The 20th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision

in co-operation with

EUROGRAPHICS

W S C G ' 2012

Conference Proceedings

Part II

Plzen

Czech Republic

June 26 - 28, 2012

Co-Chairs

Enhua Wu, University of Macau & Chinese Academy of Sciences, China
Vaclav Skala, University of West Bohemia, Czech Republic

Edited by
Vaclav Skala

Vaclav Skala – Union Agency

WSCG'2012 Conference Proceedings

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

Managing Editor: Vaclav Skala

Published and printed by:
Vaclav Skala – Union Agency
Na Mazinách 9
CZ 322 00 Plzen
Czech Republic

Hardcopy: *ISBN 978-80-86943-79-4*

WSCG 2012

International Program Committee

WSCG 2012

Board of Reviewers

Murtagh,F. (Ireland)
Myszkowski,K. (Germany)
Niemann,H. (Germany)
Okabe,M. (Japan)
Oliveira Junior,P. (Brazil)
Oyarzun Laura,C. (Germany)
Pala,P. (Italy)
Pan,R. (China)
Papaioannou,G. (Greece)
Paquette,E. (Canada)
Pasko,A. (United Kingdom)
Pasko,G. (United Kingdom)
Pastor,L. (Spain)
Patane,G. (Italy)
Patow,G. (Spain)
Pedrini,H. (Brazil)
Peters,J. (United States)
Peytavie,A. (France)
Pina,J. (Spain)
Platis,N. (Greece)
Plemenos,D. (France)
Poulin,P. (Canada)
Puig,A. (Spain)
Reisner-Kollmann,I. (Austria)
Renaud,c. (France)
Reshetov,A. (United States)
Richardson,J. (United States)
Rojas-Sola,J. (Spain)
Rokita,P. (Poland)
Rudomin,I. (Mexico)
Runde,C. (Germany)
Sacco,M. (Italy)

Sadlo,F. (Germany)
Sakas,G. (Germany)
Salvetti,O. (Italy)
Sanna,A. (Italy)
Santos,L. (Portugal)
Sapidis,N. (Greece)
Savchenko,V. (Japan)
Sellent,A. (Germany)
Sheng,B. (China)
Sherstyuk,A. (United States)
Shesh,A. (United States)
Schultz,T. (Germany)
Sirakov,N. (United States)
Skala,V. (Czech Republic)
Slavik,P. (Czech Republic)
Sochor,J. (Czech Republic)
Solis,A. (Mexico)
Sourin,A. (Singapore)
Sousa,A. (Portugal)
Sramek,M. (Austria)
Staadt,O. ()
Stroud,I. (Switzerland)
Subsol,G. (France)
Sunar,M. (Malaysia)
Sundstedt,V. (Sweden)
Svoboda,T. (Czech Republic)
Szecsi,L. (Hungary)
Takala,T. (Finland)
Tang,M. (China)
Tavares,J. (Portugal)
Teschner,M. (Germany)
Theussl,T. (Saudi Arabia)

Tian,F. (United Kingdom)
Tokuta,A. (United States)
Torrens,F. (Spain)
Triantafyllidis,G. (Greece)
TYTKOWSKI,K. (Poland)
Umlauf,G. (Germany)
Vavilin,A. (Korea)
Vazquez,P. (Spain)
Vergeest,J. (Netherlands)
Vitulano,D. (Italy)
Vosinakis,S. (Greece)
Walczak,K. (Poland)
WAN,L. (China)
Wang,C. (Hong Kong SAR)
Weber,A. (Germany)
Weiss,G. (Germany)
Wu,E. (China)
Wuensche,B. (New Zealand)
Wuethrich,C. (Germany)
Xin,S. (Singapore)
Xu,D. (United States)
Yang,X. (China)
Yoshizawa,S. (Japan)
YU,Q. (United Kingdom)
Yue,Y. (Japan)
Zara,J. (Czech Republic)
Zemcik,P. (Czech Republic)
Zhang,X. (Korea)
Zhang,X. (China)
Zillich,M. (Austria)
Zitova,B. (Czech Republic)
Zwettler,G. (Austria)

WSCG 2012

Communications Proceedings

Contents

Kenwright,B.: A Beginners Guide to Dual-Quaternions : What They Are, How They Work, and How to Use Them for 3D Character Hierarchies 1

Hast,A., Marchetti,A.: An Efficient Preconditioner and a Modified RANSAC for Fast and Robust Feature Matching 11

Vassilev,T.I., Spanlang,B.: Fast GPU Garment Simulation and Collision Detection 19

Saito,P.T.M., de Rezende,P.J., Falcao,A.X., Suzuki,C.T.N., Gomes,J.F.: Improving Active Learning with Sharp Data Reduction 27

Morik,M., Masik,S., Müller,R., Köppen,V.: Exposing Proprietary Virtual Reality Software to Nontraditional Displays 35

Lee,G.R. , Lee,H.C. , Lee,T.M. , Yoon,G.H.: Image Abstraction with Cartoonlike Shade Representation 45

Klein,A., Nischwitz,A., Obermeier,P.: Contact Hardening Soft Shadows using Erosion 53

Schmidt,M., Guthe,M., Blanz,V.: Diffusion-based parametrization of surfaces on 3D-meshes 59

Minoi,J.-L., Gillies,D.F., Robert,A.J.: Realistic Facial Expression Synthesis of 3D Human Face based on Real Data using Multivariate Tensor Methods 69

Kurowski,M.: Procedural Generation of Meandering Rivers Inspired by Erosion 79

Schiffner,D., Krömker,D.: Parallel Treecut-Manipulation for Interactive Level of Detail Selection 87

Nakata,N., Kakimoto,M., Nishita,T.: Animation of Water Droplets on a Hydrophobic Windshield 95

Noguera,J.M., Jimenez,J.R.: Visualization of Very Large 3D Volumes on Mobile Devices and WebGL 105

Crumley,Z., Marais,P., Gain,J.: Voxel-Space Shape Grammars 113

Tang,Y., Wu,Z., Zhou,M.: Interactively Simulating Fluid based on SPH and CUDA 123

Lazunin,V., Savchenko,V.: Artificial jellyfish: evolutionary optimization of swimming 131

Pimenta,W., Santos,L.P.: A Comprehensive Taxonomy for Three-dimensional Displays 139

Metzgar,J., Semwal,S.K.: Approximating the Fire Flicker effect using Local Dynamic Radiance Maps 147

Schumann,M., Hoppenheit,J., Müller,S.: A Matching Shader Technique for Model-Based Tracking 155

Tomori,Z., Gargalik,R., Hrmo,I.: Active Segmentation in 3D using Kinect Sensor 163

Marks,S., Windsor,J., Wuensche,B.: Using Game Engine Technology for Virtual Environment Teamwork Training 169

Leonardi,V., Mari,J.L., Vidal,V., Daniel,M.: A Morphing Approach for Kidney Dynamic Modeling : from 3D Reconstruction to Motion Simulation 179

Debelov,V., Kozlov,D.: Rendering of Translucent Objects, Verification and Validation of Algorithms 189

Klein,A., Tappert,B., Nischwitz,A., Obermeier,P.: Volumetric Percentage Closer Soft Shadows 197

Klicnar,L., Beran,V.: Robust Motion Segmentation for On-line Application 205

Oshita,M.: Multi-Touch Interface for Character Motion Control Using Example-Based Posture Synthesis 213

Hulík,R., Kršek,P.: Local Projections Method and Curvature Approximation on 3D Polygonal Models 223

Bahnsen,C., Dewilde,A., Pedersen,C., Tranchet,G., Madsen,C.B.: Realtime global illumination using compressed pre-computed indirect illumination textures 231

Reuter,A., Seidel,H.-P., Ihrke,I.: BlurTags: spatially varying PSF estimation with out-of-focus patterns 239

Nguyen,M.H., Wuensche,B., Delmas,P., Lutteroth,C.: 3D Models from the Black Box: Investigating the Current State of Image-Based Modeling 249

Krivokuca,M., Wuensche,B., Abdulla,W., Lavoué,G.: Investigating the Rate-Distortion Performance of a Wavelet-Based Mesh Compression Algorithm by Perceptual and Geometric Distortion Metrics 259

Bittorf,B., Wüthrich,C.: EmotiCon - Interactive emotion control for virtual characters 269

Mahiddine,A., Seinturier,J., Jean-Marc Boi,J.-M., Drap,P., Merad,D.: Performances Analysis of Underwater Image Preprocessing Techniques on the Repeatability of SIFT and SURF Descriptors 275

Bruni,V., Rossi,E., Vitulano,D.: Unsupervised Perception-based Image Restoration of Semi-transparent Degradation using Lie Group Transformations 283

Bugaj,M., Cyganek,B.: GPU Based Computation of the Structural Tensor for Real-Time Figure Detection 291

Safdar,K.: Detecting and Removing Islands in Graphics-Rendering-Based Computations of Lower Envelopes of Plane Slabs 299

Minich,C.: Search for small monostatic polyhedra 309

Warburton,M., Maddock,S.: Creating Animatable Non-Conforming Hexahedral Finite Element Facial Soft-Tissue Models for GPU Simulation 317

François,A., Raffin,R., Aryal,J.: 3D modelling and analysis: ISO standard tools for air traffic 327

Jawad,M., Yasin,M., Sarfraz,M.S.: License Plate Detection using NMF with Sparseness constraints through still Images 335

Bian,X., Krim,H.: Video-based Human Motion Analysis: An Operator-based Approach 341

Arora,N., Kumar,A., Kalra,P.: Digital Restoration of Old Paintings 347

Khurana,S., Brener,N, Benger,W., Karki,B., Roy,S., Acharya,S., Ritter,M., Iyengar,S.: Multi Scale Color Coding of Fluid Flow Mixing Indicators along Integration Lines 357

Qureshi,H., Malik,M., Ahmad,M.A., Heinzl,C.: Benchmarking of De-noising Techniques for Streaking Artifacts in Industrial 3DXCT Scan Data 367

Graca,S., Oliveira,J.F., Realinho,V.: WorldPlus: An Augmented Reality Application with Georeferenced content for smartphones - the Android example 377

Volumetric Percentage Closer Soft Shadows

Andreas Klein, Björn Tappert, Alfred Nischwitz
Munich University of Applied Sciences
Lothstrasse 64

80335 Munich, Germany
andreas.klein@hm.edu, bjoern.tappert@gmx.de, nischwitz@cs.hm.edu

Paul Obermeier
MBDA Deutschland GmbH
Hagenauer Forst 27

86529 Schrobenhausen, Germany
paul.obermeier@mbda-systems.de

ABSTRACT

Percentage Closer Soft Shadows is a popular technique to generate contact hardening soft shadows with shadow mapping. Recent research in shadow generation for translucent objects makes it possible to realize shadows for translucent objects in real-time environments. However, for multiple translucent blockers it is unclear how an appropriate blocker depth can be calculated. In this paper, we propose a method to calculate a blocker depth for multiple translucent blockers and therefore, enabling physically plausible soft shadows for opaque and translucent objects in a single approach.

Keywords
Soft Shadows, Contact Hardening Soft Shadows, PCSS, Volumetric Shadows, AVSM

1 INTRODUCTION

Shadow Mapping is a popular method to generate shadows for opaque objects in real-time rendering. The idea is to realize a visibility test by comparing the depth value as seen from the camera with the depth value stored in a depth map.

As shadow mapping assumes point light sources, only hard shadows will be produced. However, it is possible to simulate soft shadows of area light sources by making multiple shadow tests within a filter window and averaging the result. However, these shadows are not physically plausible as the shadows are uniformly soft. In order to achieve physically plausible shadows, the size of the penumbra must be adapted according to the distance between a light blocker and a shadow receiver. These shadows are called contact hardening soft shadows, as the shadow softness increases with the blocker-receiver distance. Contact hardening soft shadows can be realized with shadow mapping by adapting the filter window based on the blocker - receiver distance.

Recent work in shadow generation for translucent objects makes it possible to integrate shadows for translucent objects in real-time environments, such as games. In contrast to shadow maps, which only store the nearest depth value as seen from the light, approaches for translucent shadows store a transmittance function per pixel (Figure 1). A transmittance function encodes the light visibility at each depth value.

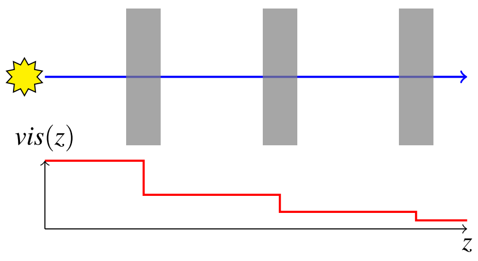


Figure 1: The light intensity is reduced as it passes through a set of translucent blockers. A transmittance function encodes the transmittance at each given z value.

Analogous to shadow mapping, a transmittance function can be used to generate shadows. A shadow exists, if the light intensity of a point light source is reduced by one or multiple translucent blockers. However, these shadows have hard boundaries, despite they appear to be soft compared to opaque blockers since these shadows have a reduced darkness according to the blocker's translucency. As with shadow mapping, it is possible to simulate soft shadows of area light sources by making multiple comparisons within a filter window. However, contact hardening soft shadows requires the distance between a blocker and a receiver to estimate the penumbra width which is used to adapt the filter window. For multiple translucent blockers, it is unclear how an appropriate blocker depth can be calculated.

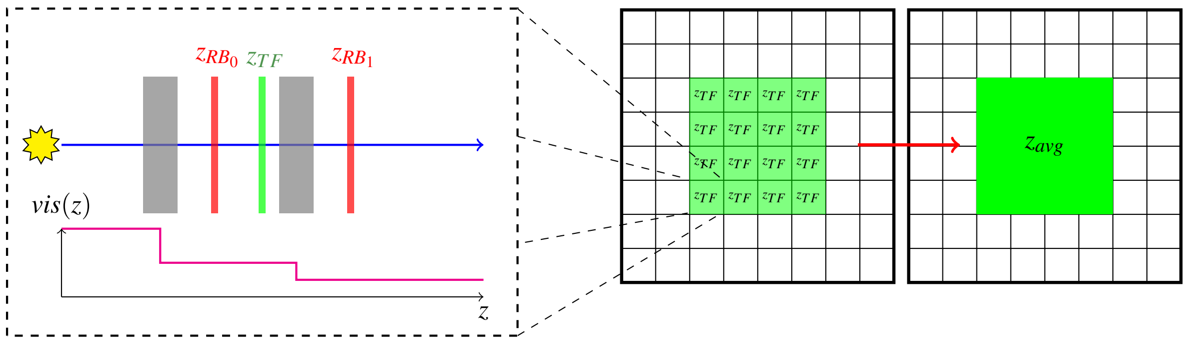


Figure 2: Our algorithm proceeds as follows. First, we sample a single transmittance function and calculate a replacement blocker z_{RB} (red) for each translucent blocker. Second, we compute a weighted average z_{TF} (green) for all replacement blockers of a transmittance function. We repeat these steps for each transmittance function in a filter window and calculate a total replacement blocker z_{avg} for the filter window. Finally, we use the total replacement blocker to estimate the penumbra size.

In this paper, we propose a method to calculate a blocker depth for multiple translucent blockers and therefore, enabling physically plausible soft shadows for opaque and translucent objects in a single approach.

2 RELATED WORK

We focus our review on publications closely related to our work. See Eisemann et al. [Eis11a] for a comprehensive survey on other shadow algorithms.

Shadows from Opaque Blockers

Percentage-closer filtering (PCF) [Ree87a] is a popular method for generating soft shadows. The idea is to build a shadow factor by making multiple shadow comparisons within a user defined filter window. Fernando [Fer05a] proposed percentage-closer soft shadows (PCSS) in order to generate contact hardening soft shadows with PCF. The idea is to first search for blockers within a given filter window and calculate an average blocker depth. A penumbra width can then be estimated using a parallel planes approximation. The penumbra width is used to scale the PCF window.

Shadows from Translucent Blockers

Deep Shadow Maps [Lok00a] stores nodes of a transmittance function per pixel and compresses them to guarantee a fixed absolute error. Salvi et al. [Sal10a] uses in their Adaptive Volumetric Shadow Maps (AVSM) an area based metric to compress a transmittance function with a fixed number of nodes. Several approaches use a basis transformation to compress a transmittance function, e.g. [Jan10a, Del11a].

Depth peeling [Eve01a, Liu06a, Bav08a] uses multiple render passes with dual depth comparison to extract the depth layers of geometry. However, this approach suffers from an unbounded number of rendering passes for complex geometry. Stochastic Transparency [End11a, McG11a] uses a randomized sub-pixel stipple pattern to realize screen door transparency with a fixed set of Multi Sample Anti Aliasing (MSAA) samples.

3 ALGORITHM

Based on the parallel planes approximation of Fernando [Fer05a], we derive an average blocker depth for a list of translucent blockers. We assume that the transmittance functions have already been generated in each frame using a translucent shadow technique, such as AVSM [Sal10a].

The algorithm proceeds as follows (Figure 2). First, we sample a single transmittance function in order to receive two pairs of transmittance - depth values, which represent an extended translucent blocker (Figure 3). From these values we calculate an infinitesimal thin replacement blocker described by a single transmittance - depth pair. Second, we integrate over all samples of a single transmittance function and thus over all blockers along a ray from the light source by computing a weighted average of all replacement blockers. Third, we process step one and two for a set of transmittance functions within a filter window to estimate a total replacement blocker and calculate its depth value. Fourth, we derivate a penumbra width with a parallel planes approximation [Fer05a]. Finally, we generate the shadows by making multiple shadow tests and averaging the result.

Shadows from Translucent Blockers

In order to calculate a shadow factor for translucent blockers with shadow mapping, a modification to the binary shadow test function is necessary.

The intensity of the light is reduced by the transmittance value of a blocker. For a single translucent blocker with depth z and an alpha value α the translucent shadow test function is given by:

$$S_T(z) = (1-\alpha) + \alpha S(z_L, z_S)$$

where z_L is the blocker depth transformed into light's coordinate system (light space), z_S is the value in the depth map and $S(z_L, z_S)$ is the binary shadow test function:

$$S(z_L, z_S) = \begin{cases} 0, & \text{if } z_L > z_S \\ 1, & \text{if } z_L \leq z_S \end{cases}$$

For multiple overlapping blockers, the translucent shadow test function can be applied using the over operator [Por84a]:

$$S_T(z_i) = (1-\alpha_i)S_T(z_{i-1}) + \alpha S(z_{i_L}, z_{i_S}) =$$
$$= \sum_{m=0}^{i}\left[\alpha_{i-m}S(z_{i-m_L}, z_{i-m_S})\prod_{n=i-m+1}^{i}(1-\alpha_n)\right]$$

Estimate a Replacement Blocker for a Single Transmittance Function

In contrast to standard shadow mapping approaches, multiple translucent blockers are available for each pixel and each would produce a different penumbra. In order to calculate a contact hardening soft shadow with a parallel planes approximation, such as in PCSS, an average blocker depth must be estimated. Our idea is to replace multiple blockers with one appropriate replacement blocker. First, we show how a replacement blocker can be calculated for a single translucent blocker and then how they can be averaged for multiple blockers.

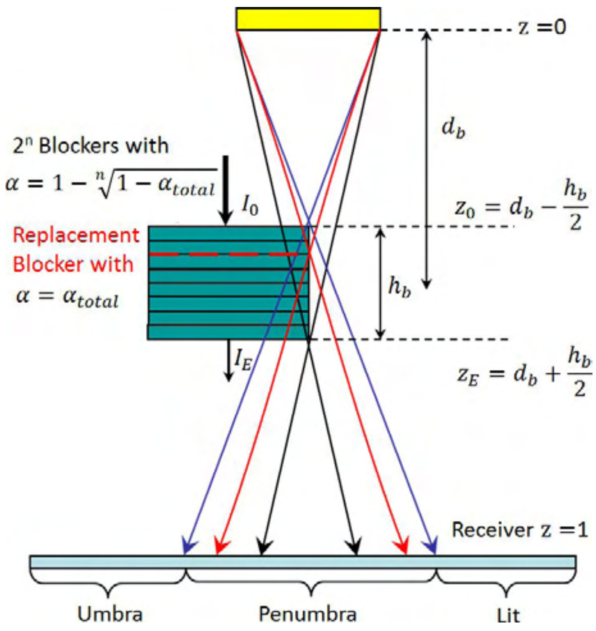

z =0
d_b
2ⁿ Blockers with
$\alpha = 1 - \sqrt[n]{1-\alpha_{total}}$
I_0
$z_0 = d_b - \frac{h_b}{2}$
Replacement Blocker with
$\alpha = \alpha_{total}$
h_b
I_E
$z_E = d_b + \frac{h_b}{2}$
Receiver z =1
Umbra
Penumbra
Lit

Figure 3: Our idea is to replace a blocker of dimension h_b with n thin layers and compute a replacement blocker (depicted as a red dashed line) with the total transmittance $(1-\alpha_{total})$.

We assume that a single blocker has a constant absorption with the coefficient α which reduces the light intensity I_0 (Figure 3):

$$I_E = (1-\alpha)I_0 \tag{1}$$

In a homogenous material, the reduction of the light intensity can be expressed using the entry depth of a blocker z_0 and an absorption coefficient b:

$$I(z) = I_0 \cdot e^{-b(z-z_0)} \tag{2}$$

Due to the exponential reduction, we place the replacement blocker at the position where the light intensity is reduced by half of the difference value:

$$\left(\frac{I_E - I_0}{2}\right) \tag{3}$$

We choose this position for the replacement blocker, as the mean squared error to a reference solution is smaller compared to a medium position (see Section 3).

The depth of the replacement blocker is then given by

$$z_{RB} = z_0 - \frac{1}{b} \cdot ln\left(\frac{I_E + I_0}{2 \cdot I_0}\right) \tag{4}$$

with:

$$b = -\frac{1}{z_E - z_0} \cdot ln(1-\alpha) \tag{5}$$

We express the depth of the replacement blocker for a single translucent blocker using the alpha value α:

$$z_{RB} = z_0 - (z_E - z_0) \cdot \frac{ln\left(1 - \frac{\alpha}{2}\right)}{ln(1-\alpha)} \tag{6}$$

In order to obtain a single replacement blocker depth z_{TF} for a transmittance function, which may consists of multiple translucent blockers, we calculate an weighted average of all replacement blockers:

$$z_{TF} = \frac{\sum_{i=1}^{n} \Delta_i \cdot z_{RB_i}}{\sum_{i=1}^{n} \Delta_i} \tag{7}$$

where Δ_i is a measure for the part of the light intensity which is blocked by the i-th replacement blocker:

$$\Delta_i = \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \tag{8}$$

Analysis of the Approximation

One way to calculate PCSS for multiple translucent blockers is to evaluate the PCSS function for each sample on a transmittance function T:

$$S_f(z_R) \approx \int_{i \in T} \alpha_i PCSS(z_R, z_i) vis(z_i) dz \tag{9}$$

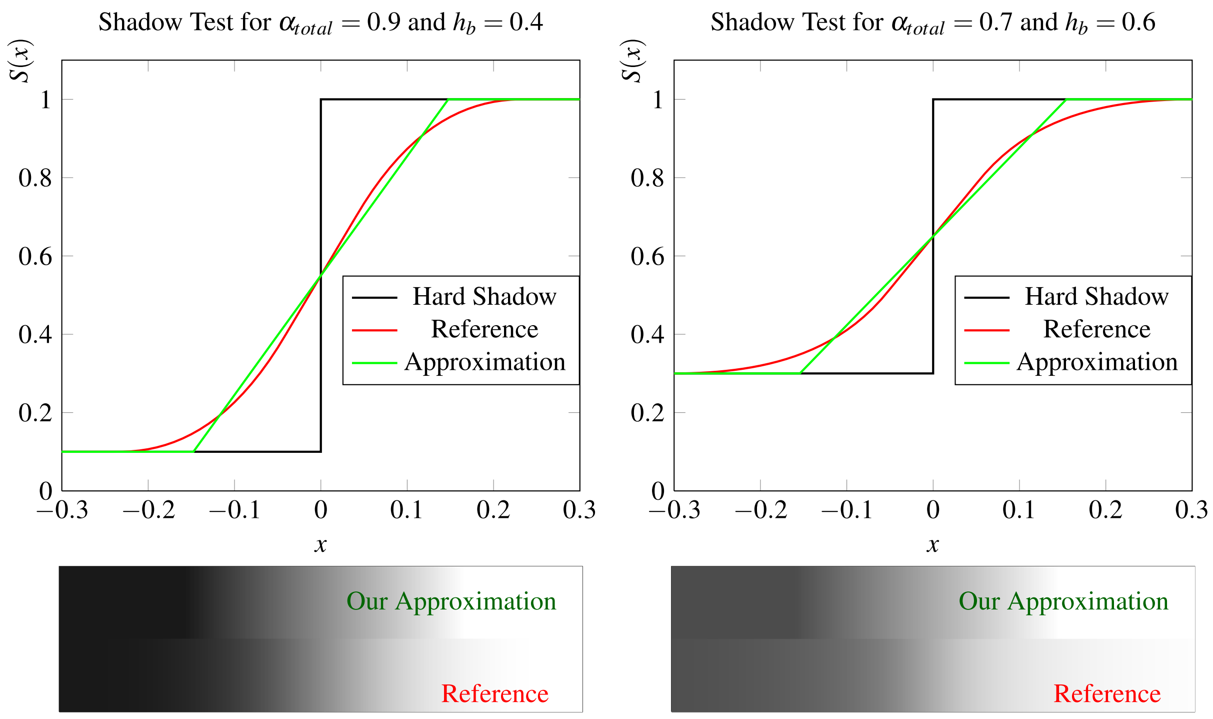


Figure 4: Comparison of the shadow test of our approximation that uses a single replacement blocker against a reference solution with n infinitesimal blockers. The plots at the bottom displays the resulting shadow factors at a shadow boundary.

with:

$$vis(z) = \prod_{z_j<z} (1-\alpha_j) \tag{10}$$

$$PCSS(z_R, z_B) = \omega_L \frac{z_R - z_B}{z_B} \sum_{n=0}^{k} f(z_n - z_r) S(z_n, z_R) \tag{11}$$

where f is a filter function, ω_L the diameter of the light source, z_R the depth of the receiver and z_B the depth of the blocker. However, this solution requires a shadow factor computation for each single blocker. Instead, we approximate this integral by using a single replacement blocker with total alpha:

$$S_f(z_R) \approx \left(\sum_i \alpha_i \right) PCSS(z_R, z_{TF}) \tag{12}$$

In Figure 4, we compare this approximation of the shadow test function against a reference solution (Eq. 9) for different blocker sizes and alpha values. It can be observed, that our approximation calculates a linear transition between penumbra and lit regions and vice versa where a smooth transition is correct.

Figure 5 compares the shadow test function when choosing different depths for a replacement blocker. By using a replacement blocker at half of the absorption, the mean squared error to the reference solution is smaller compared to a shadow test function that uses a medium depth for a replacement blocker.

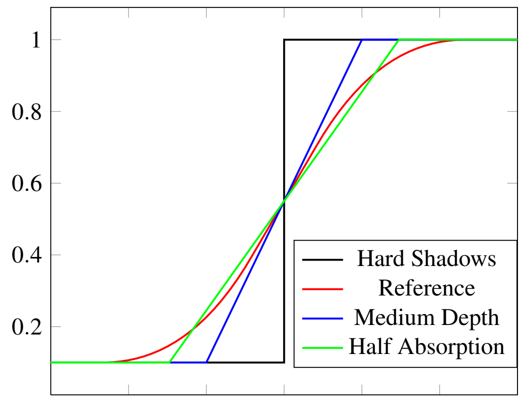


Figure 5: We choose the position where the light intensity is reduced by half of the difference value $\frac{I_E - I_0}{2}$ (half absorption), as it approximates the reference solution more accurate.

As our approximation is based on PCSS, we introduce the same limitations, such as overestimating the penumbra area.

4 IMPLEMENTATION

We implemented our algorithm using Adaptive Volumetric Shadow Maps [Sal10a].

Our implementation proceeds as follows. First, we render an AVSM in each frame. Second, we sample the AVSM within the given search radius to calculate an average blocker depth. We terminate the algorithm, if there are no pixels with a transmittance smaller than one. Otherwise, we calculate a replacement blocker for each transmittance function in the given search radius and calculate the average depth as follows:

$$z_{avg} = \frac{\sum_{i=1}^{n} \Delta_i \cdot z_{TF_i}}{\sum_{i=1}^{n} \Delta_i} \tag{13}$$

In the next step, we estimate the penumbra width using the parallel planes approximation [Fer05a]:

$$\omega_{Penumbra} = \frac{(z_R - z_{avg})}{z_{avg}} \omega_L \tag{14}$$

Finally, we generate the shadow factor by comparing multiple samples and averaging the result.

5 RESULTS

We used 8 nodes for the transmittance function in the AVSM implementation. The shadows of our algorithm were generated using 25 Poisson samples for the blocker search as well as in the final shadow factor computation. The AVSM size was 1024 x 1024 and the screen resolution was 1680 x 1050. Table 1 shows the performance results and Figure 7 compares the visual results. The reference solution is realized by replacing the area light source with 128 point light sources. Figure 6 shows an example scene with opaque and translucent blockers. The performance results were obtained on an Intel Xeon E5620 CPU with 2.4 GHz, 8 GB RAM and a NVIDIA GeForce GTX 680 graphics card with 2048 MB memory.

	AVSM Hard Shadows	VPCSS Soft Shadows
Dragon (Fig. 7a) (871K tris)	5.1 ms	10.1 ms
Particle (Fig. 7b) (5K particles)	3.8 ms	9.4 ms
Hairball (Fig. 7c) (2.88M tris)	26.3 ms	33.4 ms
Tank (Fig. 6) (136K tris 5K particles)	4.3 ms	10.9 ms

Table 1: Performance results in milliseconds with a screen resolution of 1680 x 1050. Note that these timings also include the generation of the transmittance functions with AVSM.

6 CONCLUSIONS AND FUTURE WORK

We presented a method to calculate an average blocker depth for multiple translucent blockers. Analog to standard shadow mapping, a contact hardening soft shadow

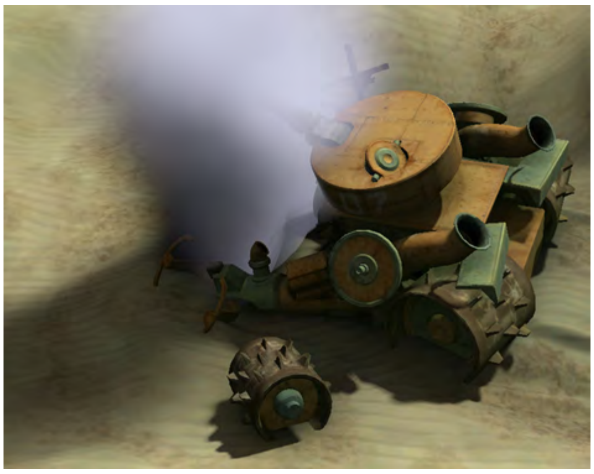


Figure 6: Example scene with soft shadows resulting from opaque and translucent blockers.

can now be generated by using a constant amount of shadow tests. The results show that our algorithm creates physically plausible soft shadows for opaque and translucent blockers.

