supports both lossy and lossless compression. ISO/IEC standard is still under development. Its basic building blocks include transform, prediction, context modeling, and entropy coding (ANS [3] or Huffman coding). We use a reference software version 0.3.3 with `-q 100 -s 9` parameters.

A dataset of 14 high-resolution photographic images [10] in RGB24 format was used for comparison. Thus, the uncompressed image has a bit rate of 24 bpp (bits per pixel). The individual images are shown in Figure 1. The dataset occupies a total of 157 megapixels. We will refer this dataset to it as Photos. We also tested the compression performance on illustrations (artificial images, still 24 bpp). This dataset occupies 10 megapixels. This dataset was composed of images found on Google. We will refer to it as Illustrations. The last dataset used in our paper consists of seven scanned book's pages in 24 bpp and comprises 27 megapixels. The scans were obtained from the collection of the Digital Library of the National Library of the CR. This dataset will be called Books.

3 METHOD AND RESULTS

We use two quantities to evaluate the compression efficiency: a bitrate and compression ratio. The bitrate is defined as the number of bits per single image pixel. The compression ratio is defined as the ratio between the uncompressed and compressed file size. The results are shown in Tables 1, 2, and 3.

We evaluated the compression performance of all mentioned image formats on all three datasets. It turned out that the clear winner is the FLIF format. It is even more efficient than the newer JPEG XL. Even though the development of FLIF was stopped in favor of JPEG XL. It is worth noting that an exhaustive search found the FLIF format parameters. JPEG XL has proven to be the second-best choice. WebP 2 is also relatively efficient.

The victory of the FLIF format is also in line with recent results of Rahman *et al.* In their paper [9], the authors compared the compression performance of the lossless JPEG, JPEG 2000, PNG, JPEG-LS, JPEG XR, CALIC, HEIC, AVIF, WebP, and FLIF formats. They concluded that the FLIF is optimal for all types of 24 bpp images, which they evaluated.

Interestingly, all algorithms achieve a compression ratio about three times higher on the Illustrations dataset than on Photos and about two times higher than on Books. It also says that the Photos dataset is the hardest to compress for lossless algorithms, whereas the Illustrations are the easiest task.

We thought even compare the computing needs of the individual formats. However, such a comparison would not be fair for two reasons. Firstly, for some formats, we use an exhaustive search for suitable parameters; for others, we do not. Secondly, compression and decompression ran in parallel on different machines with different hardware.

Format	Bitrate [bpp]	Compression Ratio
PNG	11.461131	2.094034 : 1
JPEG-LS	10.588847	2.266535 : 1
JPEG 2000	10.620645	2.259749 : 1
JPEG XR	11.575239	2.073391 : 1
WebP	10.619910	2.259906 : 1
H.265	12.460111	1.926146 : 1
FLIF	**9.318886**	**2.575415** : 1
AVIF	12.039541	1.993431 : 1
WebP 2	10.007292	2.398251 : 1
JPEG XL	9.433458	2.544135 : 1

Table 1: Compression performance on the Photos dataset. The best result in bold.

Format	Bitrate [bpp]	Compression Ratio
PNG	4.628656	5.185090 : 1
JPEG-LS	6.994191	3.431419 : 1
JPEG 2000	6.094012	3.938292 : 1
JPEG XR	8.097638	2.963827 : 1
WebP	4.294325	5.588771 : 1
H.265	7.158238	3.352780 : 1
FLIF	**3.394439**	**7.070387** : 1
AVIF	8.198868	2.927233 : 1
WebP 2	3.621495	6.627097 : 1
JPEG XL	3.473148	6.910157 : 1

Table 2: Compression performance on the Illustrations dataset. The best result in bold.

Format	Bitrate [bpp]	Compression Ratio
PNG	8.694597	2.760334 : 1
JPEG-LS	8.394430	2.859038 : 1
JPEG 2000	7.303011	3.286315 : 1
JPEG XR	8.989639	2.669740 : 1
WebP	7.451733	3.220727 : 1
H.265	10.326125	2.324201 : 1
FLIF	**6.084763**	**3.944278** : 1
AVIF	10.398287	2.308072 : 1
WebP 2	6.890983	3.482812 : 1
JPEG XL	6.216347	3.860788 : 1

Table 3: Compression performance on the Books dataset. The best result in bold.

4 CONCLUSIONS

We compared the compression efficiency of state-of-the-art formats for lossless image compression. The comparison was performed on three different datasets. It has been found that the most efficient lossless compression method is the FLIF format. And this despite the fact that its development has stopped since JPEG XL supersedes FLIF. Furthermore, it turned out that the JPEG XL has proven to be the second-best choice. The third in line is the WebP 2 format. It should be noted that some formats are still under development.