For future work we wish to investigate how a replacement blocker can be calculated for wavelength-dependent transmissive blockers.

7 ACKNOWLEDGMENTS

We thank the anonymous reviewers for their useful comments. The work of A. Klein is funded by MBDA Deutschland GmbH.

8 REFERENCES

[Bav08a] Bavoil, L. and Meyers, K. Order independent transparency with dual depth peeling. NVIDIA technical report, 2008.

[Del11a] Delalendre, C., Gautron, P., Marvie, J.E. and Francois, G. Transmittance Function Mapping. In Conf.proc of Interactive 3D Graphics and Games 2011. 2011.

[Eis11a] Eisemann E., Schwarz M., Assarsson U., Wimmer M. Real-Time Shadows, Taylor & Francis, 2011.

[End11a] Enderton, E., Sintorn, E., Shirley, P. and Luebke, D. Stochastic Transparency. IEEE Transactions on Visualization and Computer Graphics August 2011. 2011.

[Eve01a] Everitt, C. Interactive order-independent transparency. NVIDIA white paper, 2001.

[Fer05a] Fernando R. Percentage-Closer Soft Shadows. ACM SIGGRAPH 2005 Sketches, 2005.

[Jan10a] Jansen, J. and Bavoil, L. Fourier Opacity Mapping. In Conf.proc of Interactive 3D Graphics and Games 2010. 2010.

[Liu06a] Liu, B., Wei, L.-Y, and Xu, Y.-Q. Multi-layer depth peeling via fragment sort. Microsoft Research technical report. 2006

[Lok00a] Lokovic, T and Veach, E. Deep Shadow Maps. In Conf.proc of SIGGRAPH 00. 2000.

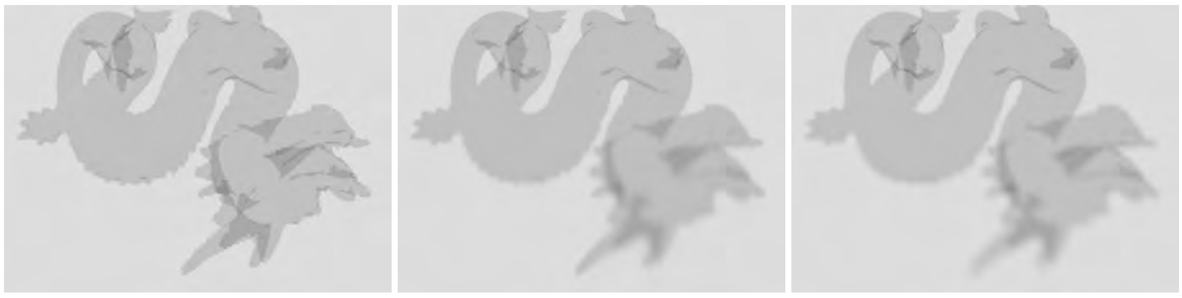
[McG11a] McGuire, M. and Enderton, E. Colored Stochastic Shadow Maps. In Conf.proc of Interactive 3D Graphics and Games 2011. 2011.

[Por84a] Porter, T and Duff, T. Composing digital images. In Conf.proc. SIGGRAPH 84, 1984.

[Ree87a] Reeves W. T., Salesin D. H., Cook R. L. Rendering antialiased shadows with depth maps. In Conf.proc SIGGRAPH 87, ACM, 283-291, 1987.

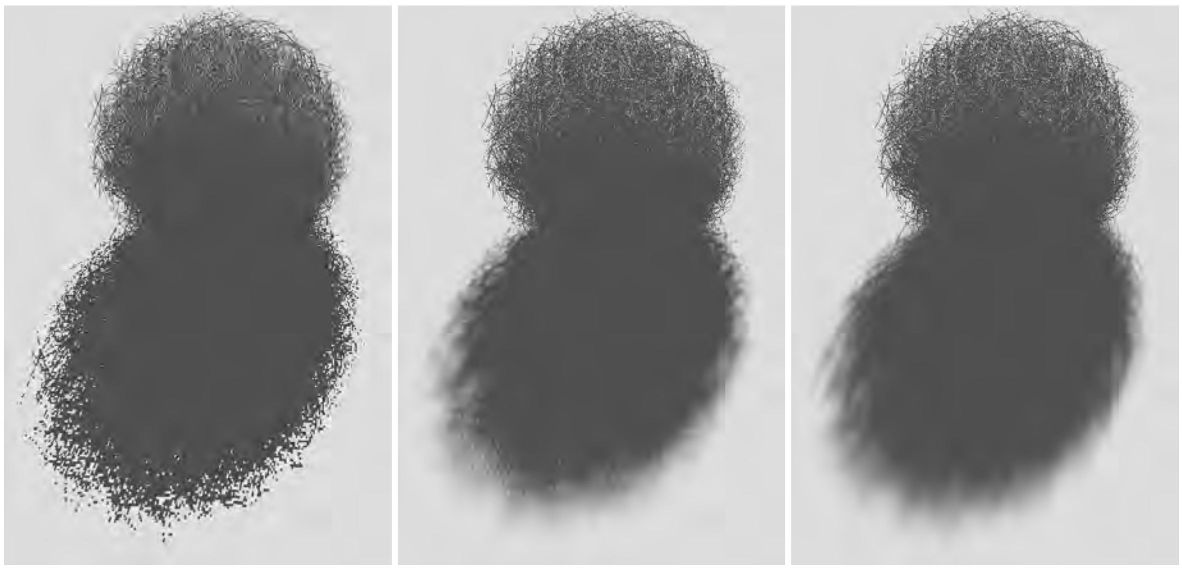
[Sal10a] Salvi, M., Vidimce, K., Lauritzen, A. and Lefohn A. Adaptive Volumetric Shadow Maps. In Conf.proc. EGSR 2010. 2010

[Wil78a] Williams L. Casting curved shadows on curved surfaces. In Conf.proc. SIGGRAPH 78, ACM, 270-274, 1978.



(a)

(b)



(c)

Figure 7: Resulting shadows from the dragon (a), particle (b) and hairball (c) datasets. From left to right: Hard shadows, our algorithm and reference solution. (a) The dragon was rendered with $\alpha = 0.3$. (b) In the particle dataset, the shadows resulting from a hard shadow test look already soft. However, our algorithm as well as the reference solution softens the shadows further, as the blocker-receiver distance increases. (c) The hairball dataset was rendered with $\alpha = 1.0$.

Robust motion segmentation for on-line application

Lukáš Klicnar, Vítězslav Beran

Brno University of Technology
Faculty of Information Technology
Department of Computer Graphics and Multimedia
Božetěchova 2, 612 66 Brno, CZ

xklicn00@stud.fit.vutbr.cz, beranv@fit.vutbr.cz

ABSTRACT

This paper presents an approach for on-line video motion segmentation. Common methods were designed for off-line processing, where time to process one frame is not so important and varies from minutes to hours. The motivation of our work was an application in robotic perception, where a high computational speed is required. The main contribution of this work is an adaptation of existing methods to a higher computational speed and on-line processing. The proposed approach is based on sparse features, we utilized the KLT tracker to obtain their trajectories. A RANSAC-based method is used for initial motion segmentation, resulting motion groups are partitioned by a spatial-proximity constraints. The correspondence of motion groups across frames is solved by one-frame label propagation in forward and backward directions. Finally, an approximation of dense image segmentation is obtained by using the Voronoi tessellation.

Keywords

Motion segmentation, moving objects detection, KLT tracker, Voronoi tessellation.

1. INTRODUCTION

The automatic pre-processing of digital content is getting high importance over the last decade. The growing importance is accelerated by the amount of video recordings, visual surveillance data or multimedia content that are easily acquired and shared. Such amount of digital data is of very limited use when only pixel-level knowledge is available.

All mentioned scenarios are convenient applications for off-line processing. Present applications of digital content analysis are usually efficient data storage, indexing, image or video retrieval, content-based copy detection, semantic indexing, etc. Such solutions pre-process the data in off-line stage, when the computational performance has low priority. Developed techniques are then designed to work off-line in general. The promising results of developed off-line techniques lead our research to design a method that processes the video stream in on-line manner. This is motivated by needs in applications like TV-broadcast monitoring, assistive systems utilizing a machine vision or robotic perception where operating in real-time is a crucial demand.

Presented work is focused to develop and evaluate a pre-processing method for video segmentation. Based on state-of-the-art methods that were developed for off-line processing, we design an approach for on-line video-content segmentation by common motion constraints. The video content is usually described by set of key-frames and any high-level processing is than applied to key-frames separately. The temporal attribute naturally included in video sequence is then suppressed or back-projected by high-level methods for wide base-line matching. Our method is designed to describe the video content in on-line manner by i) spatial segments in each frame using common motion constraints and ii) correspondences of spatial segments in temporal domain.

The rest of the paper is organized as follows: Section 2 review of existing methods extracting the information from video sequences in spatio-temporal domain. The segmentation method is introduced in Section 3. We illustrate the performance of the method and achieved results in Section 4. Finally, in Section 5 the proposed method and its possible extensions are discussed.

2. RELATED WORK

The spatio-temporal segmentation and clustering of region trajectories is related to the research topics about interesting region detection, segmentation and tracking. Numerous existing methods dealing with video motion segmentation are designed to work off-line. The motion segmentation methods based on frame-two-frame optical flow [Shi98] [Cre05] [Won02] evaluate a motion difference between objects of two adjacent frames. The methods' locality in temporal domain tends to over-segment cluttered scenes or to lose the tracks when fast motion appears. The approach based on tracking of local patches has appeared in structure-from-motion tasks [Fit00] [Rot07] where the focus is more on 3D object model building and only rigid objects are considered. The motion segmentation was applied for matching, recognition and retrieval tasks [Siv06] where not only rigid objects are considered. The existing approaches also widely differ in density of trajectories from quite sparse [Siv06] [Bas08] to highly dense [Fra09] [Che09] [Bro10] sampling. Some previous works are closely related to extract spatio-temporal segments of particular objects where the training stage takes place to adopt the method to particular object type and temporal behavior, e.g. pedestrians [Lei05] [Rod07]. The problem of object segmentation in changing environments and moving backgrounds has been addressed also in robotic field [Bea11]. Based on probabilistic models, the knowledge of the robot's motion is used to determine the shape and location of objects. In contrary to our method, the knowledge of the robot's motion constrains the usage of this method for robotic application only.

The all previous related work are quite similar in the computational cost where most of the approaches work off-line and the time needed to process the frame range from minutes to hours. One of the promising works solving the problem of unsupervised on-line video segmentation [Vaz10] can effectively handle long sequences, create and terminate labels as the video is processed, and still preserve the photometric consistency of the segmentation across several frames.

3. SEGMENTATION

The proposed approach can be divided into these four consecutive blocks: i) feature detection and tracking, ii) initial motion segmentation, iii) object extraction and tracking, and iv) dense segmentation approximation. The method is based on a sparse feature tracking, because all image points cannot be processed, if a high computational speed is demanded. Trajectories of these features are input for the motion segmentation. At first, initial motion segmentation is estimated by a RANSAC-based robust algorithm, which results to groups of tracks with a similar motion. Groups are then partitioned to satisfy spatial proximity constraints of associated tracks. These steps provide a local label for each track, which is valid for a particular frame only, so their frame-to-frame correspondence has to be solved. This is called object extraction and tracking. Finally, the Voronoi tessellation is used to obtain a dense image segmentation to overcome the low density of sparse features. A block diagram of the whole system is in Figure 1.

Feature detection and tracking

One of the most important parts of this approach is a robust tracking method, which provides "good" trajectories. This means that they are as long as possible and they don't contain outliers. The length of trajectories has an influence on correct object tracking, while the low occurrence of outliers is essential for good motion segmentation. An outlier in this case means point of measured trajectory, which significantly differs from the actual position.

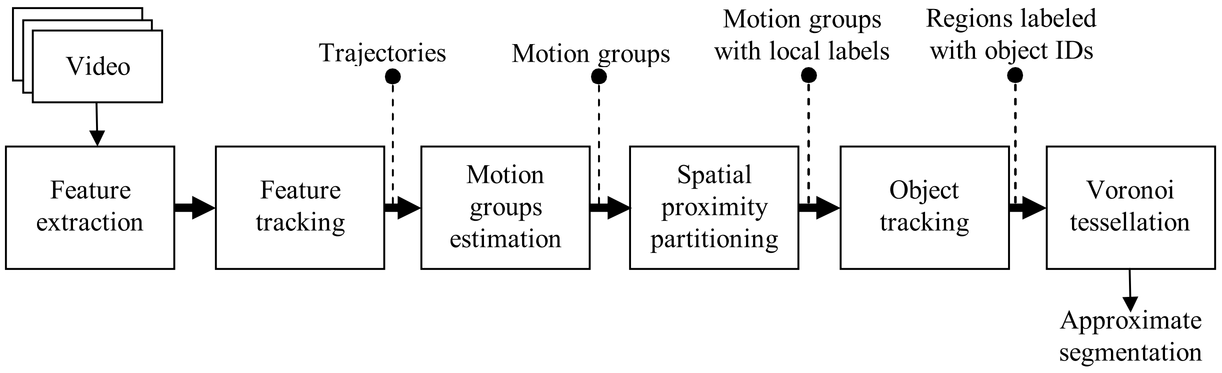


Figure 1. A block diagram of proposed system for motion segmentation.

We found tracking with the KLT tracker [Tom91] based on optical flow very suitable for this task. It doesn't match features from two frames, but it rather searches for a best occurrence in the image directly. This builds an invariance to the stability of used feature detector. Features are still detected in every frame, but they are used only for a continuous adding of new tracks. These aspects imply that any feature detector should be suitable. In the proposed approach, points provided by the Harris corner detector [Har88] are used. Features are detected on image scales of 1x, 1/2x, 1/4x and they are all merged and tracked together. This improves the ability to segment some moving objects, which are too fast for current shutter speed and appear blurred in the image – in undersampled image, the blurred structure becomes sharper and more features are detected.

The disadvantage is that the accuracy decreases with the scale, so a point detected e.g. at scale 1/4x is projected on 1x with an error of 4px. This didn't show as a problem, because the KLT tracker working on image pyramids handles these points correctly. In addition, matched features are checked by normalized cross correlation and if it's below threshold (values of 0.8-0.9 are used), the particular track is terminated instead of being extended. That results to trajectories with no visible outliers. In opposite to [Siv06] [Bas08], there is no need for short-range track repair, we found KLT tracking sufficient.

Initial motion segmentation

The initial motion segmentation is performed by a RANSAC based algorithm [Siv06] [Bas08], you can see a block diagram in Figure 2. It progressively extracts groups of features with a similar motion, a one group is extracted in every iteration. Four-tuples of tracks are randomly chosen to estimate a homography matrix that represents the motion between the current and the previous frame. A total reprojection error is then computed, that forms the criterion for suitability of found transformation – the goal is to find the homography corresponding with the lowest reprojection error. This is done iteratively until the maximal number of iterations is reached. In opposite to [Bas08], iterating can be stopped also when a desired reprojection error is satisfied (we use condition that average reprojection error is lower than 1.0px). This founds a sufficient transformation on average after 11 iterations. Compared to 100-200 iterations (which is the maximal number), this significantly increases the computational speed.

Inlying tracks (with reprojection error below 3.0px) are then removed as a single motion group and the rest of them goes through the same process in a next iteration. When a sufficient transformation is found, the corresponding homography is recomputed using inliers only. Every motion group is then divided to meet the spatial proximity constraint, that the distance between any point from this group and its nearest neighbour from the same group has to be lower than a threshold, so the group becomes spatially-compact. This step is not necessary, but it

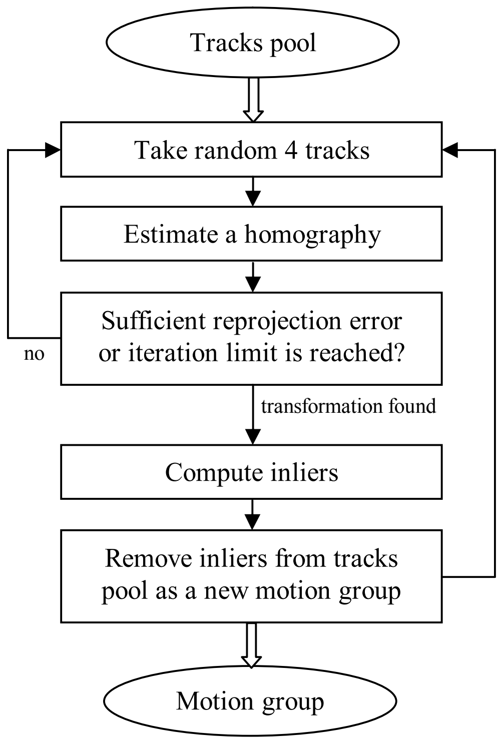

Tracks pool
Take random 4 tracks
Estimate a homography
Sufficient reprojection error or iteration limit is reached?
no
transformation found
Compute inliers
Remove inliers from tracks pool as a new motion group
Motion group

Figure 2. RANSAC motion segmentation.

allows recognizing distant objects moving in the similar way. The spatial proximity constraint may cause an oversegmentation, but this is handled by the next stage, object extraction and tracking.

Object extraction and tracking

From the previous step, every track in a motion group has a local label, which corresponds with a motion segment in a particular frame. For moving objects tracking, the frame-to-frame correspondence of these labels has to be solved. This has to be considered to be 1:N, because several motion groups from the current frame can match a one group from the previous frame (this may be caused e.g. by oversegmentation). A motion group correspondence across frames is solved with label propagation. In [Bas08], authors developed a method of propagation in forward and backward direction, which can handle situations, when motion groups split and merge, and it labels them correctly.

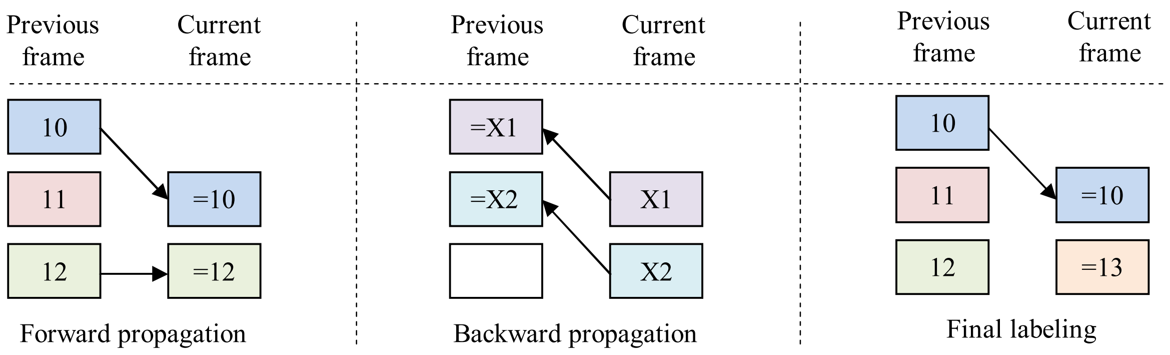


Figure 3. Label propagation in both directions and final labeling. Rectangles represent motion groups with their labels. New label (13) is generated, because groups assigned from both directions differ.

Unfortunately, this violates the condition of on-line processing, so only a limited version of this approach can be used.

The limitation consists in restricting the label propagation to a one frame, i.e. between the current and the previous frame, but in both directions. First, labels from the previous frame are propagated forwards, into the current frame. For every local label from the current frame, a corresponding label from the previous frame is found – that is the label of the group, which has the highest number of common tracks with the currently processed group. Subsequently, a new, temporary label for each group in the current frame is generated and propagated in the same manner, but in the opposite direction (backwards). Labels are then merged for each pair of groups from the current and the previous frames: The same label (from the forward labeling) is assigned only if both groups correspond together in both directions, otherwise a new one is generated (see Figure 3). New labels aren't assigned immediately to motion groups, but only if the same new label is stably suggested for at least 2-3 frames. This helps to overcome the problem that a motion group can occasionally disappear (tracks are incorrectly assigned to another group with different label), which tends to be stable for max. 1-2 frames.

Dense segmentation approximation

The disadvantage of a sparse approach is the low density of points, for which an object label is provided. That information is known only for points, where one of the tracked features occurs. To overcome this problem, the Voronoi tessellation is used to obtain an approximate dense segmentation of the whole image. The positions of tracks in the current frame are used as a cell generators, all pixels in each cell are then labeled with the same label as the track that generated the cell. This divides the image into segments, where points are labeled according to their closest track, where the motion information is known.

4. RESULTS

The evaluation of motion segmentation is a relatively difficult task, because it needs videos with masks of all moving objects, which is very time-consuming to obtain. There are very few dataset available that can be used for exact comparison of motion segmentation methods.

Dataset and evaluation description

To evaluate the proposed method, we used The Berkeley Motion Segmentation Dataset [Bro10]. It consists of a total of 26 videos: (i) 10 sequences of cars, which are rigid objects with predictable movement (ii) 13 sequences of Marple detective stories, which contains mostly people. These are non-rigid, relatively slowly moving objects, which may event stop for a while (iii) 2 sequences of people and (iv) 1 sequence from tennis match. Most of the objects in this dataset are people, typical sizes are at least 20 % of frame area. Lighting condition doesn't change significantly, but most of the videos are obtained by a moving camera. Sequences don't contain cuts.

This dataset consists of a 4243 frames, 189 frames are annotated with a pixel-precise mask. The annotation is dense in space but sparse in time, approximately every tenth frame is annotated. The resolution of frames varies from 350x288 (detective stories) to 640x480 (cars, people, tennis). Examples are in Figure 5.

Authors of this dataset also include methodology and tools for evaluation, which are even obligatory to use when evaluating on this dataset. This means that results are directly comparable to other algorithms. The following list is a free quotation from [Bro10] – for detailed information, please refer to it. The provided evaluation tool measures these parameters:

Density – the number of points for which a cluster label is provided over the total number of image points. Higher density means that more information from the image is extracted.

Overall clustering error – the number of bad labels over the total number of labels on a per-pixel basis. The tool automatically assigns clusters to regions from annotation, all points covering their assigned region are counted as good labels and all others count as bad labels.

Average clustering error – similar to the overall error but averages across regions after computing the error for each region separately. It is usually much higher than the overall error, which is caused by incorrect detection of small regions.

Over-segmentation error – corresponds with the number of clusters merged to fit the regions from the annotation.

Number of objects extracted with less than 10 % error. A background region is excluded.

Results

Experiments with a different length of videos were made (only the first 10, 50, 200 or all frames were processed). In Table 1, the results are shown. The low density is caused by setting of the used feature detector. It can be set to detect more features, but this significantly slows the computation and it is not necessary, dense image segmentation by the Voronoi tessellation extends the density to 100% coverage. Although the results are presented without the tessellation, to show the performance of a motion segmentation itself. The high segmentation error is caused by the characteristics of the dataset. More than a half of objects include people, slowly moving non-rigid objects, which are relatively difficult to segment and track with this method. Slow movement causes that the object falls below the RANSAC threshold ant it gets merged with the background group. The on-line manner prevents to label the tracks according to knowledge if they will move faster and be part of different motion group in the future. By visual evaluation of videos containing moving cars only (they usually move significantly faster), the results would be better.

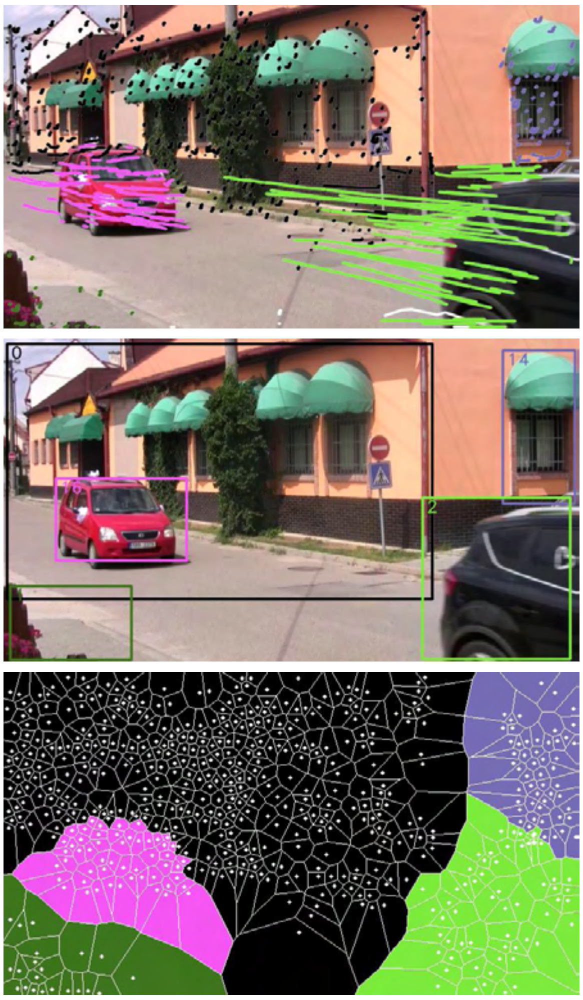


Figure 4. Motion segmentation of scene with two moving cars. From top: tracks, bounding boxes, Voronoi tessellation of the whole image.

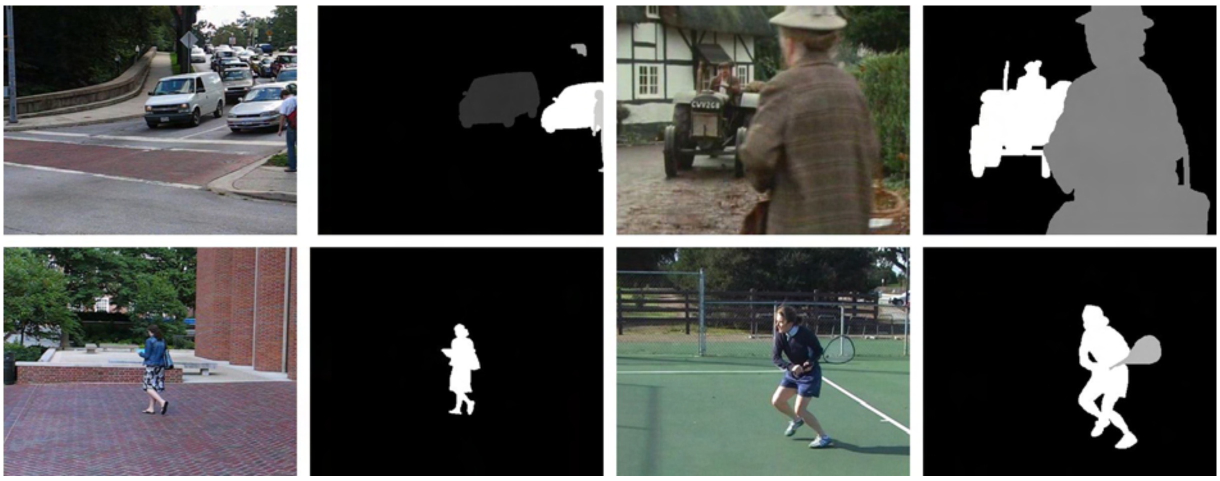


Figure 5. Examples of frames and their annotations from the Berkeley Motion Segmentation Dataset.

	Density	Overall error	Average error	Over-segment.	Extracted objects
First 10 frames (26 sequences)					
This method	**0.14%**	**19.68%**	**40.37%**	**3.87**	**14**
Brox and Malik	3.34%	7.75%	25.01%	0.54	24
GPCA	2.98%	14.28%	29.44%	0.65	12
LSA	2.98%	19.69%	39.76%	0.92	6
RANSAC	2.98%	13.39%	26.11%	0.50	15
ALC corrupted	2.98%	7.88%	24.05%	0.15	26
ALC incomplete	3.34%	11.20%	26.73%	0.54	19
First 50 frames (15 sequences)					
This method	**0.13%**	**20.22%**	**52.39%**	**2.40**	**3**
Brox and Malik	3.27%	7.13%	34.76%	0.53	9
ALC corrupted	1.53%	7.91%	42.13%	0.36	8
ALC incomplete	3.27%	16.42%	49.05%	6.07	2
First 200 frames (7 sequences)					
This method	**0.17%**	**19.39%**	**46.89%**	**4.29**	**3**
Brox and Malik	3.43%	7.64%	31.14%	3.14	7
ALC corrupted	0.20%	0.00%	74.52%	0.40	1
ALC incomplete	3.43%	19.33%	50.98%	54.57	0
All available frames (26 sequences)					
This method	**0.13%**	**19.41%**	**41.53%**	**3.54**	**13**
Brox and Malik	3.31%	6.82%	27.34%	1.77	27
ALC corrupted	0.99%	5.32%	52.76%	0.10	15
ALC incomplete	3.29%	14.93%	43.14%	18.80	5

Table 1. Results of the motion segmentaion on the Berkeley Motion Segmentation Dataset. The performance of other methods is taken from [Bro10], for their description, please refer to the original paper.

The results in Table 1 show that the segmentation error is higher than other methods. The overall performance is comparable to the ALC, but the main advantage is the computational speed, which will be discussed further. The proposed method is more sufficient for detection of rigid and not too slowly moving (frame-to-frame motion larger than threshold for inliers estimation in motion segmentation part) objects. The Berkeley dataset mostly contains moving people, which are objects that violate these conditions and this causes the overall increase of the segmentation error. The high average error means that the method doesn't work correctly on large amount of objects (people in this dataset), but it works reasonable on some types (cars). This method can better detect and larger objects, e.g. cars in the foreground, rather than smaller objects. Sometimes it can detect objects parts (wheels of the car, legs or hands of people), but it is capable to hold them only for a several frames.

The dense segmentation obtained by Voronoi tessellation is only very approximate. Results in Table 2 show that the values are not dependent on the density of tracks. The most of approximation errors is caused by image areas with low density, where the Voronoi cells are large and inaccurately approximate image segments.

Density	Precision	Recall	F-measure
0.06%	0.52	0.71	0.60
0.13%	0.56	0.73	0.63
0.46%	0.61	0.69	0.65

Table 2. Precision, recall and F-measure of dense segmentation for different densities of tracks.

In [Bro10], authors state the computation time for the first 10 frames of the people1 sequence. Unfortunately, they don't mention the parameters of used computer and 10 frames appear to be too little for reasonable comparison. The results are shown in Table 3. We measured the computation time on all sequences of the Berkeley dataset with several densities of tracks. With the low density of 0.13 %, the method could process about 420 tracks/s, but this increases up to approx. 823 tracks/s at a density of 2,0 %. The used computer has the Intel Core2 DuoT7100 CPU at 1.80 Ghz, 4 GB RAM, Windows 7 Professional x64, no GPU acceleration. We don't know the computer used in [Bro10], so these numbers are not directly comparable, but they differ so much, that the speed difference should be obvious.

	Tracks	Time	Tracks/s
Brox and Malik	15486	497s	31.16
ALC	957	22837s	0.042
This method	**Up to approx. 823 tracks/s**		

Table 3. Computation times for the first 10 frames of the people1 sequence from the Berkeley dataset. Results of other methods are taken from [Bro10]. The number of tracks and time for this method is not given, because of the different lengths of used sequences. This method was evaluated on all frames of all sequences of the Berkeley dataset.

On the Berkeley dataset, the motion segmentation part takes average 64.2% of algorithm running time, feature detection takes 10.5% and feature tracking 15.5%, which means that used features and KLT tracking are suitable for high performance motion segmentation. The spatial proximity partitioning takes 6.6% of total computational time and the Voronoi tessellation takes only a 2.9%.

5. Conclusion

The objective of the presented work was to design and evaluate a method for video on-line motion segmentation with demands for low computational cost. The solution is based on sparse feature tracking and RANSAC motion segmentation. The optical flow tracker proved to be very suitable for this task, the most time-demanding part is the motion segmentation, which makes the best candidate for further improvements. The results of evaluation show that the speed-up of this approach is a tradeoff for overall segmentation error. The Voronoi tessellation provides only a very approximate dense segmentation, it will be the one of the subjects of further improvements. But the number of extracted objects and oversegmentation appears to be usable. This method can process video of a standard TV resolution at approximately 1-1.5 fps, which is not yet sufficient for application in real-tim, such is the intended robotic perception. In the current state, this method could be used for example in the tasks involving moving objects detection, instance search, etc., when it is crucial to process to process the data quickly, but not in real-time.

6. Acknowledgments

This work was supported by the Ministry of Education project No. MSM0021630528 and the European Community's 7th Framework Artemis JU grant agreement n. 100233 (R3-COP).

7. References

[Bas08] Basharat, a, Zhai, Y., Shah, M. (2008). Content based video matching using spatiotemporal volumes. *Computer Vision and Image Understanding*, *110*(3), 360-377. doi:10.1016/j.cviu.2007.09.016.

[Bea11] Beale, D., Iravani, P. (2011). Probabilistic models for robot-based object segmentation. *Robotics and Autonomous Systems*, *59*(12), 1080-1089.

[Bro10] Brox, T., Malik, J. (2010). Object segmentation by long term analysis of point trajectories. *European Conference on Computer Vision*, 6315, 282-295. doi:10.1007/978-3-642-15555-0_21.

[Che09] Cheriyadat, A. M., Radke, R. J. (2009).Non-Negative Matrix Factorization of Partial Track Data for Motion Segmentation. *International Conference on Computer Vision*, 865 - 872.

[Cre05] Cremers, D., Soatto, S. (2005). Motion competition: A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, *62*(3), 249-265.

[Fit00] Fitzgibbon, A., Zisserman, A. (2000). Multibody structure and motion: 3-D reconstruction of independently moving objects. *European Conference on Computer Vision*, 891-906.

[Fra09] Fradet, M., Robert, P., Pérez, P. (2009).Clustering point trajectories with various life-spans. *Visual Media Production*.

[Har88] Harris, C., Stephnes, M. (1988). A combined corner and edge detector, *Proceedings of the 4th Alvey Vision Conference*, 147-151.

[Lei05] Leibe, B., Seemann, E., Schiele, B. (2005). Pedestrian Detection in Crowded Scenes. *Computer Vision and Pattern Recognition*, *1*, 878-885. Ieee. doi:10.1109/CVPR.2005.272.

[Rod07] Rodriguez, M. D., Mubarak, S. (2007). Detecting and Segmenting Humans in Crowded Scenes. *International Conference on Multimedia*. doi:10.1145/1291233.1291310.

[Rot07] Rothganger, F., Lazebnik, S., Schmid, C., Ponce, J. (2007). Segmenting, modeling, and matching video clips containing multiple moving objects. *Pattern Analysis and Machine Intelligence*, *29*(3), 477-91. doi:10.1109/TPAMI.2007.57.

[Shi98] Shi, J., Malik, J. (1998). Motion segmentation and tracking using normalized cuts. *International Conference on Computer Vision*, 1154-1160.

[Siv06] Sivic, J., Schaffalitzky, F., Zisserman, A. (2006). Object Level Grouping for Video Shots. *International Journal of Computer Vision*, 67(2), 189-210. doi:10.1007/s11263-005-4264-y.

[Tom91] Tomasi, C., Kanade T. (1991). Detection and Tracking of Point Features. T*echnical Report CMU-CS-91-132*, Carnegie Mellon University.

[Vaz10] Vazquez-Reina, A., Avidan, S., Pfister, H. (2010). Multiple hypothesis video segmentation from superpixel flows. *European conference on Computer vision*, *6315*, 268-281.doi:10.1007/978-3-642-15555-0_20.

[Won02] Wong, Y., Spetsakis, M. (2002). Motion Segmentation and Tracking . International Conference on Vision Interface, , 42(4), 80-87. doi: 10.1016/j.jbiomech.2008.11.038.

Multi-Touch Interface for Character Motion Control Using Example-Based Posture Synthesis

Masaki Oshita

Kyushu Institute of Technology
680-4 Kawazu, Iizuka, Fukuoka
820-8502, Japan

oshita@ces.kyutech.ac.jp

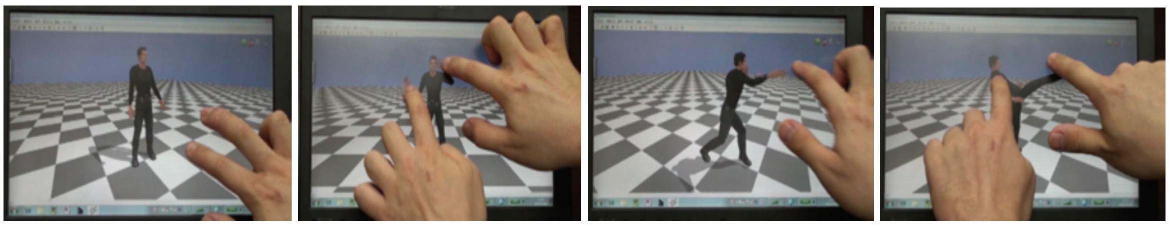


Figure 1. Example of the multi-touch interface for character motion control.

ABSTRACT

We propose a multi-touch interface for character motion control with which a user can control a character's pose freely and make the character perform various actions by simply dragging the character's body parts using a multi-touch input device. We use style-based inverse kinematics to synthesize a natural-looking pose that satisfies given constraints using a learned model of sample postures. However, the style-based inverse kinematics is suited to posture editing and not motion control. It cannot handle various types of actions and cannot generate continuous and physically valid motions. To overcome these limitations, we prepare different learned models for each action and choose an appropriate learned model according to the user's input. More specifically, we use a single learned model for posing and different learned models for each type of action. We also use different motion generation methods for posing and action controls. Furthermore, we introduce tracking control as a post-process after posing and action controls to generate continuous and physically plausible motion. We implemented the proposed method using a Windows 7 Touch API on a multi-touch-enabled personal computer, and demonstrated the power of our interface.

Keywords
Motion control, multi-touch interface, example-based posture synthesis, physics-based control.

1. INTRODUCTION

The flexibility of character motion control in interactive applications is currently very limited. Since common input devices such as a gamepad, mouse or keyboard have only a small number of degrees of freedom, the user can do nothing but simply select an action from a few pre-defined actions by pressing an associated button and make a character perform a fixed action such as walking, punching, or kicking. It is impossible for users to pose the character freely or make the character perform various actions in the user's own style. This is particularly a limitation in some interactive applications such as fighting games, dance animation and online communication using avatars. For example, users may want a character to perform various moves in fighting games and interactive dance animation or use various gestures in communication in a multi-user network environment.

We propose a multi-touch interface for character motion control with which a user can control a character's pose freely and make the character perform various actions by simply dragging the character's body parts using a multi-touch input device (Figure 1).

In theory, using inverse kinematics (IK), a user could control a character's pose in detail. However, with conventional input devices, the user can control only one end effector at a time. Moreover, it is difficult to realize natural-looking motion using IK even with a multi-touch device because multiple body parts must be controlled in a coordinated way to execute an action.

Our method uses style-based IK [GMH*04][SL06]. Style-based IK constructs a learned model of poses in advance by mapping a large number of sample poses onto a low-dimensional latent space. It then synthesizes a natural-looking pose that satisfies given constraints using the learned model. However, style-based IK has several problems. First, it cannot handle many types of actions with a single learned model because appropriate poses depend on the type of action to be expressed. Second, it is difficult to generate continuous motion from given touch strokes of body parts because there is no guarantee that a continuous trajectory in Cartesian space is mapped to a continuous trajectory in latent space. Third, resulting motions that are synthesized using style-based IK lack physical validity, since the character can take any pose at any speed without considering physical constraints. For these reasons, style-based IK cannot be simply used for the motion control of various types of actions. Basically, style-based IK is suitable for making a pose but not for interactive motion control.

To solve these problems, we prepare different learned models for each action and choose an appropriate learned model according to the user's input. More specifically, we use a single learned model for posing and different learned models for each kind of action. We also use different motion generation methods for posing and action controls. During posing control, a pose is generated simply using a learned model based on multi-touch inputs. When a touch stroke for a body part matches the initial trajectory of the primary body part of a prepared action model, the system switches to action control. During action control, we control time progression by considering conditions of sample postures with a learned model to generate continuous and natural-looking motion. We also introduce tracking control as a post-process for posing and action controls to generate continuous and physically plausible motions. Using physics simulation, physical effects such as the shaking of non-controlled body parts by controlled body parts are calculated and applied.

We implemented the proposed method using Windows 7 Touch API on a multi-touch-enabled personal computer, and thus demonstrated the power of our interface.

The rest of this paper is organized as follows. In Section 2, we review related works. Section 3 presents an overview of our method, while Sections 4, 5 and 6 describe posing, action and tracing controls respectively. Results and a discussion are presented in Section 7. Finally, Section 8 concludes the paper.

2. RELATED WORK

Multi-touch Interface for Motion Control

To our knowledge, there are only a few research works that use a multi-touch interface for motion control and generation. Krause et al. [KHS*08] applied conventional IK to a character model based on multi-touch inputs for animation. However, as explained in the previous section, it is difficult to realize complex motions with this approach. Moreover, animation takes a long time and interactive control is impossible. Kip and Nguyen [KN10] proposed a system to control one arm and hand of a character using a multi-touch interface by changing several parameters to blend arm and hand postures. However, their system is limited to the control of one arm and one hand and cannot be used to control full-body motion.

Trajectory-Based Motion Generation

Since the mouse and pen were common input devices well before multi-touch devices became available, there has been much research on the use of a single point or trajectory to control or generate character motion.

A common way to use trajectory for motion control is to use a trajectory to specify a locomotion path [PSS02]. Many animation systems have a function to generate walking and running motions according to a trajectory drawn on the ground. However, with this type of interface, the type of motion is limited to walking or running motion.

Throne et al. [TBvdP04] introduced gesture-based motion selection. Based on the gestures drawn along a trajectory, their system inserts predefined motions such as a jump or flip. Oshita [Osh05] proposed a stroke-based motion selection technique that chooses an appropriate action according to the initial and terminal points of a single stroke drawn on the screen. With these two systems, users can simply select actions by drawing a trajectory or stroke, but postures and the speed of actions are fixed and cannot be controlled.

Igarashi et al. [IMH05] proposed a spatial keyframing animation technique. By placing key poses in the three-dimensional space in advance, a new posture is synthesized by blending the key postures according to the distances between the current mouse position and the key postures. By drawing a trajectory on the screen (moving the mouse cursor), a continuous motion can be generated.

However, to generate natural-looking and intended motion, the key postures must be placed at appropriate positions and at appropriate distances and the user must move the mouse cursor properly. This requires training and careful design. It is also difficult to generate various types of action with a single set of key postures when using this method. Dontcheva et al. [DYP03] used a physical three-dimensional marker to specify body trajectories. However, since the user can control one body part at a time with their system, they need to repeat specifying trajectory for each body part. Their method is for off-line animation making and is not suitable for interactive control.

Style-based IK

As explained in Section 1, style-based IK generates a posture according to user input. The approach is not suitable for motion generation or control. Previous application of style-based IK is limited to posture editing [GMH*04] and motion generation based on trajectory in the latent space [SL06]. Motion generation based on trajectory in three-dimensional space or on a screen was not realized.

Min et al., [MCC09] introduced a statistics-based model of motion instead of postures. Using a given trajectory of a specified body part, sample motions are blended to synthesize new motion. The method can be used with multi-touch devices. However, since entire motions are blended, detailed control such as time and speed control and changing postures is not possible. Moreover, the model must be learned for a specific type of action and the system does not allow the execution of combinations of various types of actions.

Other motion control interfaces

Various devices have been used for interactive motion control by researchers. Some physical sensors such as Wii remote (acceleration sensors) can be used with gesture recognition technique [LLZ*09] [BNT08]. By performing a specific gesture, the user can select an action from predefined actions. However, this type of gesture-based interface simply substitutes for conventional interfaces such as game pad and keyboard. Control of the character's full-body motion and executing action with user's own style are not possible.

Some researches combine physics simulation with a control of limited number of degrees of freedom. Laszlo et al. [LZS05] used mouse to control a few joints and generated full-body motion by using physics simulation. Shiratori and Hodgins [SH08] used Wii remotes to determine a few parameters for physics-based controllers of several actions such as walking, running, and jumping. By using physics simulation, physically plausible motion can be generated and controlled. However, the physics-based controllers must be designed for each type of action in advance and realizing various types of actions and styles is difficult.

Recently, low-price markerless motion capture devices such as Microsoft Kinect have become available. Using such a device, the character's full body motion can be freely controlled [IWZ*09] [Osh06]. However, such a device requires a larger workspace. Moreover, to perform highly dynamic actions, the user must actually perform the actions. This is difficult for non-trained users.

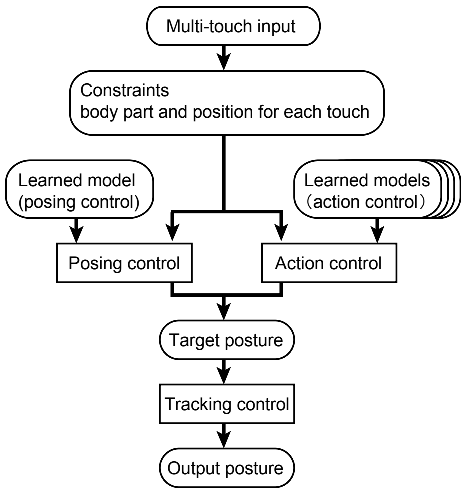


Figure 2. System overview.

3. SYSTEM OVERVIEW

The overview of our system is shown in Figure 2. The system generates a character's posture in each frame according to multi-touch inputs. Any type of multi-touch device can be used with our system. When the user touches a body part of the character and drags it on the screen, each touch input is handled as a constraint to control the character. Note that each constraint is a two-dimensional position on the screen; that is, a half-line in the three-dimensional space.

Our system uses posing and action control modes with different learned models. Under the initial condition, the posing control mode is used to change the character's posture according to multi-touch inputs. When the trajectory of the body part that is dragged by the user matches the trajectory of a predefined action, the system switches to the action control mode to generate an action that dynamically changes according to the multi-touch inputs. When the action finishes, the system switches back to the posing control mode.

In addition to posing and action controls, tracking control is applied as a post-process to change the

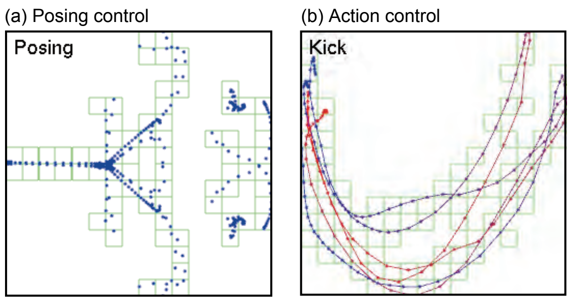


Figure 3. Example of latent space (two-dimensional).

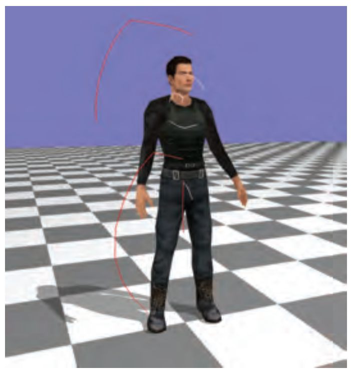


Figure 4. Example of trajectories of action models.

posture generated by posing and action controls so that the generated motion is continuous and physically plausible.

Since our system handles only the touching and dragging of character's body parts, other types of multi-touch inputs such as touching and dragging in another area, and multi-touch gestures including pinching and swiping, can be used for other functions such as viewpoint control depending on the application.

4. POSING CONTROL

The posing control is used to generate character motions that have no particular form such as a change in the standing pose or the free movement of hands and feet. For posing control, we use a method similar to conventional style-based IK. The problems of style-based IK are solved by combining action and tracking controls. In our implementation, we use a method similar to that used by Shin et al. [SL06].

As sample postures for the learned model in posing control, we use postures extracted from motion data for various poses. In addition, the initial period of motion data for action control is also used, since initial parts of actions are realized by posing control before action control is initiated.

Learned model for posing control

To apply style-based IK, sample postures are mapped on a low- (typically two- or three-) dimensional latent space to construct a learned model in advance. There are various ways to map example postures onto a latent space [SL06]. In our implementation, we employed multi-dimensional scaling (MDS). The sample postures are mapped to a latent space so that the distances between sample postures are preserved in the latent space. The distance between any two postures is calculated according to the sum of distances between all pairs of corresponding joints after two postures are aligned according to their pelvis position and orientation in the same manner as in [KGP02].

Posture synthesis during posing control

From the given constraints, which are two-dimensional positions of body parts, a posture is synthesized using the learned model by blending sample postures in the latent space.

We divide the latent space into a grid and assign a representative sample posture to each grid in advance as shown in Figure 3, where each dot represents a sample posture. By comparing the representative sample postures with given constraints, the corresponding cell can be found efficiently. Since the constraints consist of a two-dimensional half line of selected body parts, the distance between given constraints and a sample posture can be calculated from the average of the distance between the half line of the constraints and the position of the corresponding body part in the sample posture. Once the corresponding cell is determined, the distance between each sample posture in the cell and given constraints is calculated to determine the blending weight of each sample posture. By blending the sample postures with the weights, a posture is synthesized. For posture blending, rotation of each join in the sample postures is blended using quaternions [PSS02]. For pelvis position and orientation, the relative position and orientation taken from the initial state in original motion data are blended. The blended position and orientation are added to the character's original position and orientation when the posing control begins.

The synthesized posture is generated by blending sample postures so that the posture satisfies the given constraints as much as possible. However, since the number of sample postures is limited compared with the number of possible posture configurations, there is no guarantee that the constraints are satisfied. Therefore, we further apply numerical IK to the synthesized posture. However, if body parts are moved a large distance, the modified posture may become unnatural. Therefore, we limit the distance to within a certain distance (0.2 m). In addition, the foot positions of the synthesized posture may not match the current posture. We also apply conventional IK to fix the foot position when the foot is on the ground. In our implementation, we use cyclic coordinate descent (CCD) IK [Wel93].

5. ACTION CONTROL

Action control is used to generate a character's motion with a particular form such as punching, kicking, or jumping. We prepare a specific learned model for each action. An action model is selected depending on user inputs. During action control, we control time progression by considering conditions of sample postures to generate continuous and natural-looking motions.

Executing condition for action control

The executing condition for an action model is determined according to the trajectory of a body part on the screen when the user drags the body part.

Each action model has a primary body part (e.g., the right hand for a right-hand punching action) and its average trajectory, which is calculated from the motion data used to train the action model. The primary body part is manually specified. The trajectory of the primary body part taken from each action model is projected onto the screen and compared with the input trajectory.

When the primary body part is dragged by the user, the distance between the user input trajectory and the trajectory of each action model is calculated to determine whether the action is to be initiated. A trajectory is represented by a series of pairs of the two-dimensional screen point and time. To calculate the distance between two trajectories, corresponding points from two trajectories are first determined using a dynamic programming algorithm. The average distance is then calculated from the difference in positions and difference in relative times for each pair of corresponding points:

$$D = \sum_{(i,j)\in S} \left| \mathbf{t}_{\mathrm{input},i} - \mathbf{t}_{\mathrm{action},j} \right| + h \sum_{(i,j)\in S} \left| t_{\mathrm{input},i} - t_{\mathrm{action},j} \right|, \quad (1)$$

where $\mathbf{t}_{\mathrm{input},i}, t_{\mathrm{input},i}, \mathbf{t}_{\mathrm{action},j} t_{\mathrm{action},j}$ are the positions and times of the points for the input and action trajectories, S is the set of corresponding points for two trajectories, and h is a scaling parameter. When the computed distance D is lower than a threshold, the action is initiated. This process is applied only when the input trajectory has certain length and S covers more than half the action trajectory.

In our preliminary user test, we found that users tend to drag a body part quickly when they want to execute an action while they drag body parts slowly for pose control. Therefore, we change the threshold

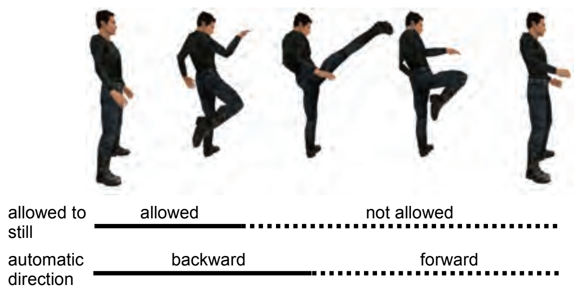


Figure 5. Conditions during an action.

depending on the speed of the touch input. When body parts are moved quickly, an action is initiated even if the distance between the trajectories is large.

The three-dimensional trajectory of each action model is projected onto a screen. Therefore, determining action execution is view-dependent. Depending on the current viewpoint, the trajectory may be normal to the screen and the projected trajectory may be too short. This makes the execution of action difficult and introduces recognition errors. For example, it is difficult to draw the trajectory of the hand for a punching action when the character is facing the viewpoint. To solve this problem, we compute a three-dimensional vector that represents the trajectory of the action model in advance and calculate the angle between the representative vector and viewpoint vector (eye vector). If the angle is smaller than a threshold (45 degrees in our implementation), the action is excluded from the action execution process. For example, when the user wants a character to perform a punching action, the user must control the viewpoint so that the user can see the character from the side or top before the user drags the character's hand, because a punching action cannot be executed when the view is toward the front or back of the character. We believe that this is a reasonable constraint because it is impossible to draw a trajectory of punching action from the front or back anyway, and it is natural for the user to change viewpoint so that the user can draw a trajectory properly. Figure 4 shows the trajectories of action models. The red trajectories represent executable actions while the pink trajectories represent non-executable actions from the current viewpoint.

Conditions on sample postures

We also use style-based IK as a fundamental technique for action control. During action control, the generated motion must follow the particular form of action. For example, when the user drags the character's right hand forward quickly during a punching action, the character should not suddenly move the hand to the specified position but should perform the punching motion until the right hand reaches the specified position. Moreover, action is not simply executed forward; it can also be stopped or executed backward depending on the user's control.

However, a stopping action should not be allowed when it results in a physically impossible motion. For example, a real human cannot stop still during a kick or jump. Even when the user stops controlling (i.e., all fingers are removed from the screen), a proper motion should be generated. The action should be either cancelled by being executed backward or finished by being executed forward automatically.

To realize such control, each sample posture has two independent conditions:

- a stillness condition that indicates whether the character is allowed to become still when the touch input stops moving but stays on the screen, and
- an automatic execution direction that indicates whether motion is to be executed forward or backward when the touch input stops and leaves from the screen.

To set these conditions, we specify the time segments of conditions on each motion as shown in Figure 5. For all motions associated with an action, the same types and order of segments are specified. Each sample posture is expressed in generalized time (0 to 1). In Figure 3(b), the color of the dot represents the generalized time of the sample posture (blue to red). Although it may be possible to determine these conditions automatically, we currently chose to set this information manually, since it does not take much time.

The learned model for each action is constructed in the same way as the model for posing control. Postures derived from motion data for the action model are used. In our implementation, we use a few (one to four) motions for each action model. An example of latent space for an action (kicking) is shown in Figure 3(b), where series of postures (dots) are connected and the color of the dots represents the generalized time (red to blue).

Posture synthesis during action control

During action control, the continuity of generated postures is considered. At first, in the same way as for posing control, the latent space coordinates are determined from given inputs (constraints on body parts). The constraints can include those on the primary body part and those on other body parts. Using a multi-touch device, positions or trajectories of several body parts can be specified. At the same time, the generalized time is computed from the sample postures near the coordinates and weights. If the generalized time for the synthesized posture is close to the current time, the synthesized posture is used as output. Otherwise, the current time is forwarded or rewound toward the synthesized time in a certain time interval. The interval is determined according to the execution frame rate (normally about 1/30 seconds). The same weights for sample postures taken from sample motions are used to generate a synthesized posture with the updated time.

If the state at the current generalized time does not allow stillness, the time keeps being forwarded or rewound, even if the user's finger is not moving on the screen. If the user's finger is lifted from the screen, the time is forwarded or rewound depending on the state at the current time. During such automatic forwarding or rewinding, the last weights for sample motions are kept and used for posture synthesis.

When the action control reaches the end of motion, the system switches back to the posing control. The discontinuity from switching control is fixed using tracking control.

6. TRACKING CONTROL

Tracking control is used as a post-process after posing or action control to generate continuous and physically plausible motion. We employ acceleration-based tracking to generate continuous motion and physics-based tracking to add a physical effect to the generated motion.

Physics-based tracking control has been widely used in previous research [ACS*07][ZMC*05][HP97]. Proportional derivative (PD) control is a simple and popular method that calculates an output torque for each joint according to the current and target states of the character. A physics simulation then updates the character's state according to the calculated torques. However, this approach has several problems. For PD controllers to work, appropriate gain parameters must be given for each joint. Appropriate parameters vary depending on motion and posture. Even though there are several methods for determining gain parameters semi-automatically [ACS*07][ZMC*05][HP97], it remains difficult to realize stable control, especially for highly dynamic motions such as jumping and kicking. Another problem is that the motions generated with this approach lag the target motion and are too slow and smooth. It is difficult to realize a target motion correctly while keeping the details of the target motion. To solve these problems, we only extract physical effects such as non-controlled body parts being shaken by controlled body parts from the result of physics-based tracking control. We use an acceleration-based method [Osh06] for more accurate and faster tracking control. By combining the two tracking controls, continuous and physically plausible motions are generated.

Acceleration-based tracking

Acceleration-based tracking uses PD controllers to calculate rotational acceleration according to the character's current state and the target posture calculated from posing or action control. The angular acceleration for each joint is calculated as

$$\ddot{\mathbf{q}}_i = k\left(\mathbf{q}_{\text{target},\,i} - \mathbf{q}_i\right) - d\,\dot{\mathbf{q}}_i\,, \qquad (2)$$

where $\ddot{\mathbf{q}}_i, \dot{\mathbf{q}}_i, \mathbf{q}_i$ are the joint rotational acceleration, velocity and rotation, respectively, $\mathbf{q}_{\text{target},\,i}$ is the target joint rotation, and k and d are gain and damping parameters. We use a quaternion to represent joint rotations. The minus symbol denotes the calculated difference between two rotations. The scaling operation scales the rotation of quaternion. Since we use rotational accelerations instead of joint torques, we can use the same k and d for all joints.

In addition to joint rotations, the spatial acceleration of the root segment (pelvis) is calculated:

$$\ddot{\mathbf{p}}_{root} = k\left(\mathbf{p}_{\text{target},\,root} - \mathbf{p}_{root}\right) - d\dot{\mathbf{p}}_{root}\,, \qquad (3)$$

where $\ddot{\mathbf{p}}_{root}, \dot{\mathbf{p}}_{root}, \mathbf{p}_{root}$ are the pelvis acceleration, velocity and position, respectively.

From the calculated acceleration, the character's joint rotations and rotational velocity are updated:

$$\dot{\mathbf{q}}_i = \Delta t\ \ddot{\mathbf{q}}_i + \dot{\mathbf{q}}_i{}',\ \mathbf{q}_i = \Delta t\ \dot{\mathbf{q}}_i + \mathbf{q}_i{}', \qquad (4)$$

where $\dot{\mathbf{q}}_i{}', \mathbf{q}_i{}'$ are the values for the previous frame.

Combining physics-based tracking

In parallel with acceleration-based tracking control, physics-based tracking control is carried out. Physics-based tracking uses PD controllers and physics simulation at first. In our implementation, we use Open Dynamics Engine (ODE) for physics simulation. For tracking control, we apply forces to each body segment according to the difference between current and target positions:

$$\mathbf{f}_i = k\left(\mathbf{p}_{\text{target},i} - \mathbf{p}_i\right) - d\dot{\mathbf{p}}_i\,, \qquad (5)$$

where $\mathbf{p}_i, \dot{\mathbf{p}}_i$ are the position and velocity of the i-th segment, $\mathbf{p}_{\text{target},i}$ is the target position, and $\mathbf{f}_i$ is the force applied to the segment. The positions and velocities are updated according to the forces calculated in physics simulation. Other physical properties such as gravity and contact forces between the feet and ground are considered during physics simulation. These forces are automatically handled by the simulation engine.

As explained above, we extract physical effects from the results. Instead of using the posture generated from physics-based control, we extract velocities and apply the accumulation of the velocities with attenuation as physical effects. The physical effects are calculated as

$$\Delta\mathbf{q}_i = s_1 \Delta\mathbf{q}_i{}' + s_2\left(\mathbf{q}_{\text{sim},i} - \mathbf{q}_{\text{sim},i}{}'\right), \qquad (6)$$

where $\Delta\mathbf{q}_i$ is the relative rotation for the physical effect, $\mathbf{q}_{\text{sim},i}$ is the joint rotation calculated in the physics simulation, $\Delta\mathbf{q}_i{}', \mathbf{q}_{\text{sim},i}{}'$ are the values for the previous frame, and s_1, s_2 are attenuation parameters (we use $s_1 = 0.2\Delta t$, $s_2 = 0.5\Delta t$). Physical effects such as vibration caused by the movement of other joints are calculated using equation (6).