Acknowledgements

This work has been supported by the Technology Agency of the Czech Republic (TA CR) project AI for Traffic and Industry Vision (no. TH04010144) and Progressive Image Processing Algorithms (no. TH04010394).

REFERENCES

[1] Alakuijala, J., van Asseldonk, R., Boukortt, S., Szabadka, Z., Bruse, M., Comsa, I.-M., Firsching, M., Fischbacher, T., Kliuchnikov, E., Gomez, S., Obryk, R., Potempa, K., Rhatushnyak, A., Sneyers, J., Szabadka, Z., Vandervenne, L., Versari, L., and Wassenberg, J. JPEG XL next-generation image compression architecture and coding tools. In Tescher, A. G. and Ebrahimi, T., editors, *Applications of Digital Image Processing XLII*. SPIE, Sept. 2019. doi: 10.1117/12.2529237.

[2] Alakuijala, J., Boukortt, S., Ebrahimi, T., Kliuchnikov, E., Sneyers, J., Upenik, E., Vandevenne, L., Versari, L., and Wassenberg, J. Benchmarking JPEG XL image compression. In Schelkens, P. and Kozacki, T., editors, *Optics, Photonics and Digital Technologies for Imaging Applications VI*. SPIE, Apr. 2020. doi: 10.1117/12.2556264.

[3] Duda, J., Tahboub, K., Gadgil, N. J., and Delp, E. J. The use of asymmetric numeral systems as an accurate replacement for Huffman coding. In *2015 Picture Coding Symposium (PCS)*. IEEE, May 2015. doi: 10.1109/pcs.2015.7170048.

[4] Dufaux, F., Sullivan, G. J., and Ebrahimi, T. The JPEG XR image coding standard [standards in a nutshell]. *IEEE Signal Processing Magazine*, 26 (6):195–204, Nov. 2009. doi: 10.1109/msp.2009.934187.

[5] Google. Compression techniques, 2020. URL `https://developers.google.com/speed/webp/docs/compression`.

[6] Han, J., Li, B., Mukherjee, D., Chiang, C.-H., Grange, A., Chen, C., Su, H., Parker, S., Deng, S., Joshi, U., Chen, Y., Wang, Y., Wilkins, P., Xu, Y., and Bankoski, J. A technical overview of AV1. *Proceedings of the IEEE*, pages 1–28, 2021. doi: 10.1109/jproc.2021.3058584.

[7] Lina J., K. Chapter 16 - lossless image compression. In Bovik, A., editor, *The Essential Guide to Image Processing*, pages 385–419. Academic Press, Boston, 2009. ISBN 978-0-12-374457-9. doi: 10.1016/B978-0-12-374457-9.00016-0.

[8] Mallat, S. *A Wavelet Tour of Signal Processing: The Sparse Way. With contributions from Gabriel Peyre*. Academic Press, 3 edition, 2009. ISBN 9780123743701.

[9] Rahman, M. A., Hamada, M., and Shin, J. The impact of state-of-the-art techniques for lossless still image compression. *Electronics*, 10(3):360, Feb. 2021. doi: 10.3390/electronics10030360.

[10] Rawzor. The new test images, 2015. URL `https://imagecompression.info/test_images/`.

[11] Roelofs, G. *PNG: The Definitive Guide*. 1999. ISBN 1-56592-542-4.

[12] Salomon, D. *Data Compression: The Complete Reference*. Springer New York, 2004. ISBN 9780387406978.

[13] Sneyers, J. and Wuille, P. FLIF: Free lossless image format based on MANIAC compression. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2016. doi: 10.1109/icip.2016.7532320.

[14] Soferman, N. FLIF, the new lossless image format that outperforms PNG, WebP and BPG, 2015. URL `https://cloudinary.com/blog/flif_the_new_lossless_image_format_that_outperforms_png_webp_and_bpg`.

[15] Sze, V., Budagavi, M., and Sullivan, G. *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Integrated Circuits and Systems. Springer International Publishing, 2014. ISBN 9783319068954.

[16] Taubman, D. S. and Marcellin, M. W. *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer, 2004. ISBN 978-1-4613-5245-7. doi: 10.1007/978-1-4615-0799-4.

[17] Weinberger, M., Seroussi, G., and Sapiro, G. The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. *IEEE Transactions on Image Processing*, 9(8): 1309–1324, 2000. doi: 10.1109/83.855427.