We apply this physical effect to the body parts that are not controlled by the user and are not in contact with the ground. We divide the character's body into five segments: two legs, two arms and the torso. For each segment, the above condition is evaluated to determine whether the physical effects are applied as:

$$\mathbf{q}_{\text{output},i} = \mathbf{q}_i + \Delta\mathbf{q}_i\,, \qquad (7)$$

where $\mathbf{q}_{\text{output},i}$ is the output joint rotation, $\mathbf{q}_i$ is the joint rotation calculated by the acceleration-based tracking control (equation (4)), and $\Delta\mathbf{q}_i$ is the relative joint rotation calculated by the physics-based tracking control (equation (6)). The same process is applied to the pelvis position and rotation. Afterward, for the legs in contact with the ground, we apply IK to fix the foot positions according to the changed pelvis position and orientation.

7. RESULTS AND DISCUSSION

We implemented the proposed method using a Windows 7 Touch API on a multi-touch-enabled PC. We are also developing an iOS (iPad/iPhone/iPod touch) version. Currently, our system has about 10 action models such as models for punching, kicking, bowing, waving a hand, jumping, and walking. The accompanying video shows various motion controls using our interface.

In a preliminarily user test, we asked a professional animator to trial our system to determine if our system can be used for animation. He understood the concept of our system and was able to use the system immediately, including the requirement of changing the viewpoint depending on the action. He noted the limitations mentioned below and that the generated posture was sometimes unnatural. This is basically due to a lack of postures that we used for learning models, and the system can be improved by adding more motion data. The professional animator agreed that a multi-touch interface is useful and noted especially that he was able to fix body parts by continuing to touch them. He commented that it might be difficult to create final motions with our system in an interactive way because the creation requires editing that is more precise, but our system is a promising tool with which to generate initial motions to be edited later with other animation systems.

In the trial, we found out that a single touch is almost enough to control various actions. In general, as several body parts must be moved in a coordinated way to realize an action, multi-touch control is considered to be necessary. However, because we introduced novel methods to realize actions such as techniques for the switching of actions and action control, various actions can be easily realized by moving a single body part. Nevertheless, multi-touch control is useful when the user wishes to add changes to postures during action by moving secondary body parts. It is also necessary to change posture specifically during posing control by imposing constraints on multiple body parts. The user sometimes wishes to fix some body parts during posing or action control, and our multi-touch interface can be used not only to move body parts but also to fix them. The beauty of our method is that it integrates single-touch and multi-touch interfaces. Many types of motion generation are possible with a single touch while detailed control is possible with multiple touching. Users can thus take advantage of both types of touch.

Similarly to other pen-based or touch-screen-based interfaces [IMH05][MCC09][Osh05][PSS02][TBvdP 04], our interface has several problems relating to the input device. First, when touching the screen, the screen is occluded by the user's hands. Another problem is that the user cannot give a command for the next action while the character is still executing the previous action, since it is difficult to touch and drag a moving body part. These are fundamental problems that are difficult to solve. However, the problems can be reduced by introducing additional interfaces and methods. For example, the use of multiple screens can solve the occlusion problem. By displaying a virtual future posture on the screen and allowing input to the posture, the problem of input delay can be solved.

A limitation of our method is that users can execute only predefined actions. By adding more motion data for new actions, our system can handle other types of action. However, it is difficult to combine two actions such as waving a hand while walking. To realize this type of combination, an additional action model for each combination must be given in advance. Blending of multiple action models during action control is future work. Another issue is the control of walking and running. As shown in the accompanying video, a cycle of walking can be executed in our current framework. However, to realize accurate control of continuous walking directions and paths, an additional method is required. There are animation systems that generate walking motions according to a given trajectory [PSS02], and they can be integrated with our system.

Since there is no alternative system that allows various controls as our interface does, it is difficult to make comparisons with other methods. However, conducting a user test to determine whether novice and professional users are able to use our interface easily is future work. Application of our methods to software that uses full-body motion control of a character and testing the effectiveness of our method are also future work.

8. CONCLUSION

We proposed a multi-touch interface for character motion control. Unlike conventional interfaces with which the user can only select an action from pre-created actions, our interface allows free posture control and the performing of various predefined actions with the user's own styles. Our methods can be used in interactive applications such as computer games and online communication. Using our interface, users will be able to express themselves by moving in their own style. In addition, our methods can be used for animation. Recently, multi-touch devices such as smart phones, tablet computers, and LCD monitors with multi-touch sensors have become popular. We believe that our methods will be a powerful interface for applications on such multi-touch devices.

ACKNOWLEDGMENTS

This work was supported in part by Research Grant from Hayao Nakayama Foundation as well as Grants-in-Aid for Scientific Research (24500238) from the Japan Society for the Promotion of Science (JSPS).

REFERENCES

[ACS*07] Brian Allen, Derek Chu, Ari Shapiro and Petros Faloutsos. On the Beat! Timing and Tension for Dynamic Characters. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2007, pp.239-247, 2007.

[BNT08] Ronan Billon, Alexis Nedelec, Jacques Tisseau. Gesture Recognition In Flow based on PCA Analysis using Multiagent System. ACM SIGCHI International Conference on Advances in Computer Entertainment Technology 2008 (ACE 2008), pp. 139-146, 2008.

[DYP03] Mira Dontcheva, Gray Yngve, Zoran Popovic: Layered Acting for Character Animation. ACM Transactions of Graphics

(SIGGRAPH 2003), Vol. 22, Issue 3, pp. 409-416, 2003.

[GMH*04] Keith Grochow, Steven L. Martin, Aaron Hertzmann, Zoran Popović. Style-based Inverse Kine-matics. ACM Transactions on Graphics, Vol. 23, Issue 3, pp. 522-531, 2004.

[HP97] Jessica K. Hodgins, and Nancy S. Pollard. Adapting Simulated Behaviors For New Characters. SIGGRAPH 1997, pp.153-162, 1997.

[IMH05] Takeo Igarashi, Tomer Moscovich, John F. Hughes. Spatial Keyframing for Performance-driven Animation. ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2005, pp. 253-258, 2005.

[IWZ*09] Satoru Ishigaki, Timothy White, Victor Zordan, C. Karen Liu. Performance-Based Control Interface for Character Animation. ACM Transactions of Graphics (SIGGRAPH 2009), 28(3), Article No. 61, 2009.

[KGP02] Kucas Kovar, Michael Gleicher, Fedderic Pighin. Motion Graphs. ACM Transactions on Graphics (SIGGRAPH 2002), Vol. 21, Issue 3, pp. 473-482, 2002

[KN10] Michael Kipp, Quan Nguyen. Multitouch Puppetry: Creating coordinated 3D motion for an articulated arm. ACM International Conference on Interactive Tabletops and Surfaces 2010, pp. 147-156, 2010.

[KHS*08] Markus Krause, Marc Herrlich, Lasse Schwarten, Jens Teichert, Benjamin Walther-Franks. Multitouch Motion Capturing. ACM International Conference on Interactive Tabletops and Surfaces 2008, 2 pages, 2008.

[LNS05] Joe Laszlo, Michael Neff, Karan Singh. Predictive Feedback for Interactive Control of Physics-based Characters. Computer Graphics Forum (EUROGRAPHICS 2005), Vol. 24, No. 3, pp. 257-265, 2005.

[LLZ*09] Xiubo Liang, Qilei Li, Xiang Zhang, Shun Zhang, Weidong Geng, Performance-Driven Motion Choreographing with Accelerometers, Computer Animation and Virtual Worlds (CASA 2009), Volume 20, Issue 2-3, pp. 89-99, 2009.

[MCC09] Jianyuan Min, Yen-Lin Chen, Jinxiang Chai. Interactive Generation of Human Animation with Deformable Motion Models. ACM Transactions on Graphics, Vol. 29, Issue 1, Article No. 9, 2009.

[Osh05] Masaki Oshita. Motion Control with Strokes. Computer Animation and Virtual Worlds, Vol. 16, Issues 3-4, pp. 237-244, 2005.

[Osh06] Masaki Oshita. Motion-Capture-Based Avatar Control Framework in Third-Person View Virtual Environments. ACM SIGCHI International Conference on Advances in Computer Entertainment Technology 2006 (ACE 2006), 9 pages, 2006.

[PSS02] Sang Il Park, Hyun Joon Shin, Sung Yong Shin. On-line locomotion generation based on motion blending. ACM SIGGRAPH Symposium on Computer Animation 2002, pp. 105-111, 2002.

[SL06] Hyun Joon Shin, Jehee Lee. Motion Synthesis and Editing in Low-Dimensional Spaces. Computer Animation and Virtual Worlds (CASA 2006), Volume 17, Issue 3-4, pp. 219-227, 2006.

[TBvdP04] M. Thorne, D. Burke. M. van De Panne. Motion Doodles: An Interface for Sketching Character Motion. ACM Transactions of Graphics (SIG-GRAPH 2004), Vol. 23, Issue 3, pp. 424-431, 2004.

[Wel93] Chris Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. M.Sc Thesis, Simon Fraser University, 1993.

[ZMC*05] Victor B. Zordan, Anna Majkowska, Bill Chiu, Matthew Fast. Dynamic Response for Motion Capture Animation. ACM Transactions of Graphics (SIGGRAPH 2005), Vol. 24, Issue 3, pp. 697-701, 2005.

Local Projections Method and Curvature Approximation of 3D Polygonal Models

Rostislav Hulík
Faculty of Information Technology, Brno University of Technology
Božetěchova 1/2
612 66, Brno, Czech Republic
ihulik@fit.vutbr.cz

Přemysl Kršek
Faculty of Information Technology, Brno University of Technology
Božetěchova 1/2
612 66, Brno, Czech Republic
krsek@fit.vutbr.cz

ABSTRACT

Mesh processing is a wide area which consists of several approaches, heavily specific for each task. We present a novel approach to mesh processing using the Local Projections method. The described method makes benefit of wide variety of image processing algorithms, very similar to 3D tasks, and implements a conversion mechanism to make possible use of these processes on polygonal models. We also designed, implemented and evaluated the curvature approximation algorithm to test and compare the proposed method usability on real and artificial data. Use of the proposed method brings significant benefits especially to noised mesh analysis.

Keywords
Local Projections, Curvature, Polygonal model, Mesh Processing, Image Processing

1. INTRODUCTION

Polygonal model processing techniques consist of a wide variety of approaches specific for each different task. Each of these methods is especially designed for its purpose, e.g. curvature approximation, mesh smoothing, simplification, matching or feature extraction. Such diversity leads to the separation of this field of study into several different domains.

Even the problems stated above are handled separately, they need to overcome the same problems emerging from the polygonal mesh data structure itself. This includes irregularities of polygons, unconnected parts of the mesh or the polygonal surface approximation itself.

In the computer graphics area, another widely used and well documented area exists with very similar operations. The image processing scope consists of well-known techniques for curvature approximation on raster images, smoothing kernels or feature extraction methods. The operations are very similar to these used in polygonal mesh processing.

The main problem appears from different data structures. We present a consistent and robust way of polygonal model representation, where each vertex is described by its own tangent raster depicting its neighborhood. In this way, we can apply arbitrary raster operators right onto the mesh and extract the necessary information. We are able to approximate the curvature, extract specific features or apply other operators, such as smoothing, right on the tangent rasters resulting in smoothing for successive operations. All of these operations can be done only on tangent rasters without alternating the original mesh topology.

As a usability demonstration, we have chosen implementation of curvature approximation, which can be easily evaluated by comparing not only with existing methods, but also with an analytical approach, which allows us to demonstrate the accuracy of our method. In the evaluation chapter, we demonstrate that this method brings significant advantages to the smoothing of largely noised meshes.

In the first part of this paper, we describe deeply the Local Projections (LP hereafter) method to understand fully the proposed conversion. The second part is concerning with curvature approximation methods using the proposed LP method. In the concluding part, we present test results gathering the efficiency of LP and the accuracy of curvature approximation using this approach by comparison with today's commonly used methods and analytically computed curvature. Comparison is done on both smoothed and noisy data sets. Last section is devoted to conclusions, future goals and possible improvements of LP.

2. RELATED WORK

As we stated above, the mesh processing field of study is a wide area consisting of many different approaches. In this section, we describe today's methods for curvature estimation on mesh structures in order to compare them with our LP curvature approximation.

The most widely known algorithms for mesh curvature estimation are the discrete differential operators presented by Meyer et al. [Mey00a]. The authors describe operators for direct approximation of curvature from mesh structures considering angles of adjacent edges. The operators exist for both Gauss and mean curvature and it is possible to induce minimal and maximal curvature respectively. Although the computation is normalized by the Voronoi area of neighboring faces, the method fails when any triangle irregularities or unconnected faces occurs. On the other hand, the approximation is very straightforward which significantly influences the efficiency. A similar method of computing directly from mesh structure is described by Rusinkiewicz [Rus00a].

Another approach makes benefits from geometric fitting of spheres directly onto mesh structures. The algebraic point set surfaces method (APSS) was presented by Guennebaud and Gross [Gue00a] [Gue00b] as an algorithm for point cloud curvature estimation, but it was also adapted to point clouds with normals and polygon meshes. Due to the approximation of surface with spheres, triangular irregularity problem is overcome. The algorithm also partially solves the problem of unconnected polygons (we fit spheres with certain radii – the time complexity increases) and the mesh holds its curvature features even on noisy data.

Simari et al. [Sim00a] presented a method for robust curvature estimation, which is based on regularly resampled polylines with intersections in current vertices and specified angular steps. From these resampled polylines (forming spider shape around current vertex), authors simply compute curvatures in specified directions given by each branch and imply principal curvature values and directions. It is clear that the authors also try to overcome problems of polygon mesh irregularities using resampling procedures. Unconnected polygons are not dealt with in the work presented.

Page et al. [Pag00a] proposed another method aimed especially at noisy datasets. Their approach tries to find a geodesic neighborhood of a vertex with specified distance, which gives us the possibility of smoothing high frequency noise. It is necessary to underline the fact that the authors do not use an Euclidean distance to estimate geodesic neighborhoods, but the shortest geodesic path. Selected vertices then vote to Taubin's curvature tensor [Tau00a] from which curvature is estimated. This method has proven to be very robust against the noise of meshes and due to possible resampling of vertices on geodesic neighborhood, robust against irregular triangulation.

There exist several other approaches to curvature estimation, described e.g. in [Ozt00a][Mok00a]. The main traits of novel methods are always the same – to overcome mesh tessellation irregularities, unconnected elements, noise and polygonal approximation. It is also clear from these examples of curvature processing that each method is developed especially for its purpose, as stated in the introduction. We tried to approach this problem from the other, well documented field – image processing.

3. LOCAL PROJECTIONS METHOD

In this section, a method for the 3D – 2D problem conversion is presented. The main purpose is to create a novel approach to mesh processing by turning the problem of polygonal mesh analysis to an raster image problem.

For vertex neighborhood representation in 2D image rasters, we have chosen to project distances from the tangent plane at a given vertex position. This operation results in a rasterized matrix of z-distances (depth image) attached to each vertex. However, it is possible to project arbitrary vertex information, e.g. color, curvature or other vertex specific values present in the mesh structure. In the following lines, we consider always depth image representation.

Rasterization must overcome all of the problems stated above, such as unconnected polygons, holes in meshes etc. All of this must be done while conserving minimal time complexity to maintain usability of the whole procedure.

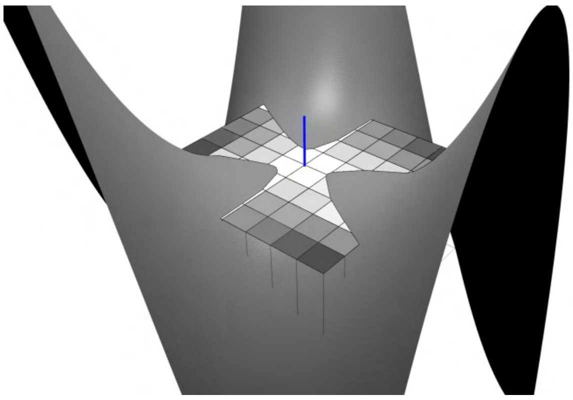


Figure 1. Tangent raster with projected z-distance

In Figure 1, an example of such a projection is presented. Tangent rasters are defined **for each vertex** and describe the vertex neighborhood.

Regarding this fact, size of matrices must be chosen appropriately to the specific task. For example when using LP method for curvature approximation, we choose smaller matrix size due to the definition of curvature – we expect the analysis of small neighborhood around pixel. On the other hand, when using the LP method for i.e. feature extraction on mesh structures points of interest, we can describe vertices with matrices of larger size and resolution.

Tangent plane direction itself is computed for each vertex using its normal vector defined as:

$$\overrightarrow{n(x)} = \frac{\sum_{i \in F(x)} \overrightarrow{n(i)}}{|F(x)|} \qquad (1)$$

where x is current vertex and $F(x)$ is a set of neighboring faces. $F(x)$ consists of not essentially adjacent faces to the current vertex – if larger smoothing is required, F can contain larger neighborhood.

In the following sections, we mention two parameters of matrices. The **resolution** stands for matrix size in pixels (matrix is square) and **size** is the relative size of the tangent plane to the median of model edge lengths. E.g., if the median of edge lengths is equal to **x** in model coordinates and the matrix size is set to **s**, the real edge size of the matrix will be equal to **xs.**

Matrix resolution and size is highly specific to the task we request from LP method. If LP is used for e.g. edge detection or curvature approximation, it is necessary to apply relatively small resolution (Sobel operators in image processing hardly exceed 9x9 resolution) and small size of matrix (the larger matrix is, the larger smoothing occurs). On the other hand, SIFT descriptors need large pixel neighborhood to compare, so size and resolution can be much higher.

In subsequent computations, each vertex matrix is treated individually – we can apply an arbitrary image processing operator to every vertex such as curvature computation, edge/blob detector, feature extractor etc. The value of raster's center pixel is the value of the used operator for corresponding vertex.

Rasterization algorithm analysis

To achieve the exact projection on tangent plane multiple algorithms can be used. We consider three possibilities:

1. **Blind rasterization of the whole mesh on each raster**

   This method rasterizes precisely even meshes with unconnected polygons and holes. It has, however, a high time consumption. On the other side, it is well parallelizable and we can imagine this solution in possible future hardware implementation.

2. **Rasterization of vertex neighborhood**

   By rasterizing only a vertex neighborhood, we can achieve a very fast projection procedure. This approach works only for well-connected meshes and can pose problems when the holes occur in the proximity of the current vertex.

3. **Ray-casting rasterization**

   Ray-casting of each matrix cell also accomplishes a total projection without problems. It has, however, the same problems as possibility 1 – an extreme time complexity.

In order to fulfill all quality requirements of projection while maintaining time complexity as low as possible, a hybrid algorithm is proposed combining all three steps described above.

In Figure 2, the final proposed algorithm overview is presented.

```
For each vertex:
   I.   Get all neighboring vertices
  II.   Rasterize triangles connected
        to them
 III.   Rasterize borders
  IV.   Ray-cast untouched pixels
```

Figure 2. Rasterization algorithm

In **step I**, all neighboring vertices are found, which will be projected onto tangent raster. The maximum distance must be specified to prevent passing the whole mesh in the case when faces are perpendicular to the projection plane (see Figure 3). This step is followed by **step II** which is very fast and poses minimal computing and efficiency problems.

Step III blindly rasterizes the borders of already found segments. This apparently useless step has the purpose of finding triangles which have all vertices outside projection plane (Figure 3 – yellow segments). Steps I-III can clearly be replaced by traversing triangles instead of vertices but this approach will increase extensively the time complexity – each triangle needs to be tested for possibility to project it onto the tangent plane - in our approach, we test only point projection.

The blind border rasterization cannot, however, find all possible interfering polygons (e.g. border search rasterizes only one-neighborhood). In **step IV**, we ray-cast the rest of provisionally non- rasterized pixel to assure the certainty of all pixels rasterization. The ray-casting can be further accelerated by using some high-level search structure such as 3D R-Tree [Gut00a] etc.

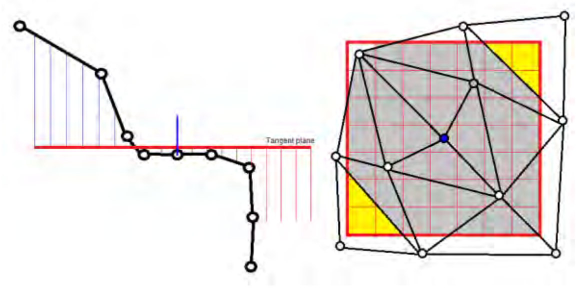


Figure 3. Problems in rasterization, infinity projection (left) and triangles having all vertices outside projection matrix (right)

All of the four rasterization steps were chosen for maximum speed-up of the whole procedure. In the evaluation section, each of the steps is evaluated and diagnosed to provide the reason for necessity of each step.

4. CURVATURE APPROXIMATION

The rasters computed by our algorithm can be used for arbitrary feature extraction of the polygon mesh. We have chosen curvature extraction for the reason that it can be easily compared with the analytical approach, i.e. we can prove the feature conservation by conversion from a polygonal mesh to an image problem.

The main goal of curvature extraction is to approximate the curvature behavior of an original model. It is necessary to accentuate that the real curvature on the polygon mesh cannot be taken into account because of null curvature on mesh faces and infinite curvature on edges.

To approximate the curvature and its directions from depth map (i.e. our tangent rasters), we can apply the Hessian matrix analysis to approximate the curvature behavior. For continuous functions, the Hessian H is the Jacobian matrix of derivatives $\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}$ of a function $f(x_1, x_2, \dots, x_n)$ with respect to x_1, x_2, …, x_n:

$$Hf(x_1, x_2, \dots, x_n) = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{vmatrix} \quad (2)$$

The Hessian H is a simple matrix of partial second order derivatives of a continuous function. In our case of a discrete raster, we must define a second derivative operator to approximate the derivative behavior on raster image. For this purpose, we have chosen second order Sobel operators to compute a derivative estimation L_{xy} by convolution with the input raster (equation 3). It is clear that the convolution is not necessary to be executed for all raster pixels, but only for the center, i.e. for the current vertex.

$$L_{xy}(i, \sigma) = \sqrt{\left(g_x(\sigma) * I(i)\right)^2 + \left(g_y(\sigma) * I(i)\right)^2} \quad (3)$$

Taking computed partial derivatives, we can easily construct a discrete equivalent of Hessian (equation 4).

$$H(i, \sigma) = \begin{bmatrix} L_{xx}(i, \sigma) & L_{xy}(i, \sigma) \\ L_{xy}(i, \sigma) & L_{yy}(i, \sigma) \end{bmatrix} \quad (4)$$

where σ is the derivation neighborhood (i.e. matrix size) and i represents a specific point in the image [Mik00a]. Corresponding curvatures and directions can be subsequently computed by eigen analysis of the computed Hessian (equation 5):

$$HX = \lambda X \quad (5)$$

Where λ is an eigenvalue and X the corresponding eigenvector. For a 2x2 Hessian matrix, we obtain precisely two eigenvalues and two eigenvectors corresponding to approximated curvatures on the tangent plane. The necessity of backward transformation to 3D coordinates is evident.

By this technique, we can obtain a robust curvature estimation technique even on noisy meshes due to the possibility of off-line smoothing right on the tangent rasters. Off-line smoothing in this case means the smoothing only on raster representation, without any mesh topology change. Such smoothing can be achieved by application of one of the image smoothing kernels, such as median, Gaussian or mean. Smoothing can be also achieved by application of some polynomial interpolation which can easily break hard mesh edges and approximate the real object surface.

Another noise cancelling feature of the LP method is the possibility of larger vertex neighborhood projection. Even this fact is against the formal curvature definition, it can be used on very noisy models when a larger neighborhood is necessary for real object feature extraction. The idea of a larger vertex neighborhood was also presented in [Gue00a] [Gue00b] [Sim00a] [Pag00a] as a means for noise cancelling. For LP, this feature produces another significant advantage – the polygon structure cancellation. By the rasterization of larger neighborhoods the image is normalized – each vertex is represented by a map of the same size.

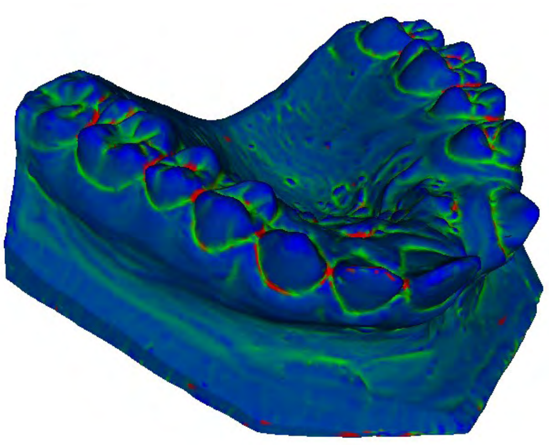


Figure 4. Maximum curvature computed from 7x7 size matrices of the size equal to the mean size of vertex one-neighborhood.

The LP curvature approximation method can also yield several drawbacks emerging from the rasterization and 3D-2D conversion.

The first produces problems when small tangent resolution is applied which can severely influence the output due to exceedingly coarse representation of vertex neighborhoods. In this case, we can observe e.g. significant deflection of curvature directions from analytically computed curvature. In the curvature computation case, it is recommended to use at least a 7x7 raster (This value was chosen experimentally as a compromise to the speed of rasterization process and accuracy of approximation). For other applications, such as SIFT or SURF descriptors, much greater resolution can be required.

Another problem can result from highly curved meshes. The projection of these meshes can result in cropping of the real surface in projection into 2D (real surface can lie "behind" already rasterized pixels – see Figure 5, yellow section). Although this fact can lead to very imprecise representation of the original mesh shape, the tests on real datasets show that this phenomenon is extremely rare due to the size of matrices (which rarely exceed one- or two-neighborhood). For curvature computation, we semi-occasionally use matrices of such large neighborhoods – in our computations, we rarely exceed one-neighborhood of corresponding vertex. On the other hand, this effect can be applied for noise canceling and detail smoothing for coarse feature extraction.

5. EVALUATION

We designed multiple tests for the proposed methods evaluation. The evaluation section is separated into two major sections – evaluation of the rasterization algorithm and evaluation of curvature approximation.

Local projections method

On preceding lines, a robust projection method of the mesh onto tangent raster was presented. As a first evaluation method, we compare the number of projected pixels at each step of rasterization.

	Average	Median	Std. dev.
After step 2	91,59%	95,08%	10,36%
After step 3	94,31%	98,25%	10,40%
After step 4	95,20%	98,87%	10,19%

Table 1. Pixels projected in each step

In Table 1, the number of projected pixels at each step is presented. We tested our approach on various polygonal meshes. In the testing dataset, there are closed meshes, data with holes and non-connected vertices and other problematic cases described above.

As tests have proven, a majority of pixels are filled in steps 1 and 2 (see the section Local Projection Method). These steps are the quickest part of the rasterization. Step 3 will rasterize up to 94,32% of pixels – it is clear that this modification plays an important role in efficiency improvement, because by simple blind rasterization we fill up a significant partition of unprocessed pixels. For ray-casting step 4, only 0,89% of pixels remain – this leads to minimal computing complexity – ray casting is used only if there is no possibility to find the polygon by neighbor search. The non 100% efficiency is due to the pixels which are projecting infinity – in the post-processing step, these are filled by a chosen maximum value – we propose this value to be equal to the size of the tangent matrix (which is the same for all vertices).

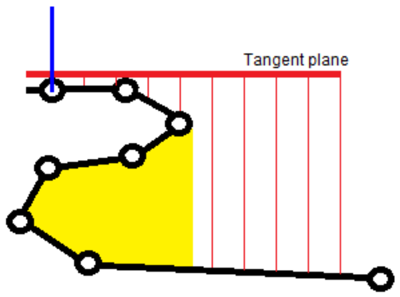


Figure 5. Problems on significantly curved mesh

	Matrix resolution 3x3, Size 2		Matrix resolution 9x9, Size 9		Curvature Meyer et al. [Mey00a]	
No of vertices	Average (ms)	Median(ms)	Average(ms)	Median(ms)	Average(ms)	Median(ms)
~20000	1478,71	1358,41	16546,2	14821,2	238,9	237,62
~30000	2088,82	1674,32	20328,55	14824,65	372,94	377,02
~50000	4433,09	2782,42	36448,18	18381,6	619,71	620,15
~200000	17714,03	12355,1	101518,96	93521,2	2538,65	2546,08

Table 2. Speed efficiency of LP method compared with Meyer et al.

By this modification, we assign a specific, non-infinite value, to each uninitialized pixel. The matrix is then able of being processed by any image processing algorithms without creating infinity artifacts. We have chosen a "cube" maximum restriction due to the expected behavior of subsequent methods – a large, possible infinite value would cause a large perturbation in succeeding operator. In this case, LPM will lose its smoothing capabilities. We need then some value which represents a maximum possible value (size of tangent matrix) but does not cause such problems. The maximum z value can be however set as a parameter of computation. The right choice will surely influence the robustness of this method with respect to the mesh imperfections.

The second test is a speed comparison between our method and the well-known curvature computation method by curvature operators [Mey00a]. We have chosen this approach to compare with our method due to similar neighborhood search and feature extraction. This operation is comparable with rasterization of 3x3 matrices of size equal to 2, because it takes into account approximately the vertex neighborhood of the same size.

In Table 2, the time test results are presented. The LP method is clearly more time expensive than simple neighborhood search, which was expected. The effectiveness is dependent not only on matrix resolution or size, but also on mesh structure – on the number of pixels searched by ray-casting (mesh complexity), number of holes and inconsistencies.

Even the software implementation is significantly slower than simple neighborhood search, the efficiency is redeemed by the wide possibilities of this method. The method is largely parallelizable, so hardware implementation is expected to be prepared in the near future.

Curvature approximation

Curvature approximation is the second type of test provided in this paper. We prepared a testing dataset consisting of both real world objects (created by 3D scanning or conversion from volumetric datasets) and artificial data. Artificial data are primarily analytical shapes, where curvature it is possible to compute analytically. Both types of data were also used with added noise.

We compared our LP method with two common curvature approximations – the differential geometry operators of Meyer et al. and APSS (see Related work). Additionally, on analytical models, the curvature was also compared with analytical computations (AC hereafter).

In Figure 6, a comparison of the $z = \cos(x^2 + y^2)$ surface is presented. To the model, some unconnected vertices were artificially added to demonstrate the method's behavior in such case. Such unconnected polygons/vertices are common problem on polygonal meshes created by e.g. point cloud reconstruction. From this figure it is clear, that both Meyer's and the APSS methods are more precise in curvature approximation on well triangulated surface (visible peaks are well aligned with analytically computed curvature). The LP method holds the progression of AC.

A different testing schema occurs when artificial noise is added to the mesh (see Figure 7). The noise was added by the arbitrary fractal displacement method. In this case, the main advantages of the LP method are visible. Even the APSS holds major extrema due to the variable fitting radii, perturbations are still present on low curvature regions. The LP method holds all major progressions of the analytical computation. This fact makes this method very suitable for noised and incomplete meshes, where the curvature can be approximated very robustly.

6. CONCLUSION

A novel approach for mesh analysis has been presented along with one validation algorithm – a curvature approximation. The main contribution of this method is that it brings possible conversion of 3D polygonal mesh problem to the raster image processing issue. By this modification, we are able to apply multiple operators well known from image processing area right onto the mesh structure and treat it like simple raster image.

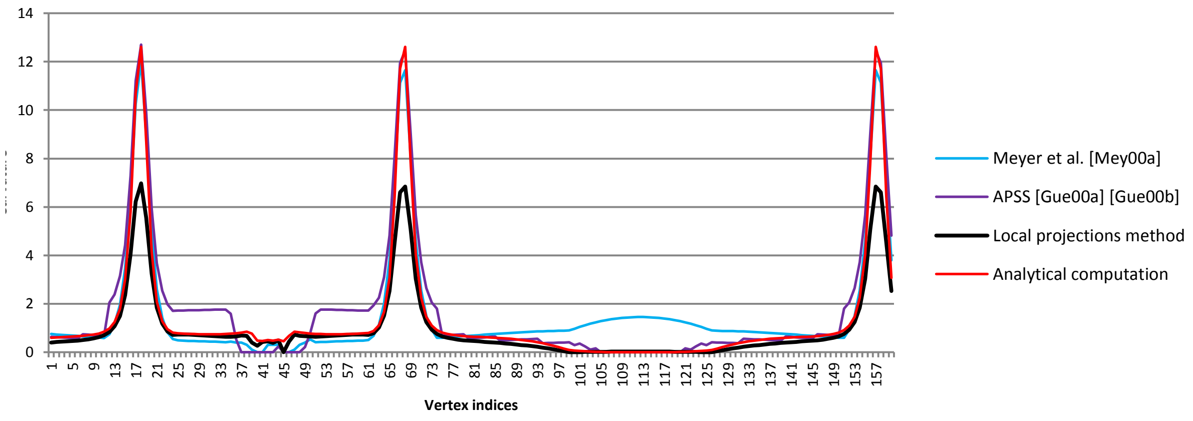


Figure 6. Surface $z = \cos(x^2 + y^2)$ – maximum curvature

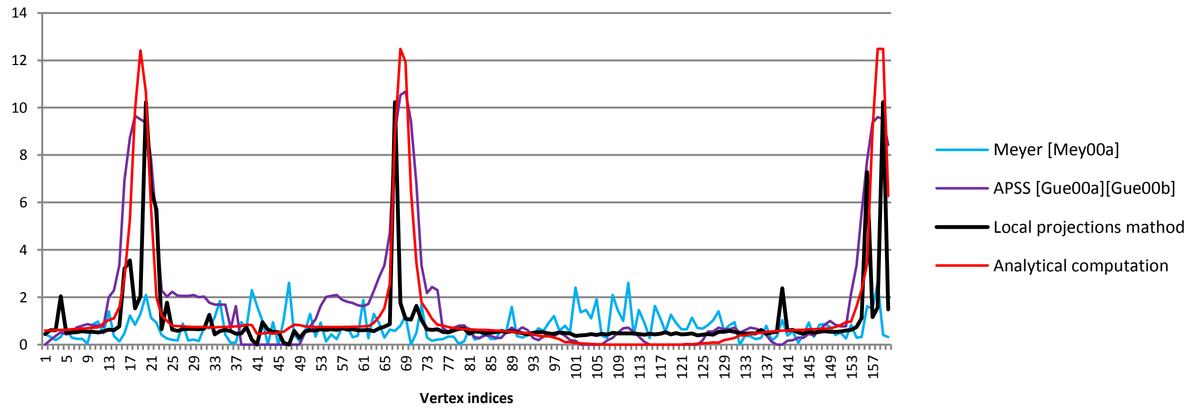


Figure 7. Surface $z = \cos(x^2 + y^2)$ – maximum curvature, added random noise to triangulated model

As tests have shown, the LP method has a big disadvantage in efficiency because of multiple projections computed for each vertex of the mesh. These problems were partly solved by application of a robust rasterization algorithm which speeded up the projection process. Additionally, due to the proposed hybrid method, we are able to analyze an arbitrary mesh – even meshes with topological inconsistencies, holes, unconnected elements, noise etc. Due to the method's simplicity, a hardware implementation is possible which is supposed to significantly speed up the computation of rasters.

The main advantage of the LP method is that it allows us to apply an arbitrary image processing operator right on the 3D mesh structure, which is in our case represented as a set of vertex tangent rasters. It is then possible to apply e.g. curvature approximations or feature extractors from the well-known image processing area. In preceding articles, multiple drawbacks associated with 3D to 2D projection were discussed. In tests on real and artificial data, these projection problems are irrelevant and they influence the subsequent computation minimally.

We also presented a new curvature approximation method, which comes from the LP method and image processing. The curvature is computed using Hessian matrix analysis from arbitrary sized tangent matrices, which allows us (in combination with smoothing operators) to estimate curvatures on any mesh even with structural noise present. The smoothing and noise cancelling capabilities of method presented makes the LP method a perfect candidate in the field of feature extraction from damaged and noised meshes (Figure 8).

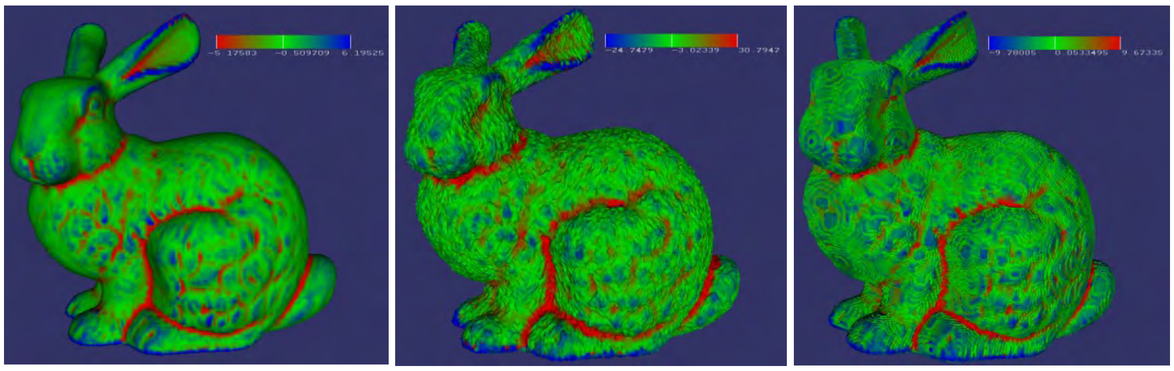


Figure 8. Mean curvature comparison on smoothed mesh (left), mesh with added random noise (middle) and marching cubes created model (right)

Future work in this field of study can be aimed at already mentioned hardware implementation of the LP method, which can significantly increase the computation speed of all methods and make our method comparable in efficiency with other largely used vertex-neighbor searching algorithms. Another research tendency will aim at other image processing operators possibly usable with LP method, especially in the field of mesh matching and feature extraction.

7. ACKNOWLEDGMENTS

This work was supported by the European Community's 7th Framework Artemis JU grant agreement n. 100233 (R3-COP) and European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.00 70).

8. REFERENCES

[Mey00a] Meyer, M., Desbrun, M., Schröder, P., Barr, A.H. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds: Visualization and Mathematics III (p. 35-57), Springer-Verlag, Heidelberg, 2003

[Gue00a] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces: ACM Trans. Graph. 26, 3, Article 23, July 2007

[Gue00b] G. Guennebaud, M. Germann, and M.H. Gross, Dynamic Sampling and Rendering of Algebraic Point Set Surfaces. In Proceedings of Comput. Graph. Forum. 2008

[Gut00a] Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching: Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD '84. pp. 47, 1984

[Mik00a] Mikolajczyk, K., Schmid, C.: An Affine Invariant Interest Point Detector: In Proceedings of the 7th European Conference on Computer Vision-Part I (ECCV '02), Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen (Eds.). Springer-Verlag, London, UK, UK, p. 128-142, 2002

[Mok00a] Mokhtarian, F., Khalili, N., Yuen, P.: Curvature Computation on Free-Form 3-D Meshes at Multiple Scales, Computer Vision and Image Understanding, Volume 83, Issue 2, August 2001

[Ozt00a] Oztireli, C., Guennebaud, G. and Gross, M.: Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression: In proceedings of Computer Graphics Forum 2, vol. 28, 2009

[Pag00a] D. L. Page, Y. Sun, A. F. Koschan, J. Paik, and M. A. Abidi. 2002. Normal vector voting: crease detection and curvature estimation on large, noisy meshes. Graph. Models 64, 3-4 (May 2002)

[Rus00a] Rusinkiewicz S. Estimating Curvatures and Their Derivatives on Triangle Meshes: In Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium (3DPVT '04). IEEE Computer Society, Washington, DC, USA, 486-493, 2004.

[Sim00a] Simari, P., Singh, K., Pedersen, H.: Spider: A robust curvature estimator for noisy, irregular meshes: Technical report CSRG-531, Dynamic Graphics Project, Dept. of Comp.Sci., Univ. Toronto, 2005

[Tau00a] G. Taubin, Estimating the tensor of curvature of a surface from a polyhedral approximation, in Proceedings of the Fifth International Conference on Computer Vision, 1995, pp. 902–907

Realtime global illumination using compressed pre-computed indirect illumination textures

Chris Bahnsen Antoine Dewilde Casper Pedersen Gabrielle Tranchet Claus B. Madsen[*]

Department of Architecture, Design and Media Technology
Aalborg University
Niels Jernes Vej 14
9220 Aalborg Øst, Denmark

ABSTRACT

In this paper, we present a way to render images in real time, with both direct and indirect illumination. Our approach uses precomputed indirect illumination images, produced at certain intervals, which need not be constant. When rendering a scene, the two closest images are then interpolated and added to the direct illumination to produce the total illumination. Depending on the type of image produced, the algorithm allows a camera to move, and even objects to be added or modified at runtime to some extent. Finally, we will see that the amount of data to store and process can also be reduced using a dimensionality reduction algorithm such as PCA.

Keywords

Global illumination, Realtime Rendering, Principal Component Analysis, Phong Shading, Compression

1 INTRODUCTION

Rendering global illumination in real time is a challenging problem of today's computer graphics. Although several techniques exist to compute a full global illumination model, and hence produce a realistic scene, these techniques are usually extremely expensive in terms of computation time, and so are not usable in interactive applications, even though effort has been made to make these renders as fast as possible [WKB+02]. On the other hand, algorithms than can achieve interactive sampling intervals usually do not take into account the whole illumination model for a dynamic scene and, in the best cases, only compute a few bounces for light rays [NPG03].

Current state-of-the-art techniques to render global illumination include, but are not limited to, ray tracing, photon mapping, and usual algorithms used for interactive applications such as the use of Phong shading with an ambient term to simulate indirect light. Most algorithms that aim at rendering indirect lighting efficiently, then use a combination of these techniques, together with other machine learning elements such as interpolation, clustering or neural networks. A method using clusters to render efficiently global illumination has been studied by Christensen et al.[Chr99]. For instance, interpolating images characterizing a luminance distribution can be done in a non-linear way, using the data to train a neural network [DRP09]. In a similar way, Christensen et al.[Chr99] has written about how to make a faster photon map for rendering global illumination, using a technique similar to clustering as it will group together during one stage of the algorithm photons having similar irradiance.

In this paper, our goal is to be able to render a scene with daylight using precomputed indirect illumination images. The technique we will explain allows for the light source to change dynamically on a set path, and the camera to move freely within boundaries, as well as dynamic objects to be added and moved around the scene, if we accept minor, probably unnoticeable inaccuracies in the indirect illumination. Although this paper will focus on daylight, considering the sun as the only light source, this technique is actually applicable to any scene where all the possible lighting conditions are known beforehand.

The idea our algorithm is based on is quite simple: we consider the fact that the illumination of a scene is actually given by both the direct illumination, and the indirect illumination. Since the direct illumination changes frequently when it comes to daylight, and since computationally efficient algorithms to produce it exist, we will just compute them in real time. In our test program, we used Phong shading, coupled with basic shadow mapping; any other model that gives realistic

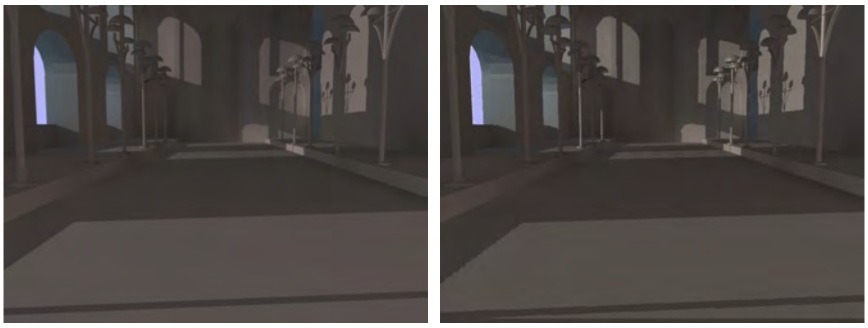

(a) 3DSMax render (b) Our render

Figure 1: Comparison between a render of total lighting done in Autodesk 3DSMax 2011 and with our technique.

results in real time would work of course. The indirect illumination, however, will be precomputed at certain time steps and stored into images used as textures. To render a certain frame, we will then just compute the direct illumination (without any ambient term), and add the indirect illumination that we get by interpolating the two frames the closest to the one we are rendering. The interpolation to approximate results is a method that has proved itself, for instance in [RGS09] where it is discussed that computing an approximating physically plausible illumination over physically correct illumination takes less computation time and gives, as the name states, very plausible results. Other methods, including [RGKS08], also include compressed pre-computed information in order to speed up rendering (in this case, precomputed depths and coherent surface shadow maps).

So, the problem can be narrowed down to finding the right sampling interval for indirect illumination, and reducing the amount of data to process as much as possible by compressing the precomputed frames. Defining a good sampling interval is important so that the preprocessing is not too long, as we try to avoid rendering similar images, but the quality of the interpolation is still acceptable. Reducing the amount of data is important both for disk space issues, and to reduce the loading time.

In the next following sections, we will explain the algorithm and the above-mentioned issues in more details. In section 2, we will give a general overview of our algorithm. In section 3 we will explain our method in further details, including the possible refinements (compressing the data using PCA, and defining a dynamic sampling interval). Finally, section 4 will be dedicated to some results analysis, and considerations for future inspection. All our example images feature an indoor scene inspired by the castle of Koldinghus in Denmark, as the first application considered for the method is a virtual tour of the castle.

*cbahns08 | adewil11 | cped08 | gtranc10@student.aau.dk, cbm@create.aau.dk

2 OVERVIEW OF THE APPROACH TAKEN

As explained before, the main idea of our algorithm is to separate the direct illumination from the indirect lighting. Since the direct illumination can be computed easily and holds high-frequency data which makes it hard to compress, we will use off-the-shelf algorithms to calculate it in real time; in our demo program, Phong shading coupled with basic shadow mapping. The indirect illumination, however, will be precomputed at certain timesteps and stored in a compressed form. When rendering a frame, we will then interpolate between the two closest frames computed, and add that indirect illumination to the direct light. Figure 1 shows the result we get, compared to a scene rendered in a standard 3D modeling program that uses ray tracing and photon mapping.

However, as we would like to compress the indirect illumination images, we would like to get rid of as much high-frequency data as possible. This is the reason why we also compute the contribution from the skylight separately. Actually, in our approach, we will consider the skylight as being a constant contribution that just has to be scaled depending on the time of the day. In our demo program, we use a precomputed image for the skylight, which we scale depending on the position of the sun. This gives acceptable results for our application but, of course, any other technique that gives good results and can run in real time can be used, such as [NJTK95] where a model of skylight is built and its illuminance is calculated.

Figure 3 shows the different elements needed. The final render of an arbitrary frame will then be the sum of:

- The direct illumination, computed in real time.
- The indirect illumination without skylight. This is obtained by interpolating between the two precomputed frames closest to the one we are rendering.
- The skylight, precomputed according to the lighting model of your choice.

Aside from compression issues, removing the direct illumination from the precomputed frames also has other advantages. For instance, it allows for dynamic objects to be included into the scene with only minor errors in indirect lighting. Depending on the type of object, these errors might not even be noticeable to the untrained, naked eye. However, compared to the scene with the teapot rendered in a 3D modelling program, the color bleeding and the blocking of indirect illumination of the teapot is absent in our render. Figure 2 shows an example of such a scene, where a teapot has been added dynamically.

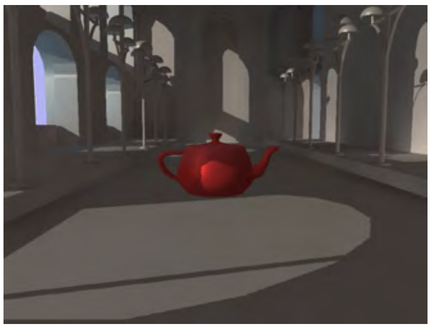


Figure 2: A scene with global illumination rendered by our technique, where a teapot has been added dynamically.

3 DETAILS ABOUT THE METHOD USED

3.1 Basic idea

Implementing our approach in an application is done in two steps: first of all, pictures for the indirect lighting must be generated, at some point before the application starts. Only after that step can we actually render anything. In the next subsection, we will focus on the issues related to the generation of the images. Let us just assume that we were able to produce images containing the indirect illumination (without skylight). We will also assume for now that these images are uncompressed and taken at regular intervals; we will deal with compression and dynamic sampling interval issues in the next section. In general, a value that produces high-quality results while still keeping the number of precomputations reasonable, is a framerate of one picture every five minutes – which, for a sequence of a full day, gives 288 images to render.

So, let us assume that we have a set of images containing information about the indirect illumination. In our tests, we used bitmap images, which gives the advantage of being encoded over 24 bits, hence containing more information than compressed file formats. To render the scene, we will use a custom set of shaders that will compute the direct illumination, and “paste" the indirect illumination and the skylight on it.

Loading images from disk takes a significant amount of time. Because of that, loading the precomputed images on-the-fly results in a significant drop in frame rate, while still achieving interactive frame rate though. So, if the amount of data to load is acceptable (which it should be once we introduce compression), and if the application requires fluid animations, it might be beneficial to consider pre-loading all the images in memory at startup, keeping pointers to the different data, and accessing them when needed.

The next step is to take advantage of the GPU’s computing capabilities to interpolate and render the scene. Let us say that we want to render a frame for an arbitrary time. We will first find the closest precomputed images (one before the current frame, one after), then we will interpolate them to get the correct luminance value. With pictures taken at a fixed frequency, the index of the image just before the current frame would be

$$N = \text{floor}\left(\frac{H \times 60 + M}{\Delta T}\right) \tag{1}$$

where H and M represent the time that is represented in the frame, ΔT is the time between two precomputed images, and $N+1$ is the image right after the current frame.

Once the two images closest to the current frame are retrieved, they are then passed as textures to a specific fragment shader. This fragment shader calculates the fragment color by interpolating the two frames and adding the result to the direct illumination and skylight contributions, as shown by Equation (2). In our test renders, we used basic linear interpolation, which gives plausible results since the indirect illumination has only low-frequency information. Other kinds of interpolation will be discussed in section 4.2. The equation used to compute the total luminance of a pixel is:

$$\begin{aligned} L(t) = L_{\text{direct}}(t) + L_{\text{sky}}(t) + (1-\alpha)\,L_N(t) \\ + \alpha L_{N+1}(t) \end{aligned} \tag{2}$$

where L_N and L_{N+1} are texels retrieved by sampling the textures calculated in Equation (1). Since we currently have a fixed interval between each precomputed frame, calculating the value for α is quite straightforward. The computation is given in Equation (3).

$$\alpha = \frac{(M \bmod \Delta T) - \text{offset}}{\Delta T} \tag{3}$$

where offset is the time (in minutes past midnight) of the first frame.

The next issue to consider is how to do the texture lookup, and consequently how to generate the illumination textures themselves. Basically, three different methods can be considered, depending on the type of application:

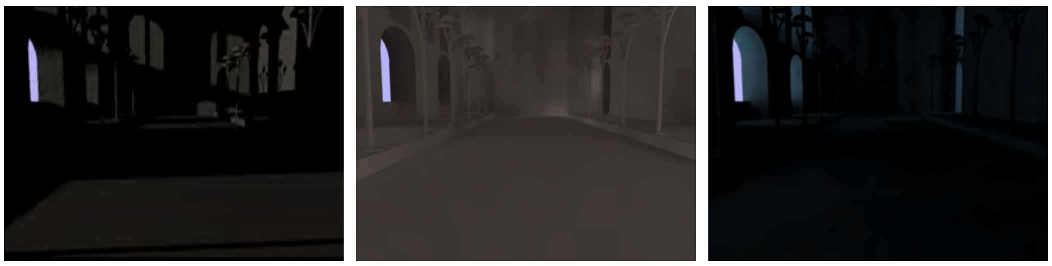


Figure 3: The different elements that make up the final result. Left: direct illumination (computed at runtime). Center: indirect illumination (precomputed at several time steps). Right: skylight (precomputed once and scaled)

- If the camera cannot move at all inside the application, simply pasting the texture onto the final render might be enough. In that case, the illumination texture is a simple 2-dimensional render as shown in Figure 4.
- If the camera can rotate but not move, using cube maps might be an option. In that case, the illumination texture is a cubemap-looking texture or set of textures (one for each direction).
- In the general case where the camera is allowed to move freely, using texture atlases might be the option. In that case, the texture is "pasted" onto the geometry, and the files are a list of textures.

All options, however, have their shortcomings. The first option makes having reflective objects in the scene difficult, as rendering the reflection cube map for such objects would be tricky. Indeed, when rendering the cube map for a particular object, the illumination image for that specific render has to be used. In other words, this technique needs 6 sets of images for each reflective object (unless indirect illumination is disabled in the cube map of course). Furthermore, the illumination cube map for that object has to be aligned with the actual render of the scene, which is not trivial.

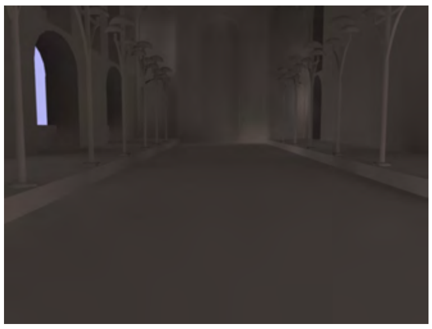


Figure 4: Illumination texture used if the camera is not allowed to move

The second technique has the same problem as the first one, plus the fact that a cube map is also used for the actual rendering of the scene. Furthermore, since a cube map takes six times as much space as a regular 2D texture, the amount of data to precompute rises by almost as much.

Thus, in the end, it turns out that the third method should be the best suited in most of the applications. It does not have problems of alignment, and the amount of data to process is acceptable if the scale of the scene is limited, as in this example. However, the amount of memory for storing the frames will increase with the complexity of the scenes and thus be an expensive solution for dense scenes. In the specific case where the camera is not allowed to move, then the first or second method might be best.

In the case of our demo program (and hence all the pictures of this article), we did not allow the camera to move, and so chose to implement the first technique. The results we got could easily apply to another technique though, as the main change is the way lookup coordinates are computed, and illumination textures are generated.

Up till now, we assumed that we had indirect illumination textures available for several time steps. It is now time to define how to get those images. This is what the next subsection will be about.

Tips and tricks for rendering images with indirect illumination only

Obtaining pictures with indirect illumination only has to be done before the program starts running. Since our method's only requirement is that indirect illumination pictures are available at runtime, the way these images are obtained is actually flexible: any method that provides the right output would work – and finding the optimal way to produce them might be an optimization in itself. As finding such an algorithm is a different subject, we will only present here the simple method we implemented, and that gives good results, while it might be overly simplified.

For producing the indirect illumination pictures the commercial modeling tool Autodesk 3DS max® is used. The model of Koldinghus used in the demo program is modeled in this tool that can also produce rendered images with lighting. The settings used are set up to be as similar as possible to the demo program so as to enable a comparison of the resulting illumination later on. These settings are with regards to light and material color and intensity etc. A point light source is inserted in the scene to act as the sun. A point light is chosen to get the same shadow casting as in the demo program where a point light is used in the shadow map pass and in the direct light computations. The sun is animated both in terms of movement where it's following a path resembling the sun's passing on a chosen day, and in terms of intensity to give a realistic sunrise and sunset. The skylight is added to render its contribution to the global illumination. The images for the two light sources are rendered separately as we want to add them to the final illumination separately. Firstly a render with the point light including only direct light is done, which is a sequence of images corresponding to a day. Next global illumination for this light source is rendered and another sequence is produced. These two sequences are subtracted to obtain the indirect illumination only, which is the image sequence that will be used as textures.

One image is only rendered for the skylight as it is, as mentioned earlier, a static light source which we just multiply by an intensity value. All the rendering is done with the state-of-the-art Mental ray renderer which gives physically realistic looking results of global illumination using photon maps and final gather to compute diffuse indirect illumination.

3.2 Refinements

Compressing the textures using PCA

We apply the principal component analysis (PCA) on the indirect illumination frames in the time-lapse sequence in order to reduce dimensionality and the amount of storage required to save the individual frames. With a resolution of 800×600 pixels and a frame every five minutes, this results in a uncompressed file size of about 400 MB for the total sequence. We use the PCA to represent as much of the uncompressed information as possible by maximizing the variance in the time-lapse sequence onto a lower-dimensional subspace.

The PCA is implemented on only 180 of the total 288 frames to save computational time and requirements. The discarded frames are all placed in the night and evening and contain only immerse darkness. The image vectors for the different frames are put together to produce 480 000 vectors with 180 dimensions, a vector for each pixel. When the PCA is applied on those vectors, we get 180 eigenvectors with variance as shown in Figure 5.

Taken into account that the variance is plotted onto a semi-logarithmic plot, we see that the amount of variance contained by the single eigenvector is dramatically decreasing as we go through the eigenvectors. With only five eigenvectors, we may thus capture 95 % of the total variance, and if we double the numbers of eigenvectors to ten, they contain 98.6 % of the variance in the

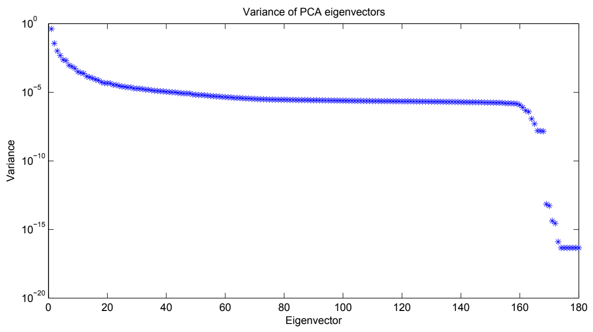


Figure 5: Variance of eigenvectors used for PCA

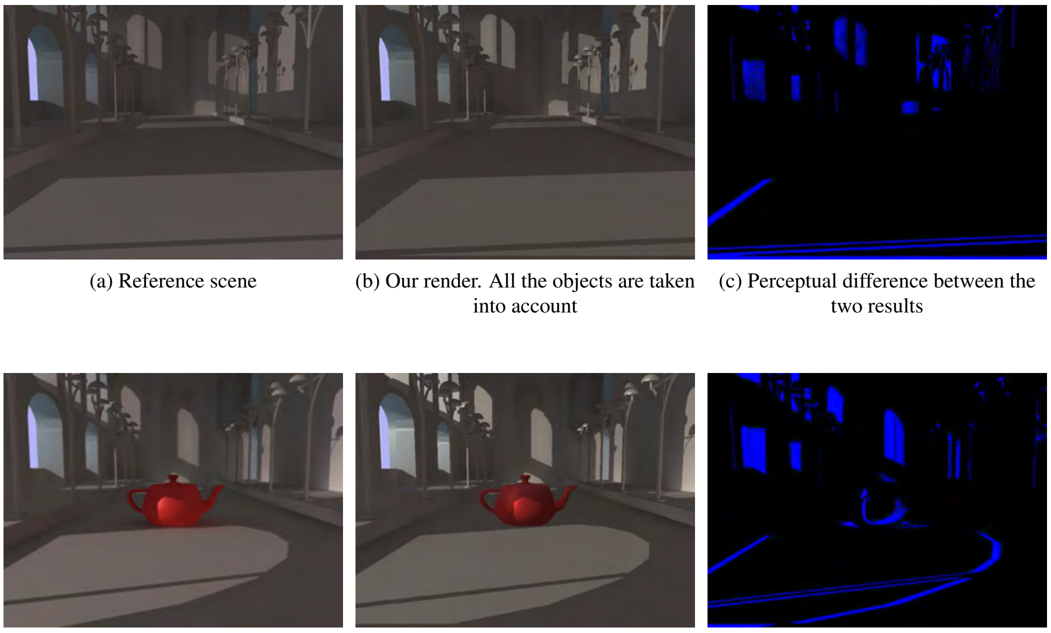
time-lapse sequence. In this case, we get a compression of $94,5\%$.

In order to get the image vectors onto the ten eigenvectors, the transposed of the assembled frame vectors are projected on each eigenvector. To restore the images, the reverse process is simply executed. With ten eigenvectors with a dimension of 180 and ten projected vectors with a dimension of 480 000, we are able to reconstruct the image sequence. The 121st reconstructed frame of this sequence is shown in Figure 7 on page 7 together with the original, with little noticeable difference. Further comparison of the frames is found in section 4.3 on page 8.

Dynamically select the sampling interval depending on light changes

As mentioned we want to have a realistic looking passing of a day with regards to the indirect illumination with the least amount of data. If we can discard some of the images which hold the indirect illumination, and instead interpolate images to produce the illumination, so that it is not noticeable for the user, then we can achieve exactly this lower data amount that is desirable. For this a algorithm has been produced which can be explained in some simple steps;

1. First we have two equally big intervals with images representing the day.
2. The first and the last image in each interval is linearly interpolated to a middle frame and the produced images is compared to the first and last one of the interval.
3. If the interpolated image is noticeably different from the first or last in the interval, the interval is split in two, and a new frame is rendered for the time of the interpolated image, the middle frame. This new frame becomes the first and last image in the two new intervals.
4. This continues for every interval produced until the images needed are rendered. With this relative simple algorithm you save the computation time of doing big and complicated renders of global illumination by only rendering the images needed to make


(a) Reference scene (b) Our render. All the objects are taken into account (c) Perceptual difference between the two results

(d) Reference scene (e) Our render. The teapot is added at runtime (f) Perceptual difference between the two results

Figure 5: Differences between our render and a reference. Top: all objects are taken into account for the indirect lighting. Bottom: dynamic objects are added

the indirect illumination look realistic for the sequence of the day.

The way the interpolated frame in an interval is compared to the first, is by utilizing a tool called PerceptualDiff [YCS+04]. This tool uses knowledge about the just noticeable difference [SJ10], in the field of color, to calculate the number of perceivable different pixel in the two images and then evaluates if a person is able to see a difference between the two images.

For the demo program the algorithm is used on the sequence of indirect illumination images which contains images for every five minutes. This reduces the sequence density and thereby the data loaded by the demo program, which also reduces the loading time on startup. Since the images now don't have the same spacing between them, an application like the demo program, using them also requires this info to interpolate the sequence correctly.

On Figure 6 the result of applying the algorithm is seen. The pictures at night are left out because they are totally black. We see that there is a lot of activity at sunrise and sunset, and also at two instances in between where the light source representing the sun goes behind a wall with no windows (mimicking sunrise and sunset). The peaks around 900 minutes are due to increased activity at the right wall with the lamp posts. The amount of images needed per hour varies from twelve, meaning that every frame in the sequence needs to be rendered, to intervals of 30 minutes which gives only two per hour.

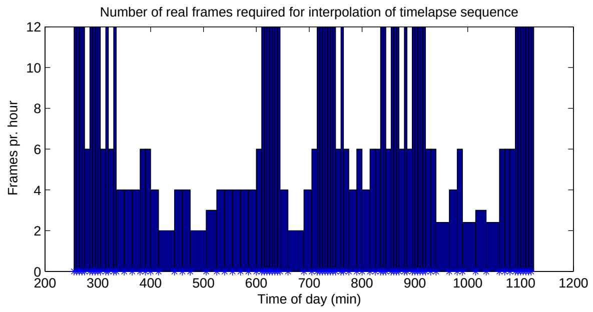
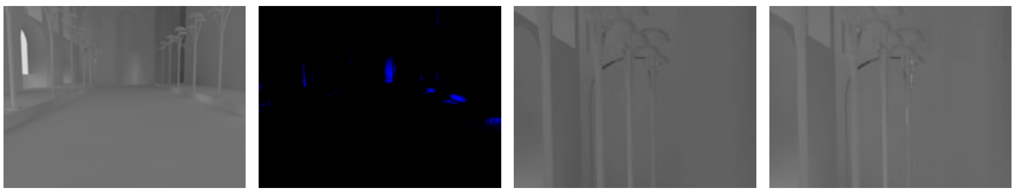


Figure 6: The dynamic sampling interval through the day

4 RESULTS AND DISCUSSION

The very first and most trivial way to analyze the results obtained is just to look at the rendered pictures and judge, with the naked eye, how they look. Figure 1, in section 2, shows a render of a random frame in our program, and compares it to the same scene, rendered in a state-of-the-art 3D program. At first glance, we can easily agree on the fact that both images look similar.

This, however, is not a sufficient way of analyzing results, and we will need a way to quantify the difference between the two techniques. In this section, we will consider the state-of-the-art render as the standard, and compare our results to this reference. For that, we will, like in 3.2, use the notion of perceptual difference [YN04], which describes the difference between two images in terms of how a human would perceive it. First of all, we will just analyze our basic renders



(a) Original frame (b) Perceptual difference of restored frame with 10 eigenvectors (c) Blow-up of region with 10 eigenvectors (d) Blow-up of region with 20 eigenvectors

Figure 7: Comparison of information loss produced by PCA. A gamma correction of 2 has been applied to the frames.

(with and without objects added dynamically), then in the next sections we will analyze the impact of the various compression methods on the final result.

4.1 Perceptual difference between our renders and a reference

As explained above, we will focus on the perceptual difference, in normal conditions, between an image rendered in our program and an image rendered using the commercial program 3DSMax by Autodesk. For that, we will use [YCS$^+$04], a utility program that computes the perceptual difference between two images. Its implementation is highly based on [YN04].

Figure 5 summarizes the results for two types of scene:

- The top row shows a scene where all the objects have been taken into account during the preprocessing stage, so that the indirect illumination is correct.
- The bottom row shows a scene where the red teapot has been added during runtime, i.e. it hasn't been taken into account while calculating the indirect illumination. In the reference picture, however, the whole scene is subject to indirect illumination, and so the teapot is included in it.

As we can see, in the first case, the perceptual difference is mainly due to minor differences in the way both scenes are rendered, because the light's intensities can't exactly match. The perceptual difference is then mainly due to these discrepancies in direct illumination, as well as a small geometrical mismatch in the areas covered by shadows. All the regions in shadow, lit by indirect illumination only, present no noticeable difference.

Unsurprisingly, results are similar when an object is added dynamically. In Figure 5f, we see that there are only few errors on the teapot itself as well as the nearby floor where the indirect illumination is approximated; the other discrepancies are due to the setup in both environments, as explained above.

4.2 Choosing the right interpolation function

One of the criteria to decide on when using our technique is: "what interpolation function to use?". In theory, that choice should influence the quality of the final render, and you might want to choose the most accurate interpolation method, to get the best results. In practice, however, we observe that this is not the case, as most interpolation functions need a lot of computation in order to choose the non-linear parameters, for results that are only slightly better than linear interpolation.

The main problem with most interpolation methods is that they need to run on the whole dataset to compute the parameters needed. In our work, that means that most images must be preprocessed to generate the interpolation function. Linear interpolation, however, only needs two images to compute a third one, which is the easiest function, albeit not very accurate.

Accuracy, however, is not really an issue with our dataset. As explained in section 2, we weeded out all the high frequency data from our images to only keep the low-frequency indirect illumination. That means that indirect illumination only present small changes between two frames, and so interpolating them should not provide major artifacts. Furthermore, if we use the Just Noticeable Difference as a threshold, then the interpolation should present no noticeable artifacts. Figure 8 shows results for such an interpolation graphically. A perceptual difference test on these images shows that no pixel is visibly different, although the arithmetic difference between the two images shows some minor interpolation errors.

4.3 Quality loss induced by PCA

In section 3.2, we described the method of reducing the dimensionality of 180 frames into only ten vectors. The 121'st frame in the sequence has been restored and contains 99.6 % of the variance of the original frame. However, as it might be seen from a the image of the perceptual difference in 7b and as a closer inspection of the frames reveals, differences in color nuances and transitions appears. This is apparent at the back wall of the church and on the floor where some transitions from light to darker grey is smoother in the original than the restored frame and other transitions does not appear. Further differences occur in the lamps when a hardly visible, small pocket of white in the distant lamp is converted to grey. A blow-up of this region is shown in the third frame of Figure 7.

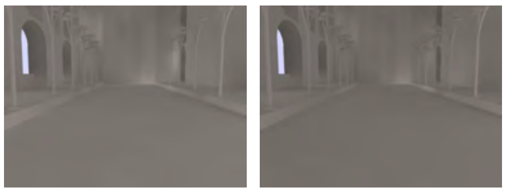

(a) Original frame (b) Interpolated frame

Figure 8: Comparison of a random frame and its interpolated counterpart.

In order to acquire a more vivid representation of the frame with more transitions and nuances imminent, the number of vectors used for compressing the frames is doubled to 20. The overall improvement of the sequence may be small however, as the variance only increases to 99.83%, a relative improvement of approximately 0.23 %. This small improvement in the total variance captured is visible on the blown-up frame on figure 7d as the original, white ray of light on the distant lamp now is apparent. Additionally, with this increase in the number of eigenvectors, there are no longer a perceptual difference between the precomputed and the estimated frames.

5 CONCLUSION

In conclusion, we see that our algorithm gives good results, and allows using total indirect illumination in an interactive context. If combined with a texture atlas to provide the indirect illumination, it allows for a camera to move freely within boundaries, and if programmed carefully, dynamic objects could also be added with only minor errors. Furthermore, selecting a varying amount of frames across the sequence, and compressing them with an algorithm such as PCA, allows getting similar results while greatly reducing the amount of data to store and process.

The main area of improvement of our algorithm, that could be the topic of another research work, is the way the indirect illumination textures are precomputed. Finding a way to have indirect illumination images precomputed as fast as possible might be interesting in most applications.

REFERENCES

[Chr99] Per H. Christensen. Faster photon map global illumination. *Journal of graphics tools*, 4/3:1–10, 1999.

[DRP09] Samuel Delepoulle, Christophe Renaud, and Philippe Preux. Light source storage and interpolation for global illumination: A neural solution. *Studies in Computational Intelligence*, 240/2009, 2009.

[NJTK95] Eihachiro Nakamae, Guofang Jiao, Katsumi Tadamura, and Fujiwa Kato. A model of skylight and calculation of its illuminance. *Image analysis applications and computer graphics*, 1024/1995:304–312, 1995.

[NPG03] Mangesh Nijasure, Sumanta Pattanaik, and Vineet Goel. Interactive global illumination in dynamic environments using commodity graphics hardware. In *11th Pacific Conference on Computer graphics and Applications*, pages 450–454, 2003.

[RGKS08] Tobias Ritschel, Thorsten Grosch, Jan Kautz, and Hans-Peter Seidel. Interactive global illumination based on coherent surface shadow maps. In *Proceedings of graphics interface 2008*, 2008.

[RGS09] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, 2009.

[SJ10] Melissa K. Stern and James H. Johnson. Just noticeable difference. *Corsini Encyclopedia of Psychology*, pages 1–2, 2010.

[WKB+02] Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. Interactive global illumination using fast ray tracing. In *EGRW '02 Proceedings of the 13th Eurographics workshop on Rendering*, 2002.

[YCS+04] Hector Yee, Scott Corley, Tobias Sauerwein, Jeff Breidenbach, Chris Foster, and Jim Tilander. Perceptual image diff. http://pdiff.sourceforge.net/, 2004.

[YN04] Yangli Hector Yee and Anna Newman. A perceptual metric for production testing. In *SIGGRAPH '04 ACM SIGGRAPH 2004 Sketches*, 2004.

BlurTags: spatially varying PSF estimation with out-of-focus patterns

Alexander Reuter
Goethe University Frankfurt, Germany
reuter@vsi.cs.uni-frankfurt.de

Hans-Peter Seidel
MPI Informatik, Germany
hpseidel@mpi-inf.mpg.de

Ivo Ihrke
Saarland University/ MPI Informatik, Germany
ihrke@mmci.uni-saarland.de

ABSTRACT

Current research is targeting the estimation and correction of lens imperfections. often modeled as a set of spatially varying point spread functions (PSFs). One way to measure these PSFs is their calibration with checkerboard patterns. Previous work, however, does not fully exploit all benefits of using a checkerboard. In particular, we show in this paper that the pose of the checkerboard with respect to the camera can be exploited to yield information on the circle of confusion, and thus the image blur of an ideal camera. By removing this expected blur, we can estimate residual PSFs that are due to the deviation of the optical system from a thin-lens model. The residual PSFs can then be used to sharpen images at comparable lens settings.
Practical side effects of our method are the design of a self-identifying pattern that can be robustly detected even in the case of image blur, and a corresponding algorithm for its detection.

Keywords
spatially varying PSF, self-identifying markers, deconvolution

1 INTRODUCTION

2012June 25 – June 28

The optical resolution of camera images is often limited by the quality of the lens system used for taking the picture. Even expensive lens systems show aberrations, especially at high aperture settings (low f-number). Alternatively, cheap plastic lenses are mass-produced and show less then ideal imaging capabilities. The deviation from perfect imaging is usually expressed by the system's point spread function (PSF). Though it is often assumed to be invariant with respect to shifts on the sensor and the settings of the camera, this is in general an invalid assumption. The space of PSFs is high-dimensional, varying with position on the sensor, aperture, focus and, potentially, zoom-setting of the camera. Knowing the distribution of the PSFs enables the computational removal of aberrations from the image via deconvolution.

It is therefore important to calibrate the PSFs, ideally for all possible combinations of camera settings. Using the EXIF information of an image taken with such a calibrated lens, it can be sharpened in a computational post-process. Recently, the work of Kee et al. [KPCW11] has introduced a method for estimating the PSF variation for aperture and zoom changes. However, they do not address the focus-setting of the camera. This leads to the problem that the calibration pattern has to be kept in-focus during the calibration process and thus, has to be moved.

In our approach, we take a different route. We exploit the planarity of the checkerboard to estimate its pose with respect to the camera. This, in turn, allows for estimating the circle of confusion, the PSF of an ideal thin lens. The observed calibration pattern is convolved with a different PSF, which we consider to be a combination of the ideal PSF and the in-focus PSF due to lens aberrations. Our goal is the estimation and removal of the latter. The resulting images are approximations to images that would be taken by a perfect lens system with a finite aperture.

Our algorithm relies on the ability to detect blurred patterns. For this reason, we design a checkerboard pattern that is well suited to estimate PSFs and which can be robustly detected in the presence of blur. We call the resulting pattern and detection routines *BlurTag*.

2 RELATED WORK

The correction of image blur is content of much research today. Usually, the goal is to correct for scene related degradations like motion blur [SJA08], out-of-focus blur [VBC$^+$02], or camera shake [FSH$^+$06]. Blur removal is a deconvolution problem which comes in two flavors: blind deconvolution e.g. [AD88, Car01] and deconvolution based on a known kernel e.g. [Ric72, Wie49]. Joshi et al. [JSK08] determine the kernel by edge prediction using the rationale that part of the blur kernel can be observed in a direction orthogonal to the edge. Based on this approach, Kee et al. [KPCW11] propose a non-blind approach to estimate and remove lens aberrations. They employ checkerboard-like patterns including circles that provide edges in all orientations to estimate the kernel. As mentioned earlier, the pattern has to be in focus for the method to yield unbiased estimates. Point spread functions may change when the focus changes. If the calibration image has been taken with a different focus setting from the one that is to be corrected the results will be inaccurate. Our work aims at relaxing this requirement.

Our approach is based on pose estimation with respect to a planar target, a standard problem in computer vision [Zha00]. An implementation is available in Bouguet's calibration toolbox [Bou02]. The toolbox estimates the common parameters related to a pinhole camera such as focal length, camera orientation, as well as radial distortion. In this area, it is common to use uncoded checkerboards where the user has to define the outer checkerboard corners manually.

In the field of augmented reality, tag-based systems such as [GGSC96, ZFN02, dIMH02] are often employed. These methods address the automated pose estimation problem, but do not consider the aspect of calibration. The focus is on the automatic detection and identification of markers. In general, the tags of augmented reality can be adapted to checkerboards by using them as codes. The approach of Atcheson et al. [AHH10] places two-dimensional codes into the squares of the checkerboard. To enhance the detection quality in blurry regions, the authors introduce a new design where codes are rotated and separated from the feature points. Due to this, the resolution of the codes is reduced. In addition, the pattern employs straight edges which are not suitable to sample general two-dimensional blur kernels densely. We therefore opt for an alternative design.

Our goal is the automatic estimation of spatially varying PSFs which are distributed over a captured image. We capture an image which provides a checkerboard pattern. This checkerboard pattern is optimized for

1. automatic detection as in the CALTag implementation [AHH10],

2. accurate PSF estimation as in [KPCW11], and

3. blur robust detection for out-of-focus checkerboards.

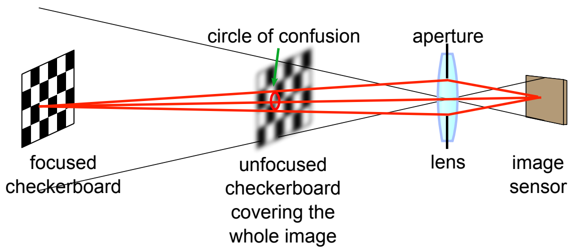


Figure 1: In our setup a calibration pattern is imaged at an out-of-focus position. The circle of confusion is blurring our image in addition to the lens PSF at the in-focus setting.

Further, we want to estimate the pose of the checkerboard. Knowing the distance of the focal plane as well as the aperture and focal length, we are able to estimate the circle of confusion of the image regions covered by the checkerboard pattern. These circles of confusion offer the ability to compute in-focus PSFs from out-of-focus PSFs.

In summary, we combine and extend different aspects of previous work in order to provide a practical solution to spatially varying PSF estimation for lens-induced aberrations.

3 OVERVIEW

The goal of our approach is the calibration of the spatially varying PSFs for different focus settings of an optical system using a static calibration pattern at a fixed position. The main motivation for this setting is that it is difficult to move a single calibration pattern to different focal planes while maintaining a full coverage of the image and a comparable apparent resolution of the target at different distances without changing the pattern on the target.

We therefore choose to image the calibration pattern in an out-of-focus position, see Fig. 1. Consider first the checkerboard to be placed in the focal plane. In this case, an ideal, finite aperture, thin-lens imaging system would produce a perfect image since the system response of the thin lens is Dirac. In a real system, however, the system response is different and the observed image I_{obs}^{IF} is given by the object space pattern I_{obj}, convolved with the system PSF p:

$$I_{obs}^{IF} = I_{obj} \otimes p, \tag{1}$$

where IF stands for "in-focus". In general, the convolution has a spatially varying kernel $p(x,y)$. If the pattern is placed in a different position, a finite aperture, thin-lens system will produce out-of-focus blur which is usually described by the circle of confusion, a special PSF which is a scaled version of the aperture shape. The ideal image I_{obs}^{TLOF} under these circumstances is given by

$$I_{obs}^{TLOF} = I_{obj} \otimes p_{coc}. \tag{2}$$

Here, p_{coc} is the circle of confusion PSF and TLOF stands for "thin-lens out of focus". Again, as in the in-focus case, a real system does not produce I^{TLOF}_{obs} since its PSF is not an ideal circle of confusion which would have a pill-box shape (assuming a circular aperture). Instead, the out-of-focus PSF of the real system is given by $p_{coc} \otimes p$. The observed out-of-focus image I^{OF}_{obs} in a real system is thus given by

$$I^{OF}_{obs} = I_{obj} \otimes p_{coc} \otimes p. \qquad (3)$$

Our method aims at estimating p, given the observed out-of-focus image I^{OF}_{obs} and the pattern definition I_{obj}. In order to make this problem tractable we need a reliable way of estimating p_{coc}. We also aim at relaxing the condition that the pattern has to be placed parallel to the image plane.

The above requirements can be fulfilled if the pose of a (planar) target with respect to the camera can be estimated from the calibration image I^{OF}_{obs}. Knowing the pose, and the aperture and focus settings of the camera, $p_{coc}(x,y)$ can be computed at every position in the image. This implies, that we can recover the PSF of the in-focus plane $p(x,y)$ by inverting Eq. 3.

In practice, we compute the orientation of the planar target by tracking a checkerboard coded with self-identifying markers (tags). This automates the calibration process which is important if several focus settings are to be processed, potentially for several aperture/zoom settings of the camera. In this article, we only discuss the computation of the PSFs from a single image. Calibrating an arrangement of settings would apply our method to each image independently.

The processing pipeline is shown in Fig. 2. First, we detect the individual squares of the checkerboard pattern and verify the contained tags. Then we estimate the PSFs, which are blurred by the circle of confusion. In parallel, we estimate the pose of the checkerboard, which can also be regarded as the extrinsic parameters of the camera with respect to the checkerboard. We can extract the focal plane parameters from the EXIF header provided by the captured image. The combination of pose information and focal plane parameters allows us to compute the circle of confusion for each position covered by the checkerboard pattern. Finally, we deconvolve each PSF with the related circle of confusion. The result is a full image covering set of PSFs for an image in the focal plane of the camera.

In order for this process to work well, we have to regard two aspects: First, a checkerboard design that is well suited to PSF estimation. Second, a detection algorithm that is robust to blur.

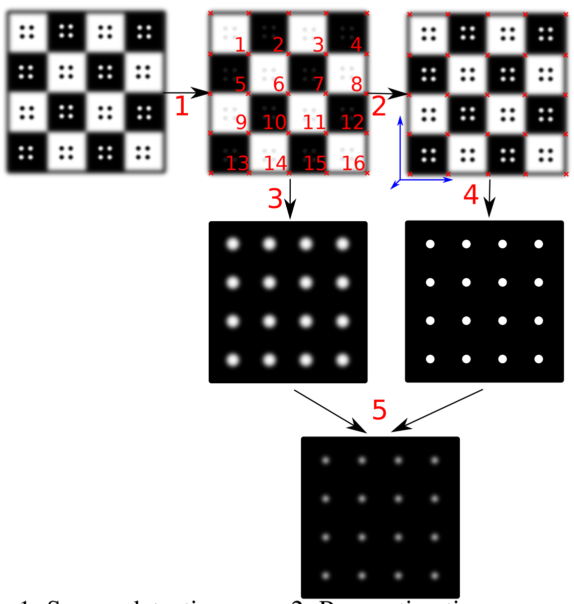
4 CHECKERBOARD DESIGN

Estimating the PSF distribution based on a captured checkerboard pattern requires its initial detection. In our approach, the checkerboard detection fulfills two goals at once: First, we want to estimate one PSF for each checkerboard square, and second, we want to estimate the pose of the checkerboard for estimating the circle of confusion. Based on these two goals, we can derive criteria, which have to be met by our checkerboard design:

1. We seek to estimate accurate PSFs,
2. the detection must be robust to blur, and
3. the process should be automatic.

Based on these criteria, we derive a checkerboard pattern (Fig. 5, upper left) providing a code for each checkerboard square, for an example see Fig. 3.

For our initial tests we use a simple binary code. This code could be made error-correcting as e.g. in [AHH10] by sacrificing some bits. The codes, however, are not only used to identify a marker, but also to serve as a pattern for PSF estimation, i.e. I_{obj} in Eq. 3. It has been observed in previous work that it is important to use patterns that provide edges in every direction [JSK08]. We therefore choose circles as the basic elements of our codes. Instead of coding a bit as circle present (1) vs. circle absent (0) in a particular position, we use an inner-/outer-circle hierarchy, where the presence of an inner circle indicates a logical 'one', see Fig. 3. This measure ensures the presence of edges even for codes that contain many zeros and thus improves the stability of the PSF estimation.



1. Square detection 2. Pose estimation
3. PSF estimation 4. CoC estimation
5. PSF sharpening

Figure 2: Overview of our approach

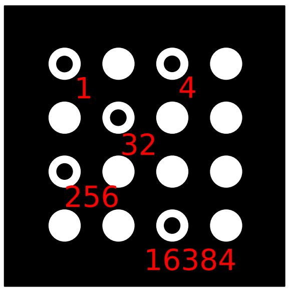


Figure 3: Sample code with active bits 2^0, 2^2, 2^5, 2^8 and 2^{14} leading to the ID 16677. The bits are using a lexicographical order. Codes are unique under rotation and mirroring.

Further, we decided to make the codes unique in terms of rotation and mirroring by removing symmetric ones from the list of available codes. This property is simplifying the detection process. Knowing the orientation helps when assembling a complete checkerboard pattern from single detected squares. We are dealing with two dimensional codes providing w horizontal and h vertical code elements. For two reasons, we decided to set $w = h = 4$. First, we are dealing with squares so we set $w = h$ to distribute the corresponding objects homogeneously. Second, regarding uniqueness with respect to symmetries and robustness to errors, 3×3 codes offer too few candidates to fill a reasonably sized checkerboard pattern with unique codes. On the other hand, 5×5 codes become too dense for small resolutions in the captured image.

5 IMPLEMENTATION

The implementation of our algorithm is split into several parts. At first, we detect the checkerboard and identify the markers, Sect. 5.1. The important aspect in this step is the robustness of the detection routine to blur. Next, we compute the combined PSFs $p_{coc} \otimes p$ from the image and the pre-defined pattern, Sect. 5.2. In order to remove the influence of the circle of confusion p_{coc}, we compute the pose of the checkerboard and estimate p_{coc}, Sect. 5.3. Finally, we compute the in-focus PSFs p by deconvolving the combined PSFs $p_{coc} \otimes p$ with the circle of confusion PSF p_{coc}, Sect. 5.4.

5.1 Detection process

The critical aspect in the detection process is the identification of the individual checkerboard squares in the presence of blur. The main difficulty is the apparent merging of neighboring regions of the pattern [AHH10]. The main differentiating aspect of our detection scheme is that we interpret the thresholded image as a hierarchical tree structure: a connected component is a child of another if it is completely included in the other. This feature lets us remove parts of the image (the codes) for the early parts of the detection routine. This way, we can focus on separating the checkerboard squares without fearing to damage the codes inside, see Fig. 4.

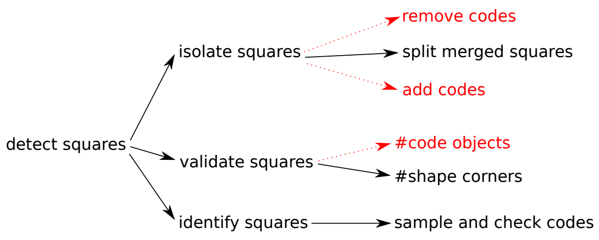


Figure 4: Overview of the detection process.

An overview of the processing pipeline is shown in Fig. 5. The number of the processing step refers to the number of the subfigures in the pipeline overview. In the following, we describe the individual steps in more detail.

1. Thresholding the image

Checkerboard patterns are usually black and white to simplify the detection process. During capture, ambient light, shades and light gradients lead to a gray-scale checkerboard pattern. Our first step is to binarize the image via thresholding. We use a median based adaptive thresholding algorithm [Jai89]. The definition of the window size is the most critical parameter within the process. Too small windows lead to noisy results while too large windows may lead to shrunk, open, or vanishing circles. In our implementation, we use a window width of about 4 to 5 percent of the image width. After that we remove noise from the thresholded image.

2. Removing circles

In this step, we remove the circles that represent the code of a square from the binary image. The result is a binary image without code structures as in Fig 5 (2). This leads to significant improvements in the following *splitting* step because the fine detail of the code structures can be ignored. As discussed above, we interpret the thresholded image as a tree of connected components, Fig. 6. The tree construction is performed in two steps. First, the binary image is decomposed into connected components. Second, the connected components are organized in the component tree [NC06]. The component tree encodes the complete interrelations between the nested structures. Using it, we can remove the circles by removing the corresponding nodes. In our pattern, we remove the nodes which provide exactly one leaf and the leafs themselves. After that, we draw the tree again as a thresholded image, this time without the circles.

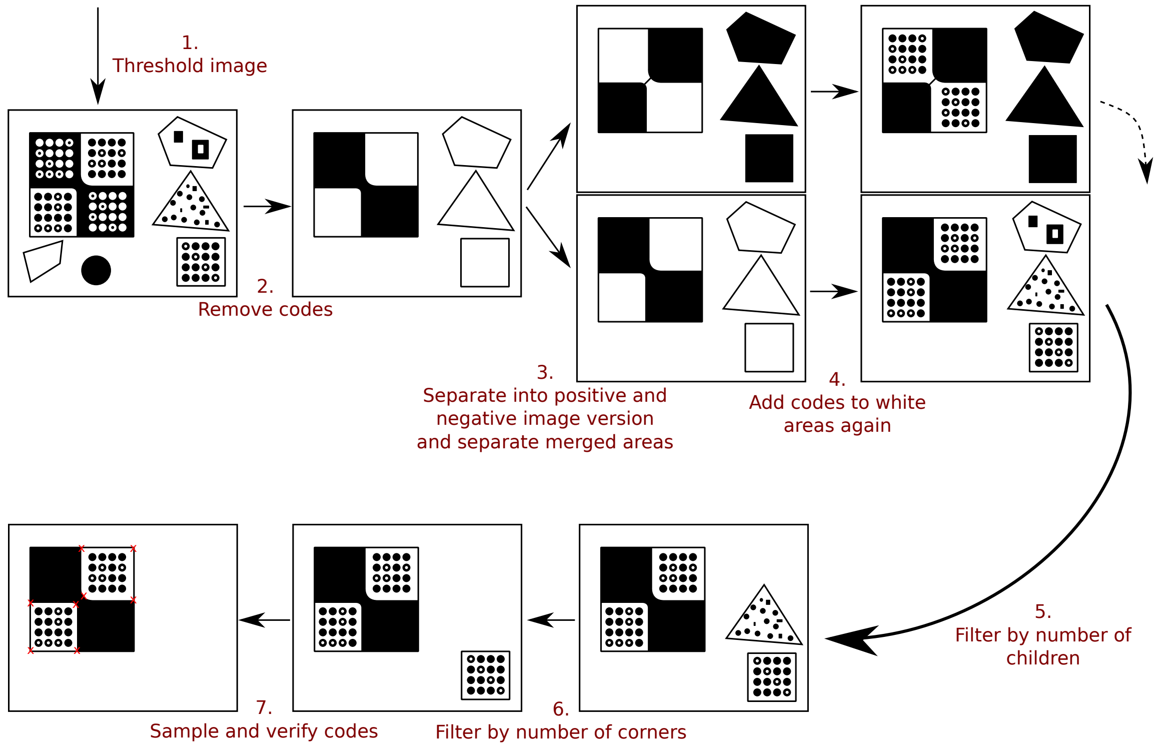

1.
Threshold image
2.
Remove codes
3.
Separate into positive and negative image version and separate merged areas
4.
Add codes to white areas again
5.
Filter by number of children
6.
Filter by number of corners
7.
Sample and verify codes

Figure 5: The tracking pipeline.

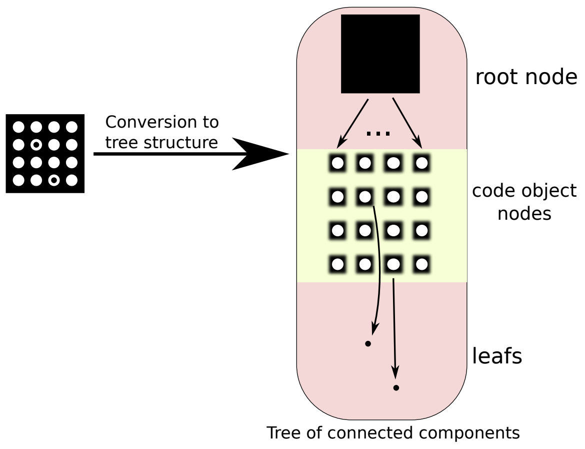


Figure 6: Conversion from pattern to tree structure. The hierarchy of the connected components is encoded as parent/children relations in the tree.

3. Splitting adjacent squares

At this step we have a binary image of the checkerboard without the circles. However, due to image blur the initial thresholding step results in merged squares, i.e. squares of equal color meeting at one corner appear as a single connected component. The goal in this step is to split these structures.

First, we determine the dominant points for each structure in the image by using the IPAN algorithm [CV98]. We use an angle threshold of 160°. Now we search for pairs of dominant points that only have a small distance from each other. To specify this distance we use a threshold T_x. The goal is to identify point pairs on opposite sides of a “bridge”, i.e. an area where squares are merged. If T_x is chosen too large, squares can be split diagonally which has to be avoided to preserve their quadrilateral shape. For this reason we set $T_x = 20$ pixels which is related to the minimal diagonal size of usable squares. A square is usable if each circle can be detected. Consider the minimal case: the cross-section along the diagonal of a square provides 4×2 outer circle edges + 4 inner circles + 3 spaces between the circles and 2×2 parts between the outer circles and the square border. Assuming that each part consists of at least one pixel, the resulting minimal diagonal length is $2 \times 4 + 4 + 3 + 2 \times 2 = 19$ pixels.

After having identified the candidates for splitting we perform this operation by drawing lines between neighboring dominant points. The resulting binary image contains what our algorithm believes to be squares. In order to avoid separate implementations for the white and the black squares we work with two copies of the image, one being the inverse of the other. In the following, we describe the further processing for only one of those images.

4. Adding circles

At this point, we have a superset of square candidates. Our goal is to filter out all false positives. We achieve this by observing the inner structures of each square candidate. For this, we add the circles that were removed in the second step back into the image. This can be done by taking the original thresholded image (step 1) and the negated result of step 3 and performing a logical 'and' operation between the two images.

5. Filtering by number of circles

Our checkerboard pattern provides only squares with 16 circles. We take advantage of this fact by filtering out all candidates which do not meet this requirement. In our implementation, we convert the square candidate image to a component tree as in step 2. Now, we can traverse the tree and check the number of children of all connected components. All squares which do not provide 16 children are filtered from the tree. After that, the filtered tree is converted back to an image.

6. Filtering squares by number of corners

In this step we filter out all shapes that do not have four corners. They tend to be no squares. We compute the potential corners by a variant of the IPAN algorithm. We then identify the point with the steepest angle less than 135°. This threshold is used to avoid treating edges that have accidently been split by a dominant point. These points usually make angles close to 180° with their neighbors. In addition, if a perspective projection is so strong as for the 90° corner to project to an angle of larger than 135°, then the internal code of the square is likely to be of very low resolution and thus unusable. The output of this step is an image that contains square candidates with exactly 4 corners and 16 code objects.

7. Sampling the codes and verification of the squares

Next, we estimate the homography between the detected quadrilateral and an ideal two-dimensional $[0,1]^2$ space, [HZ10] which is used for sampling the code.

However, the corners computed so far are of low quality since they have been computed from the binary image. We increase the quality of the estimated points by merging close points of different squares to their mean position. It is important to ensure, that these points share the same real point. In addition we apply the Harris corner detector [HS88] on the gray value image.

5.2 PSF estimation

As outlined in Sect. 3, we can compute the combined PSF $p_{coc} \otimes p$ by deconvolving the observed image I_{obs}^{OF} with the definition of the pattern I_{obj}. For this it is necessary to remove the perspective projection of the recorded pattern, or, alternatively to warp the pre-defined pattern into the pose of the checkerboard. We choose the latter option to compute the PSFs directly in screen space. We do this by applying the inverse of the homographies computed in step 7 of the detection scheme to the pre-defined pattern. This process is performed independently for each square of the checkerboard. We use Wiener deconvolution [Wie49] to solve for $p_{coc} \otimes p$.

5.3 Pose and CoC estimation

As outlined above, the pose of the checkerboard allows for the estimation of the circle of confusion, the PSF of an ideal finite aperture camera system. Computing the checkerboard pose is equivalent to determining the camera's extrinsic parameters. We re-use the estimated homographies for this purpose. Coupled with the known real world size of each checkerboard square and the camera intrinsics, we derive the position and orientation of the checkerboard with respect to the camera [HZ10].

Based on the pose of the checkerboard, we compute the distance d between a selected point on the pattern and the camera. With the known distance of the focal plane S, the aperture diameter A and the focal length f we can compute the circle of confusion as

$$c = A\frac{|d-S|}{d}\frac{f}{S-f} \tag{4}$$

The circle of confusion diameter c needs to be converted to pixel size. Therefore, we need to regard the pixel dimensions of the camera sensor (e.g., Canon EOS 5D Mark II: 6.41 µm). The circle of confusion estimation can be applied at each position within the image, which is covered by the captured checkerboard pattern. The information on aperture size and focal length, and focal distance can be extracted from the EXIF tags accompanying the captured images.

5.4 In-focus PSF estimation

Once the circle of confusion p_{coc} and the combined PSF $p_{coc} \otimes p$ are known, the in-focus PSF p can be recovered by deconvolution. We again employ Wiener deconvolution for this task. A validation example of our procedure is shown in Fig. 7. In the first row we show ground truth PSFs obtained by applying Eq. 1. The checkerboard is placed in the focal plane which is placed at a distance of about $2.5m$. Due to the large field-of-view, the checkerboard cannot cover the complete image. In the second row we have moved the checkerboard to a much closer distance of about $30cm$. Here, the checkerboard covers the full image but appears blurry. The insets show the same region on the checkerboard. Correspondingly, the estimated combined PSFs are much larger than in the first row. The third row of the figure shows the estimated circles of confusion as well as the in-focus PSFs computed using our method. They agree quite well with the ground truth of the first row.

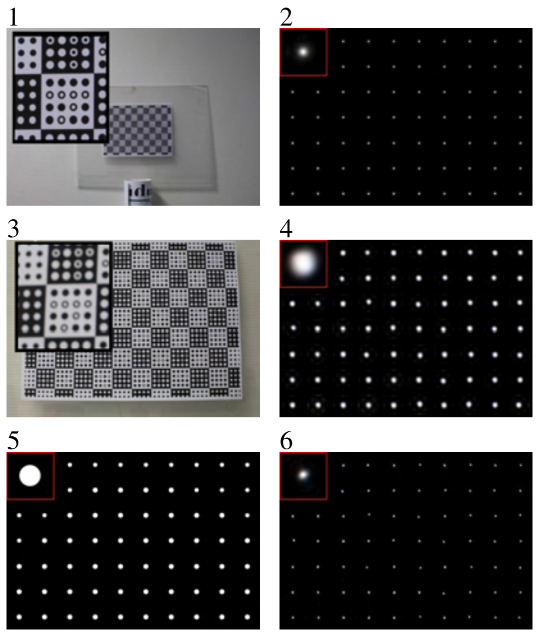


Figure 7: Estimation process of in-focus PSFs
1. Well focused, checkerboard in focal plane (~250cm)
2. Estimated PSFs from 1. incl. close-up
3. Same focus as in 1., but differently positioned checkerboard (~30 cm)
4. Estimated PSFs of 3.
5. Estimated circles of confusion of 3.
6. In-focus PSF estimated from 3.

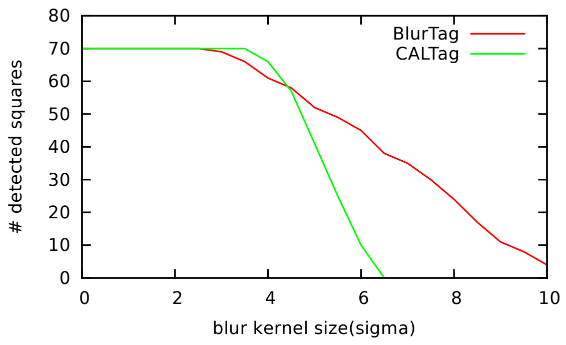


Figure 8: Comparison of BlurTag and CALTag

6 RESULTS AND DISCUSSION

We perform a number of tests to evaluate our algorithm. First we verify the detection process. Each square, which we fail to detect leads to a missing PSF. So the quality of our detection algorithm is critical. Further discussion about interpolating missing PSFs can be found in [KPCW11]. As mentioned in the overview section, we tend to capture blurry images which are challenging to process. For this reason, we implement tests to compare its stability with recent work.

6.1 Robustness to blur

Our first test compares our BlurTag with CALTag, the recent approach of [AHH10]. We generate synthetic images using our pattern and the one proposed

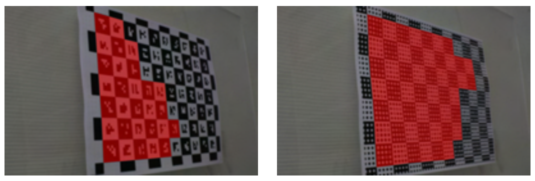


Figure 9: Detected squares in CALTag vs. BlurTag.
Left: Detected squares of CALTag-based checkerboard.
Right: Result of BlurTag-based checkerboard.

in [AHH10]. Both synthesized images provide the same amount of squares and the same perspective transformation. During the test, we successively increase the blur. The number of detected squares versus the size of the blur kernel (sigma) is shown in Figure 8.

We see that CALTag and BlurTag have comparable detection results when the kernel size is small. Initially, BlurTag's performance decreases earlier than CALTag's due to the increase complexity of the circle pattern as compared to the squares of CALTag. Circles require a higher resolution to work properly compared to square based codes. With larger blur kernels, however, we see that CALTag's results deteriorate more quickly as compared to BlurTag. The conclusion of this test is that CALTag is preferable when the camera resolution is limited and the amount of blur is small. BlurTag shows its strengths with an increasing amount of blur at sufficient image resolution.

This observation is also manifest in a real-world test, Fig. 9. Large variations in square resolution, combined with significant amounts of blur lead to a reduced performance. However, The impact on BlurTag's performance is smaller than for CALTag. Whereas CALTag is able to detect $29/70 \approx 41\%$ squares, BlurTag extracts $52/70 \approx 74\%$ squares.

6.2 PSF quality for blurred capture vs. ground truth

The major goal of our approach is the estimation of circles of confusion to avoid their impact on the estimated PSFs. In Fig. 7 we show the final, in-focus PSFs (6) as compared to PSFs estimated from a pattern recorded in the focal plane (2). We see that the in-focus PSFs are in close agreement with those, which we estimated with a well focused checkerboard. It is important to keep in mind, that the focus was not changed during the experiment.

For numerical comparison, we compute the RMS error between the ground truth PSFs p of the in-focus recorded checkerboard (computed via Eq. 1) against the combined PSFs $p_{coc} \otimes p$ as well as the estimates in-focus PSFs using our method, Fig, 10. We see that the quality increase obtained from our method becomes better with an increasing diameter of the circle of confusion, i.e. the larger the blur, the more benefit to using

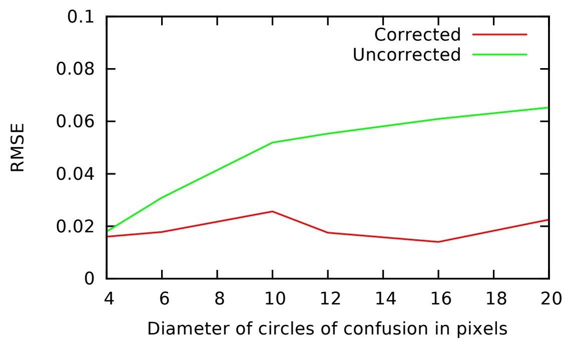


Figure 10: Root mean square error test:
Error between corrected or uncorrected PSFs and in-focus PSFs

our method. Another insight is that our computed in-focus PSFs have an approximately constant quality with respect to the ground truth over the tested blur sizes.

6.3 Video rates

While our main focus is on improved and more flexible PSF estimation, our method is fast enough to run at interactive rates and could thus be of interest for blur-resistant tracking in applications where AR-markers are traditionally used. We run on a test on a video with size 640×480 pixels. We achieve a mean speed of 13.04 fps. Our unoptimized implementation runs in a single-threaded process on an AMD Athlon(tm) 64 X2 Dual Core Processor 4600+. It should be possible to parallelize the algorithm on a GPU to achieve real time performance.

6.4 Application

Finally, we show results for sharpening a given target image using our estimated PSFs. Fig. 11 shows the application to a 22 Mpixel image taken with a Canon 5D mark II camera and a 50mm f/1.4 lens. We see that blur can be removed and the final image becomes sharper. Especially structures with high gradients like the handrails and the flowers improve in sharpness.

7 CONCLUSIONS AND FUTURE WORK

In this work, we introduced a new checkerboard pattern which is well suited to PSF estimation. Further, we introduced a new tracking algorithm, which is robust to resolution variations and large amounts of blur. In addition, we exploited the planarity of checkerboards to estimate out-of-focus blur of the calibration pattern. This allowed us to generate 'sharp' PSFs independent of the focus setting of the camera.

Our tracking approach is directly tailored to the designed checkerboard pattern. Therefore, changes in the design require different detection algorithms as well. In general, it would be possible to analyze the checkerboard patterns automatically to derive an

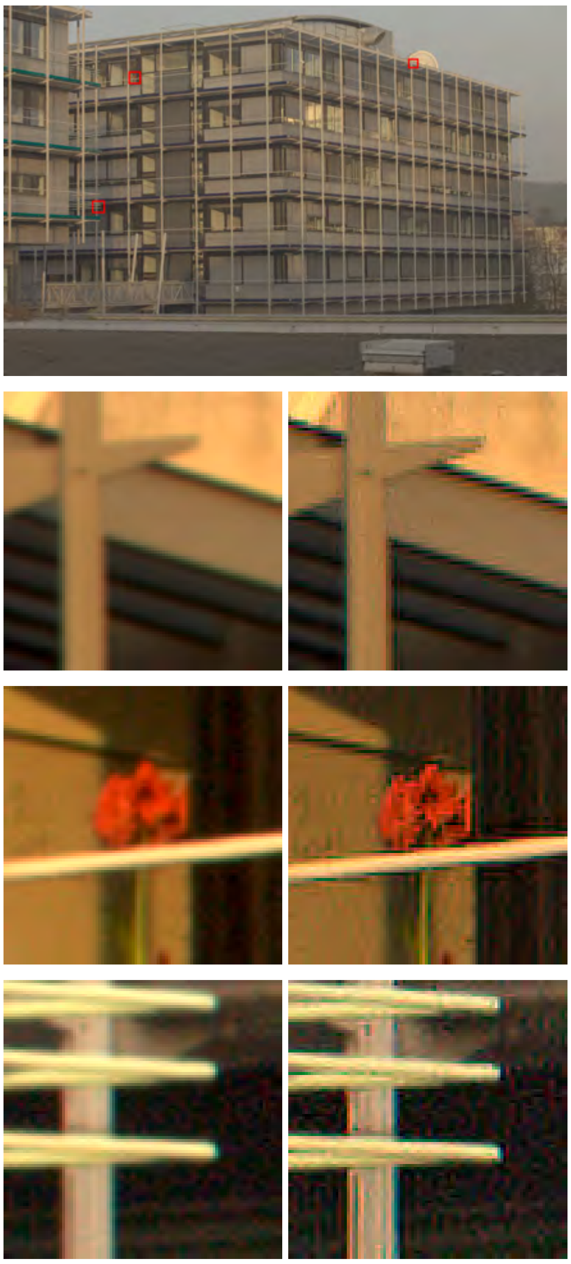


Figure 11: Results after deconvolution
Top: Target image with marked regions (best viewed in color)
Left: Blurred captured image regions
Right: Corresponding deconvolved image regions

appropriate tracking algorithm by analyzing its tree of connected components, as we did in the second step of our pipeline.

Further, it would be interesting to analyze the shape of a PSF under varying focus settings. This could help manufacturers to enhance the quality of their lens systems with respect to different focus settings.

8 REFERENCES

[AD88] G.R. Ayers and J.C. Dainty. Iterative blind deconvolution method and its applications. *Optics letters*, 13(7):547–549, 1988.

[AHH10] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. Caltag: High precision fiducial markers for camera calibration. In *VMV'10*, pages 41–48, 2010.

[Bou02] J.Y. Bouguet. Complete camera calibration toolbox for Matlab. *Retrieved from the World Wide Web: http://www. vision. caltech. edu/bouguetj/calib doc/index. html*, 2002.

[Car01] Alfred S. Carasso. Direct blind deconvolution. *SIAM Journal on Applied Mathematics*, 61(6):pp. 1980–2007, 2001.

[CV98] D. Chetverikov and J. Verestoy. Tracking feature points: a new algorithm. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1436–1438. IEEE, 1998.

[dIMH02] D.L. de Ipiña, P.R.S. Mendonça, and A. Hopper. Trip: A low-cost vision-based location system for ubiquitous computing. *Personal and Ubiquitous Computing*, 6(3):206–219, 2002.

[FSH+06] R. Fergus, B. Singh, A. Hertzmann, S.T. Roweis, and W.T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics (TOG)*, 25(3):787–794, 2006.

[GGSC96] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM, 1996.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[HZ10] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision 2nd ed.* Cambridge Univ Press, 2010.

[Jai89] A.K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.

[JSK08] Neel Joshi, Richard Szeliski, and David J. Kriegman. PSF estimation using sharp edge prediction. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2008.

[KPCW11] E. Kee, S. Paris, S. Chen, and J. Wang. Modeling and removing spatially-varying optical blur. In *Computational Photography (ICCP), 2011 IEEE International Conference on*, pages 1–8. IEEE, 2011.

[NC06] Laurent Najman and Michel Couprie. Building the Component Tree in Quasi-Linear Time. *Transactions on Image Processing*, 15(11):3531–3539, 20006.

[Ric72] W.H. Richardson. Bayesian-based iterative method of image restoration. *JOSA*, 62(1):55–59, 1972.

[SJA08] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. In *ACM SIGGRAPH 2008*, pages 1–10. ACM, 2008.

[VBC+02] P. Vivirito, S. Battiato, S. Curti, M. La Cascia, and R. Pirrone. Restoration of out of focus images based on circle of confusion estimate. In *Proceedings of SPIE 47th Annual Meeting*, volume 4790, pages 408–416, 2002.

[Wie49] N Wiener. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*, volume 47. Wiley, 1949.

[ZFN02] X. Zhang, S. Fronz, and N. Navab. Visual marker detection and decoding in AR systems: A comparative study. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 97. IEEE Computer Society, 2002.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

3D Models from the Black Box: Investigating the Current State of Image-Based Modeling

Hoang Minh Nguyen
The University of Auckland, New Zealand
hngu039@aucklanduni.ac.nz

Burkhard Wünsche
The University of Auckland, New Zealand
burkhard@cs.auckland.ac.nz

Patrice Delmas
The University of Auckland, New Zealand
p.delmas@cs.auckland.ac.nz

Christof Lutteroth
The University of Auckland, New Zealand
lutteroth@cs.auckland.ac.nz

ABSTRACT

3D models have become an essential part of many applications ranging from computer games to architectural design, virtual heritage, and visual impact studies. Traditionally, 3D model creation is done using modelling systems such as Maya or Blender. However, these systems have a steep learning curve and require a considerable amount of training to use. Thus, there is a critical need for tools which allow non-expert users to easily and efficiently create complex 3D scenes. To answer to that demand, a number of commercial image-based modelling packages have been introduced recently. Such software offers a very intuitive means to create 3D models from a sequence of images. However, the algorithms employed by these systems are usually kept secret, which makes it difficult to compare them algorithmically and identify common underlying concepts. This paper evaluates the most promising 3D reconstruction software packages with regard to efficiency, accuracy, limitations, constraints and compares them with a system developed by us in order to give an insight into their performance. To achieve that, we first describe our own 3D reconstruction system as a reference in order to make deductions about common concepts and differences. Then, we use a set of benchmark datasets to evaluate all considered systems, and gage their limitations with regard to the number of input images they need and the image resolution. Our evaluation shows that as the number of input images decreases, the geometry of models created using correspondence-based approaches contains more holes. However, the structure and geometry still reflect the original model. In contrast, silhouette-based methods produce coarse and distorted geometry as the number of input images decreases. Models obtained using silhouette-based methods from few input images are often unrecognizable.

Keywords
image-based modeling, correspondence-based reconstruction, silhouette-based reconstruction

1 INTRODUCTION

Creating 3D models of a scene has long been an important task in computer graphics. While conventional geometry-based modeling approaches enable the construction of highly realistic and complex 3D models via interaction with 3D meshes, they have a steep learning curve and require a considerable amount of training to use. These restrictions render them unsuitable for non-expert users. The recent advancement in hardware specialized in 3D model reconstruction has made it possible for non-professionals to reconstruct 3D scenes. However, tools such as laser scanners and structured lighting systems are often costly, have a limited range and resolution, are not very portable and flexible to use. Additionally, they have constraints with respect to material properties and environmental conditions such as string sunlight.

Digital cameras overcome many of these limitations and their ubiquitous use and integration into computing devices such as smart phones is making them an increasingly attractive proposition for 3D scene reconstruction. Recovering 3D structure from photographic images is an efficient and intuitive way to create 3D digital models of objects. Compared with conventional geometry-based modeling and hardware-heavy approaches, the image-based modeling method can be employed to extract original texture and illumination directly from images for visual 3D modeling, without the need for complicated processes, such as geometry modeling, shading and ray tracing. The techniques are usually less accurate, but offer very intuitive and low-cost methods for reconstructing 3D scenes and models.

The past few years have seen significant progress toward automatic creation of 3D models. There are now a number of software packages that offer the ability to acquire 3D models from a set of images without any a priori information about the scene to be reconstructed. Once supplied with the input images, these systems automatically process and produce a 3D model. As the algorithms used by these systems to reconstruct 3D models are usually kept secret, it is difficult to identify fundamental limitations and commonalities, and hence to compare them on an algorithmic level. The objective of this paper is to give an insight into the performance of the currently best-performing systems by evaluating them using benchmark datasets. We use the following methodology to provide some insight into the current state of image-based modeling: we include a reference system that allows us to make grounded assumptions on algorithmic differences, and we systematically vary the number of input images and their resolution to identify the current limitations.

After a description of related work on image-based modeling, a brief overview of our reference system is presented. In the second part, a set of benchmark datasets is used to stress test these systems under various conditions.

2 RELATED WORK

Ideally, image-based modeling algorithms can be categorized depending on the visual cues employed to perform reconstruction, i.e. silhouettes, texture, transparency, defocus, shading or correspondence. Traditionally, the most well-known and successful visual cues have been shading, silhouettes, and correspondence [HVC08]. Silhouettes and correspondence offer the highest degree of robustness due to their invariance to illumination changes. The shading cue requires more control over the illumination environment, but can produce excellent results [HVC08]. However, the requirement for strict constraints over lighting conditions renders shape from shading impractical for general applications.

The shape from silhouette class of algorithms is very efficient and has been proved to be stable with regard to object surface properties (color, texture and material). It is, however, very limited in the object geometries it can handle [FLB06, MBR+00, NWDL11]. The earliest attempt of using silhouettes for 3D shape reconstruction was made by Baumgart in 1974. In his pioneering work [Bau74], Baumgart exploited silhouette information from four input images to compute the 3D shapes of a baby doll and a toy horse. Following Baumgart's work, many different variations of the shape from silhouette paradigm have been proposed

Grauman et al. [GSD03] presented a Bayesian approach to account and compensate for errors introduced as the result of false segmentation. The approach has been shown to produce excellent error-compensated models from erroneous silhouette information. The disadvantage of this method is that it requires prior knowledge about the objects to be reconstructed and large ground-truth training data. This makes them impractical for general applications.

Cheung et al. [CA84, mCBK05a, mCBK05b] proposed a method that aligns multiple silhouette images of a non-rigidly moving object over time in an attempt to improve the quality of the constructed visual hull. Their method showed a significant improvement in reconstruction quality over previous methods.

Amongst the vast body of literature available on image-based modeling techniques, recent work on multiple view reconstruction has become a growing area of interest with many different techniques achieving a high degree of accuracy. These techniques are based mainly on correspondence cues and focus on producing models that resemble the original 3D scene from a sequence of calibrated or uncalibrated images. The concept underpinning these techniques is the extraction and combination of information from several overlapping images taken from distinct locations at different instants to deduce the relations between those images. When relations between images are properly established, the 3D structure of the observed scene can be inferred.

One of the most famous and successful reconstruction systems is the Façade system, which was proposed by Debevec et al. [DTM96]. The Façade system was designed to model and render simple architectural scenes by combining a hybrid geometric and image-based approach. The system requires only a few images and some known geometric parameters. It was used to reconstruct compelling fly-throughs of the Berkeley campus and was employed for the MIT City Scanning Project, which captured thousands of calibrated images from an instrumented rig to compute a 3D model of the MIT campus. While the resulting 3D models are often impressive, the system requires considerable time and effort from the user to decompose the scene into prismatic blocks and manually select features and their correspondence in different views, followed by the estimation of the pose of these primitives. Consequently, the system is impractical for reconstructing large scenes.

More recently, Xiao et al. [XFT+08] developed a semi-automatic image-based approach to reconstruct 3D façade models of high visual quality from a sequence of street view images. Their method employed a systematic and automatic decomposition scheme of façades for both analysis and reconstruction. The decomposition is achieved by a recursive subdivision that partitions the whole façades into small segments, while still preserving the architectural structure. Users are required to provide feedback on façade partition. This method demonstrated excellent results.

Brown et al. [BL05] presented an image-based modeling system that aims to recover camera parameters, pose estimates and sparse 3D scene geometry from a sequence of images. Snavely et al. [SSS06] introduced the Photo Tourism (Photosynth) system which is based on the work of Brown, with some significant modifications to improve scalability and robustness. Agarwala et al. [AAC$^+$06] proposed another related technique for composing panoramas of roughly planar scenes. Although these approaches address the same SfM concepts as we do, their aim is not to reconstruct and visualize 3D scenes and models from images, but only to allow easy navigation between images in three dimensions.

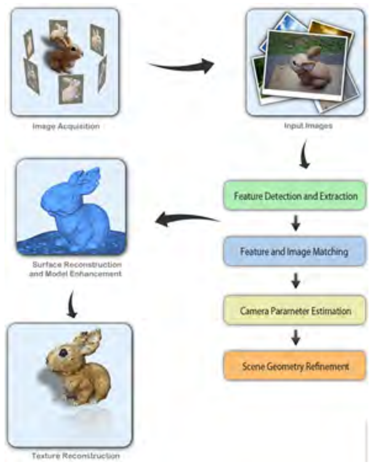


Image Acquisition
Input Images
Feature Detection and Extraction
Feature and Image Matching
Camera Parameter Estimation
Scene Geometry Refinement
Surface Reconstruction and Model Enhancement
Texture Reconstruction

Figure 1: Stages of the reconstruction process.

3 DESIGN OF 3D RECONSTRUCTION ALGORITHMS

Ideally, 3D reconstruction algorithms can be categorized depending on the visual cues employed to perform reconstruction, i.e. silhouettes, texture, transparency, defocus, shading or correspondence. The best-known and most successful visual cues have been shading, silhouettes, and correspondence [HVC08, Qua10]. Silhouettes and correspondence offer the highest degree of robustness due to their invariance to illumination changes. The shading cue requires more control over the illumination environment, but can produce excellent results.

In this section, we review and analyze the two most popular reconstruction techniques: silhouettes and correspondence based methods. In order to be able to reconstruct a 3D scene without any a prior knowledge, the intrinsic and extrinsic parameters of the camera being used must first be estimated. This process is further divided into three sub-steps: feature extraction, feature matching, and camera parameter estimation. 3D scene geometry can then be recovered by either back projecting and interpolating 3D points (correspondence-based), or using silhouette information (silhouette-based). Figure 1 depicts several stages of the reconstruction process.

3.1 Feature Detection and Extraction

The objectives of this step are to identify features of interest in each image and to match the features across views. The accuracy of the entire reconstruction process relies on the features of the scene that can be identified, extracted and automatically matched. Consequently, occlusions, illumination variation, limited locations for the image acquisition and reflective surfaces are problematic. However, recent invariant feature detector, such as SIFT [BL05], have proved to be fairly robust under large image variations. Feature points extracted by SIFT are highly distinctive and invariant to different transformations and changes in illumination, as well as having a high information content [BL05, HL07].

The SIFT operator works by first identifying potential points of interest. This is achieved by isolating points located at the extrema of the Difference-of-Gaussian (DoG) function in scale space. The location and scale of each key point is then computed and key points are selected based on measures of stability. Unstable extremum points (key points with low contrast or edge response features along an edge) are rejected as they are too sensitive to noise for accurate localization. Each detected key point is then assigned one or more consistent canonical orientations based on local image gradients. The key point descriptor is then described relative to this canonical orientation, thereby achieving invariance to rotation. Finally, using local image gradient information, a descriptor is produced for each key point [SSS06]. Figure 2 shows an example of detected key points from the Queen Victoria statue dataset (Auckland, New Zealand) before and after localization.

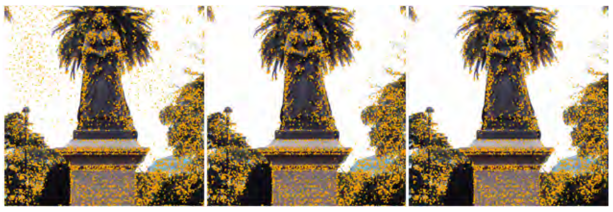


Figure 2: Left: Candidate key points detected from the first stage. Middle: After discarding low contrast key points. Right: After discarding key points located on edges (keypoints near the edges and corners of images are now removed).

3.2 Feature Matching

Once features have been identified and extracted from all the images, they are matched. This is known as the correspondence problem. Given a feature in an image I_1, what is the corresponding feature (the projection of the same 3D point) in the other image I_2? This can be solved by defining a distance function that compares the two feature descriptors. All the detected features in I_2 are tested and the one with the minimum distance is selected [CA84]. The Euclidean distance is employed to measure the similarity between two key points A and B.

A small distance indicates that the two key points are close and thus similar. However, a small distance does not necessarily mean that the points represent the same feature. For instance, a scene can contain many similar features such as corners of windows in an large building. It merely indicates that the two features have the highest resemblance of all processed features. In order to accurately match a key point in the candidate image, we identify the closest and second closet key points in the reference image using a nearest neighbor search strategy. If the ratio of them is below a given threshold, the key point and the closest matched key point are accepted as correspondences, otherwise that match is rejected [Low04, Low99].

Since multiple images may view the same point in the world, each image is matched to the nearest neighbors. During this process, image pairs whose number of corresponding features is below a certain threshold are removed. In our experiment, the threshold value of 20 seems to produce the best results.

As the matching procedure is subject to errors and mismatches, many of our matches are spurious. It is possible to eliminate many spurious matches by enforcing a geometric consistency. This is predicated on the fact that, assuming a stationary scene, not all corresponding features between two images are physically resizable, regardless of what the actual shape of the scene is. This geometric constraint is known as the epipolar constraint. The epipolar constraint requires that a pair of corresponding features, $(x_1, y_1) \rightarrow (x_2, y_2)$ between two images satisfies the equation:

$$\begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} \mathrm{F} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = 0 \qquad (1)$$

Where F denotes the *Fundamental matrix*, which defines a bilinear constraint between the coordinates of corresponding image points. Thus, for a given image pair, only matching features that agree with the epipolar constraint are admissible. All other matches are rejected.

3.3 Camera Parameter Estimation

Given a set of matching images, the goal of this stage is to recover the geometry of the scene and the motion information of the camera (camera parameters) simultaneously. The motion information includes the extrinsic (position, orientation) and intrinsic parameters of the camera for the captured images. This is accomplished using the Structure from Motion (SfM) technique [SSS06, Sze, COM$^+$11].

The reconstruction process begins by estimating parameters for an initial pair. The selection of the initial image pair to be reconstructed is highly critical. If the reconstruction of this initial pair gets stuck in undesirable local minima, the optimization is unlikely to ever converge. To avoid such cases, the initial pair must be selected carefully. The chosen images should have a large number of correspondences, but also have a relatively large baseline (the distance between camera optical centers). This is to ensure that the location of the 3D observed point is well-conditioned, so that the initial two-frame reconstruction can be robustly estimated.

The estimation of the extrinsic parameters for this initial pair is as follows [SSS06]: First, the Essential matrix is approximated using the five-point algorithm. Next, the projection matrix can be retrieved by decomposing the Essential matrix. Feature tracks visible in the two images are then triangulated, giving an initial set of 3D points. Once the structure of the scene and the motion information have been estimated for the first pair, they are further refined using Bundle Adjustment. Bundle Adjustment refines a visual reconstruction to produce the optimal 3D structure and motion information. This last step is critical for the accuracy of the reconstruction, as concentration of pairwise homographies would accumulate errors and disregard constraints between images. The recovered geometry parameters should be consistent. That is, the reprojection error, which is defined by the distance between the projections of each key point and its observations, is minimized (Figure 3).

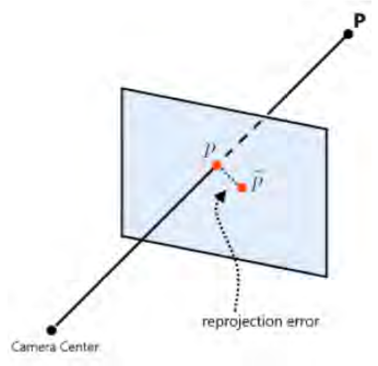


Figure 3: The reprojection error is the distance between the projected image point p and the observed image point $\widetilde{p}$.

Subsequent images are added to the optimization one at a time, with the best matching image being added at each step. Best matching images are those that share the largest number of tracks whose 3D locations have already been estimated. Each new added image is initialized with the same orientation, and focal length as the image that it matches best. This has proved to work very well even though images have different rotation and scale. Next, Bundle Adjustment is applied to refine the solution. This procedure is repeated, one image at a time, and terminates when no more images can reliably be added. The reliability test is determined based on the number of correspondences. A camera is only added when it shares a sufficient number of correspondences. In our system, we use a threshold value of 25, which was empirically selected from our experiments. Figure 4 demonstrates several stages of the SfM algorithm.

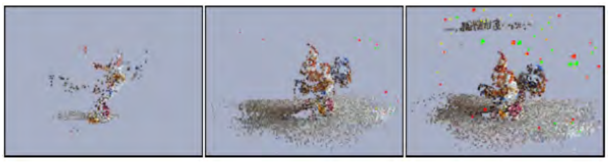


Figure 4: Several SfM stages of the reconstruction of our Rooster dataset. Left: the initial two-frame reconstruction. Middle: an intermediate stage after 20 images have been added. Right: the final construction with 42 images.

3.4 Correspondence- and Silhouette-based Reconstruction

Once the camera parameters have been successfully estimated, the 3D scene geometry can be computed. For correspondence-based approaches, the final steps involved triangulating correspondences to obtain a sparse set of 3D points. This initial sparse set of points then undergoes a refinement process which often involves denoising and resampling. Surfaces are then applied onto the point clouds to produce the final 3D model [ASG, SSS06, HL07, NWDL11, FP09].

For silhouette-based approaches, the 3D model is produced by exploiting silhouette information to create intersected visual cones, which are subsequently used to derive the 3D representation of an object. The construction of these cones requires the coordinates of a set of silhouette contour vertices for a given camera, and the coordinates of the camera's optical center as input. Rays extrapolated from the camera's optical center through the contour image points and beyond define a silhouette cone for that view, which is guaranteed to contain the original object. The intersection of silhouette cones from different viewpoints defines a polyhedral visual hull as an approximation of the object [HVC08, Lam02, FB10, LW10].

4 EVALUATION

In this section, we present an evaluation of four 3D reconstruction systems. These include two systems that seem to be correspondence-based (our own correspondence-based system and *Hyper3D*) and two reconstruction systems that seem to be silhouette-based (*Agisoft* and *123D Catch*). These systems were chosen because they are among the best with regard to the quality of reconstruction. Our goal is to evaluate their efficiency, accuracy, constraints and limitations in order to provide insight into the current state of image-based reconstruction in general.

In order to evaluate a system we used a repository of 40 objects. After an initial tests using different objects we selected one object which reflected the capabilities of all tested algorithms and used it to investigate the effect of image acquisition parameters. For this selected object, we created eight different datasets. Some of the input images are shown in Figure 5. The datasets vary in the number of input images they contain and the resolution of the input images.

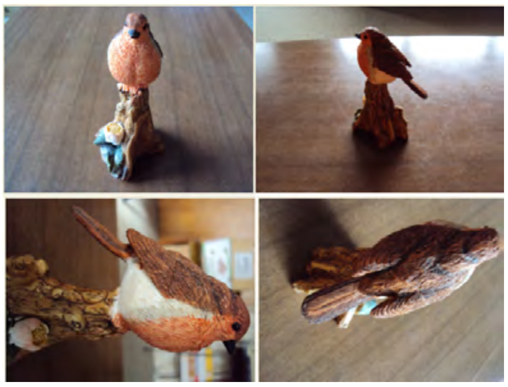


Figure 5: Four input images from four distinct viewpoints.

All images are taken with a *Logitec Webcam* in an indoor environment. Some input images are intentionally captured with other unrelated moving objects, to test how different algorithms handle unrelated moving object in the scene. The original model has a bumpy surface with a reasonable number of distinctive features.

4.1 Effect of the Number of Input Images

The objective of this test is to determine the effect of the number of input images on reconstruction quality. We evaluated 4 datasets of the bird model with a constant resolution of 1600 $\times$ 1200 and varying numbers of input images: 30, 20, 12 and 7. Figure 6 shows the reconstruction results for all the evaluated systems. The perspectives in which the reconstructed models are shown in the table were chosen so that reconstruction deficiencies are most visible.

30 Images For this dataset, our system, *123D Catch*, and *Agisoft* produce a qualitatively good model. There

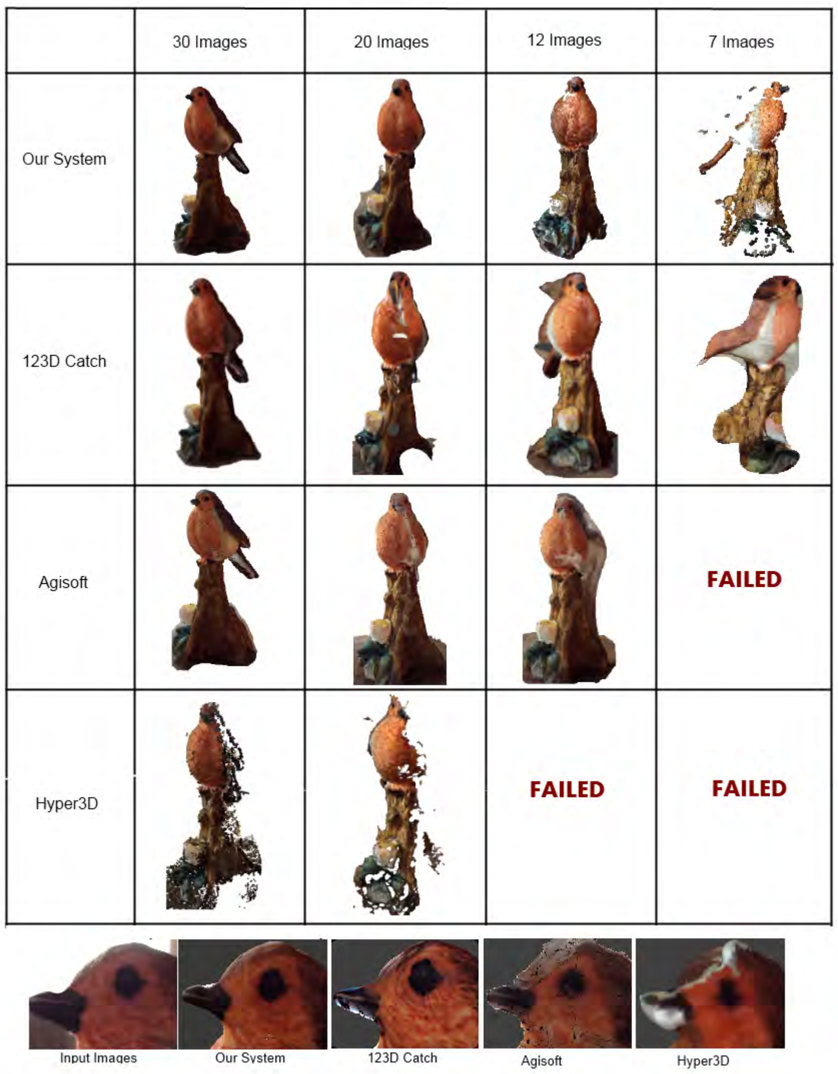


Figure 6: Reconstruction results of the first test suit. Last row: Close-up screen shots of the reconstructed models from 30 images.

is a slight deformation (on the side) in *123D Catch*'s model, but overall the resulting reconstruction is visual-pleasing. Our reconstruction has a more blurry texture than that generated using *123D Catch*. This is because we generate the texture by performing color interpolation between 3D points while they use a projection-based texture. However, in terms of geometry, our reconstruction bears the highest resemblance to the original model. *Hyper3D* was only able to construct a partial model.

20 Images *Agisoft*'s system produced a reasonably good model, although there is a slight disruption in the geometry in the chest of the bird model. *123D Catch*'s model is not as good. There is a large chunk of the background glued to the model. This is probably caused during the background subtraction process in which the object was not properly segmented. The model produced from our system has the best geometry, however the texture has become even more blurry. This is understandable as there are fewer distinctive features in the input image sequence leading to fewer 3D points generated. *Hyper3D*, again, was only able to produce a partial reconstruction.

12 Images For the third dataset, *Hyper3D* was unable to produce any result. *Agisoft* model's geometry was disfigured. There is a large bit missing on the side of the model. *123D Catch* has reasonably good geometry, although similar to the previous case there is a bit of the background attached to model (Figure 6). For our model, there is a small missing region at the top of the model. Apart from that, the geometry still retains the highest resemblance to the original model.

7 Images For this dataset, the reconstruction results from our system and *123D Catch* are shown in Figure

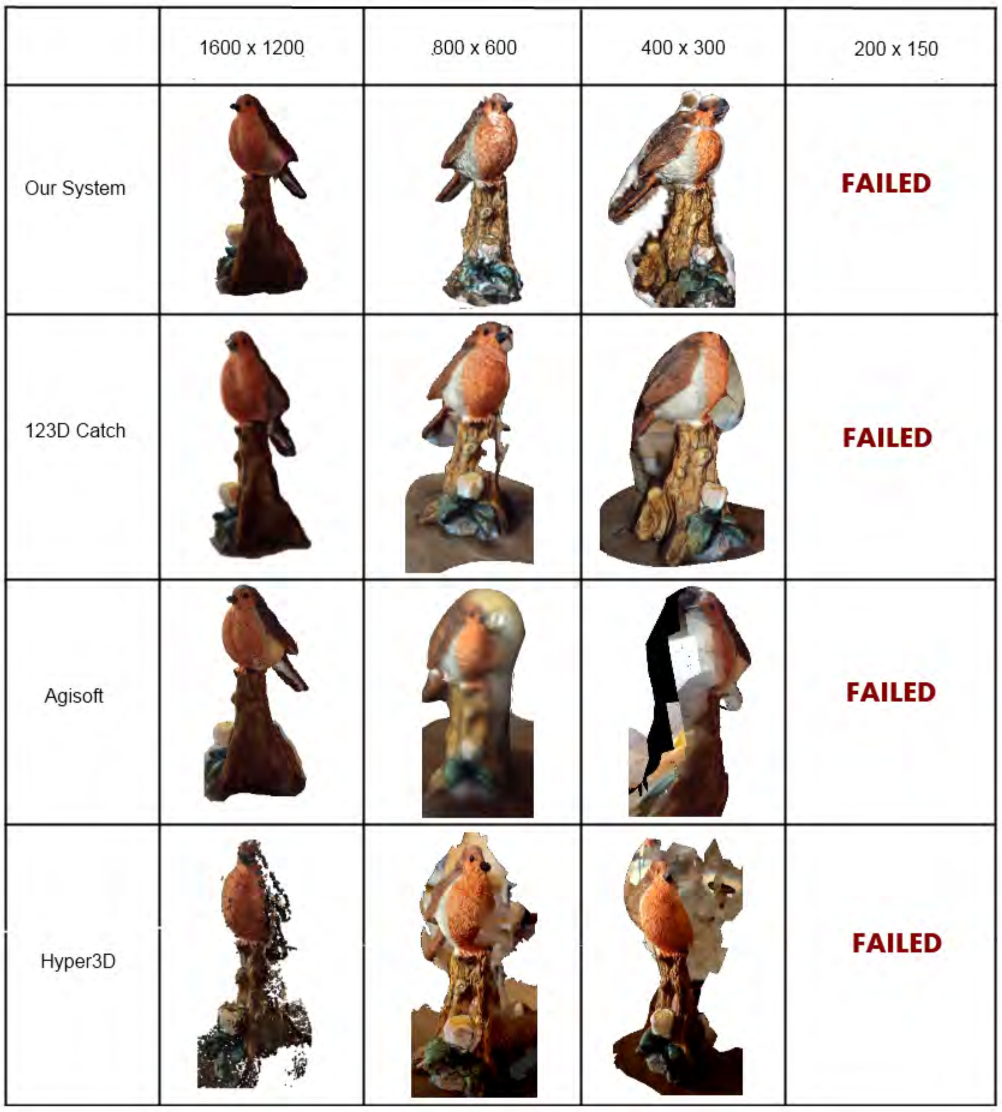


Figure 7: Reconstruction results of the second test suit.

6. *Agisoft* and *Hyper3D* system were unable to produce any result. Both *123D Catch* and our system were only able to construct a partial model. Although the geometry of *123D Catch* model seems more complete, it is almost unrecognisable and does not share much in common with the original model. In contrast, the model created using our system still has some resemblance to the original model. This test indicates that our system and *123D Catch* are amongst the most robust with regard to limited number of input images.

The result of this test clearly differentiate correspondence-based from silhouette-based approaches for a decreasing number of input images. Correspondence-based approaches, although producing models with good geometry, tend to have more missing geometry when the number of input images decreases. Additionally, textures generated from this approach are often blurry as the result of interpolation. Silhouette-based approaches do not create holes, but they show coarse and distorted geometry for small numbers of images. The resulting models become more refined with an increasing number of input images. To be able to construct a reasonable quality model of a small sized object, today's systems would need at least 20 input images from a standard consumer-level camera. Adding more than 30 images does not improve the reconstruction quality significantly. Models reconstructed from 12 images or less are usually unsatisfactory.

4.2 Effect of the Input Image Resolution

We aim to evaluate the performance of these systems with regard to image resolution. For this test suit, we reduce the resolution of the input images. There

	Our System	123D Catch	Agisoft	Hyper3D
Speed	Medium	Fast	Slow	Medium
Geometry	Good	Good	Good	Average
Texture	Average	Good	Good	Good
#Images	Small	Small	Medium	Medium
Resolution	Small	Small	High	High
Constraints	None	None	None	None
Min #images	12	12	12	20
Min resolution	400×300	400×300	800×600	1600×1200

Table 1: Summary of the four system's performance.

are four datasets in this test suit, each contains 30 input images with resolution of 1600 × 1200, 800 × 600, 400 × 300, and 200 × 150 respectively. The objective is to stress the systems further to determine how well each system performs in the case of low resolution input images. Figure 7 illustrates the resulting reconstructions from all the systems.

1600 × 1200 Due to the large resolution of the input images, most resulting models are well reconstructed. *Hyper3D* produces the worst model, which has a large missing region.

800 × 600 For this dataset, *123D Catch*'s system yields the most qualitatively accurate model which has its texture properly recreated, although there remain many missing regions in the final model. Our model has the most complete and well-reconstructed geometry of all resulting models. Our texture, however, is noisy. Models from *Agisoft* and *Hyper3D* are mostly disfigured. One half of the reconstructed models has completely vanished. This is mostly due to the fact that their systems are not able to register views when there only a limited number of features in each input images. In the case of *Agisoft*, the texture appears very blurry.

400 × 300 For this test, *Agisoft*'s model is completely unrecognizable. There is a large bit missing from the reconstructed model. Our system, *123D Catch* and *Hyper3D* were able to produce some outputs. Although the resulting reconstructions are only partial. This is also the result of insufficient number of distinctive features, which leads to failure to establish global image correspondence. Models from our system and *Hyper3D* are reasonably reconstructed. The resulting models still reflect the structure of the original object. In the case of *123D Catch*, the resulting model bears almost no resemblance to the original

200 × 150 In this test, all the systems were unable to register images due to insufficient overlap between image features.

The results show that silhouette-based approaches seem to be less robust for low resolutions. This is probably because silhouette-based methods are naturally deterministic and do not account for errors that might be present in views. The errors are typically caused by inaccurately estimated camera parameters. For silhouette-based systems, resolution significantly below 1600 × 1200 does not seem to yield satisfactory models anymore. Correspondence-based approaches are slightly more robust with regard to image resolution.

Models reconstructed with low-resolution images using correspondence-based approaches often have noisy surfaces, but appear more complete. However, for correspondence-based approaches, a resolution below 800 x 600 does not seem to produce satisfactory models anymore.

5 CONCLUSION

We described the overall design of image-based reconstruction algorithms, and evaluated a number of 3D reconstruction systems. The evaluation shows that there are general differences between the different algorithms, particularly between correspondence-based and silhouette-based algorithms.

Correspondence-based algorithms produce good details for larger numbers of input images ($\geq$20), but tend to produce missing geometry (holes) as the number of input images decreases. The textures they generate are often blurry because of interpolation. They are fairly robust with regard to image resolution and still produce models with fairly complete geometry for low resolutions, although the surfaces become noisy.

Silhouette-based approaches do not create holes, but they show coarse and distorted geometry for small numbers of images. They tend to produce better textures because they backproject the original images as the silhouettes are constructed. However, they tend to be less robust for low-resolution input images, as they are more sensitive to camera parameter estimation errors. A summary about various aspects of the four systems is shown in Table 1.

To gain a deeper understanding into today's reconstruction algorithms, it is necessary to investigate the effect of other parameters such as illumination, distortion, occlusion and object types.

6 REFERENCES

[AAC+06] Aseem Agarwala, Maneesh Agrawala, Michael Cohen, David Salesin, and Richard Szeliski. Photographing long scenes with multi-viewpoint panoramas. *In ACM Transactions on Graphics*, 25(3):853–861, 2006.

[ASG] Pierre Alliez, Laurent Saboret, and Gael Guennebaud. Surface reconstruction from point sets. Available at `http://www.cgal.org/Manual/3.5/doc_html/cgal_manual/Surface_reconstruction_points_3/Chapter_main.html`. Last accessed on May 22nd 2011.

[Bau74] Bruce Guenther Baumgart. *Geometric modeling for computer vision*. Doctoral Dissertation, Stanford University, 1974.

[BL05] Matthew Brown and David Lowe. Unsupervised 3D object recognition and reconstruction in unordered datasets. *In International Conference on 3D Digital Imaging and Modelling*, pages 56–63, 2005.

[CA84] Chang Ho Chien and J. K Aggarwal. A volume surface octree representation. *In Seventh International Conference on Pattern Recognition, Montreal, Canada*, pages 817–820, 1984.

[COM+11] Wei Cheng, Wei Tsang Ooi, Sebastien Mondet, Romulus Grigoras, and Geraldine Morin. Modeling progressive mesh streaming: Does data dependency matter. *ACM Transaction on Multimedia Computing*, pages 1–24, 2011.

[DTM96] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach. *In ACM Transactions on Graphics*, pages 11–20, 1996.

[FB10] Jean Sebastien Franco and Edmond Boyer. Efficient polyhedralmodeling from silhouettes. *IEEE Transaction on Pattern Analysis and Machine Intelligences*, 31:853–861, 2010.

[FLB06] Jean-Sebastien Franco, Marc Lapierre, and Edmond Boyer. Visual shapes of silhouette sets. *In 3D Data Processing, Visualization and Transmission*, pages 397–404, 2006.

[FP09] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multi-view stereopsis. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2009.

[GSD03] Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell. A bayesian approach to image-based visual hull reconstruction. *In IEEE International Conference on Computer Vision and Pattern Recognition*, 1:187–194, 2003.

[HL07] Shungang Hua and Ting Liu. Realistic 3D reconstruction from two uncalibrated views. *In International Journal of Computer Science and Network Security*, 7:178–183, 2007.

[HVC08] Carlos Hernandez, George Vogiatzis, and Roberto Cipolla. Multi-view photometric stereo. *In IEEE Transaction on Pattern Recognition and Machine Intelligence*, 30:548–554, 2008.

[Lam02] Bruce Lamond. *An Investigation into the Recovery of Three-Dimensional Structure from Two-Dimensional Images*. Master Thesis, School of Computer Science, University of Edinburgh, 2002.

[Low99] David G Lowe. Object recognition from local scale-invariant features. *In International Conference on Computer Vision*, 2:1150–1157, 1999.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *In International Journal of Computer Vision*, 60:91–110, 2004.

[LW10] Chen Liang and Kwan Yee Wong. 3d reconstruction using silhouettes from unordered viewpoints. *Image and Vision Computing*, 28(4):579–589, 2010.

[MBR+00] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven Gortler, and Leonard McMillan. Image-based visual hulls. *In Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374, 2000.

[mCBK05a] Kon man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette across time part 1: Theory and algorithms. *In International Journal of Computer Vision*, 62(1):221–247, 2005.

[mCBK05b] Kon man Cheung, Simon Baker, and

Takeo Kanade. Shape-from-silhouette across time part 2: Applications to human modeling and markerless motion tracking. *In International Journal of Computer Vision*, 63(1):225–245, 2005.

[NWDL11] Hoang Minh Nguyen, Burkhard Wunsche, Patrice Delmas, and Christof Lutteroth. Realistic 3d scene reconstruction from unconstrained and uncalibrated images. *In Proceedings of GRAPP 2011, Algarve, Portugal*, 31:67–75, 2011.

[Qua10] Long Quan. *Image-Based Modeling*. Springer Press, 2010.

[SSS06] Noah Snavely, Steven Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3D. *In ACM Transactions on Graphics*, 25(3):835–846, 2006.

[Sze] Richard Szeliski. Image alignment and stitching. A tutorial in Computer Graphics and Vision, 2006. Available at `http://research.microsoft.com/apps/pubs/default.aspx?id=70092`. Last accessed on May 21st 2011.

[XFT+08] Jianxiong Xiao, Tian Fang, Ping Tan, Peng Zhao, Eyal Ofek, and Long Quan. Image-based façade modeling. *In ACM Transactions on Graphics*, 27(5):26–34, 2008.

Investigating the Rate-Distortion Performance of a Wavelet-Based Mesh Compression Algorithm by Perceptual and Geometric Distortion Metrics

Maja Krivokuća[1] Burkhard Wuensche[2] Waleed Abdulla[3] Guillaume Lavoué[4]

[1,2,3] The University of Auckland, New Zealand
[4] Université de Lyon, France

[1] mkri012@aucklanduni.ac.nz [2] b.wuensche@auckland.ac.nz [3] w.abdulla@auckland.ac.nz
[4] glavoue@liris.cnrs.fr

ABSTRACT

The rate-distortion performance of wavelet-based mesh compression algorithms is usually only evaluated in a purely geometric sense, by measures such as the Root Mean Square Error and the Hausdorff distance, which do not capture the human visual perception of distortion. This lack of quantitative information about the perceptual effects of wavelet compression has prompted us to present a more complete evaluation of a classic wavelet-based mesh compression algorithm, by measuring its rate-distortion performance both with geometric metrics (the Hausdorff distance and Root Mean Square Error) and perceptual metrics (the recently introduced Mesh Structural Distortion Measure 2 (MSDM2) and the Mean Opinion Scores (MOS) that we obtained by conducting a subjective experiment with human observers), where the rate is measured as the percentage of wavelet coefficients used in reconstruction. The MSDM2 has already been proven to outperform other existing perceptual metrics for several different types of distortions, but this is the first time that it has been tested for this type of geometric distortion in a real-use case scenario. We found that, in this context, the MSDM2 generally correlates well with the MOS but seems to under-estimate the perceptual error in cases of low-frequency (large scale) shape distortion. Due to the disparities in the distortion values produced by the tested distortion metrics, we also conclude that a complete evaluation of any mesh compression algorithm should include several different distortion metrics, to allow the developers and users of these compression algorithms to make more informed decisions about the applicability of those algorithms in different application areas.

Keywords
Distortion metrics, quality metrics, wavelets, mesh compression, rate-distortion evaluation.

1. INTRODUCTION

Since the influential work of Lounsbery [Lou94], who introduced the notion of multiresolution analysis on surfaces, wavelet-based mesh compression has been a topic of much interest in the research community. This is reflected by the variety of wavelet-based mesh compression algorithms proposed in the literature, notably [HP05, KG02, KSS00, SS95, VP04]. The main idea in wavelet-based mesh compression is to decompose a high-resolution input mesh into a coarse representation called a *base mesh* and a set of detail coefficients termed *wavelet coefficients,* which can be used to refine the base mesh at multiple levels of detail [Lou94]. Geometry compression may then be obtained either by discarding small (unimportant) wavelet coefficients at each resolution level, and/or by quantizing and entropy coding the remaining coefficients that are to be transmitted. The aim in wavelet-based mesh compression is to optimise the trade-off between data size and approximation accuracy, which is measured by the *rate-distortion* (*R-D*) curve. The *rate* refers to the amount of information transmitted in a compressed mesh, and the *distortion* refers to a quantified measure of difference between the reconstructed and original meshes. While the rate is a relatively straightforward measure to quantify (usually represented as the number of bits transmitted per vertex, or the number of wavelet coefficients transmitted), the term *distortion* still lacks a formal definition. Without such a definition, it is difficult to claim that any lossy mesh compression algorithm developed to date (including wavelet-based algorithms) has been fully evaluated. While wavelet-based mesh compression has shown some very promising results and is still an active area of research, the performance of existing algorithms has usually only been reported based on a single error metric. In addition, the distortion metrics that are currently used to evaluate these algorithms are normally purely geometric measures, such as the Root Mean Square Error and the Hausdorff Distance, which are not designed to capture the *visual* disparity between two 3D models. Due to the previous lack of availability of an objective perceptual distortion metric, the visual distortion caused by discarding wavelet coefficients in a wavelet-based mesh compression system has not yet been documented in a quantitative manner. However, this information is important for many graphics applications where the ultimate judge of the transmitted model is a human. The surge in recent years to design perceptually-based distortion metrics (a survey and extensive comparison can be found in [LC10]) has produced some promising perceptual metrics, particularly the *Mesh Structural Distortion Measure 2* (*MSDM2*) recently introduced by Lavoué [Lav11]. While the *MSDM2* has been proven to outperform other existing perceptual metrics for several different types of classical distortions [LC10, Lav11], it has not been tested in a real-use case and has not been investigated for the type of geometric distortions that result from discarding wavelet coefficients in a wavelet-based mesh compression system.

The objective of this paper, therefore, is twofold: (1) to evaluate how the *MSDM2* metric compares to human perception of distortion, in the real-use case of wavelet-based mesh compression, where the distortion is caused by discarding different percentages of wavelet coefficients in mesh reconstruction; and (2) to offer a more complete evaluation of the effects of discarding wavelet coefficients in a wavelet-based mesh compression system. The latter is achieved by measuring the rate-distortion performance of a classic wavelet mesh compression algorithm with several different error metrics - two commonly used geometric measures (the Hausdorff distance (d_H) and the Root Mean Square Error (*RMSE*)) and two perceptual metrics (the subjective Mean Opinion Score (*MOS*) from a group of human observers, and the objective visual distortion metric, *MSDM2*). In this way, we also aim to demonstrate that, due to the disparities that exist between the performance results generated by these distortion metrics, a complete evaluation of a lossy mesh compression algorithm is not possible with only one distortion metric. We use our investigation of the wavelet compression technique as a case study to suggest how an appropriate error metric may be chosen for evaluating a lossy mesh compression algorithm based on different application needs.

Section 2 of this paper introduces the wavelet-based compression method that we have implemented and describes our evaluation procedure, Section 3 discusses the basic concepts behind the distortion metrics that we have investigated, Section 4 describes the perceptual distortion test that we carried out, Section 5 presents our results and discusses the insights gained in relation to our objectives, and the conclusion ties up the key messages of this paper.

2. WAVELET COMPRESSION CASE STUDY

The progressive compression algorithm that we have investigated is an own implementation of the fundamental Lounsbery *subdivision wavelet* method [Lou94]. We only cover here the basic concepts of this method that are necessary for an understanding of our investigation. For further details, we refer the interested reader to the original thesis [Lou94].

2.1 Background and Implementation

The subdivision wavelet method applies the notion of multiresolution analysis to subdivision surfaces. The basic idea is that we can take a high-resolution input mesh with subdivision connectivity and decompose it into a lower-resolution mesh, together with a set of detail coefficients that can be added to the lower-resolution mesh to reconstruct the higher-resolution mesh. This decomposition, or *analysis*, is done by two separate filtering operations on the original mesh: a low-pass filtering where we compute weighted averages of the vertices in the higher-resolution mesh to obtain the (sparser) set of vertices in the resulting lower-resolution mesh (we call this lower-resolution mesh an *approximation* of the higher-resolution mesh that it was obtained from), and a high-pass filtering where we compute weighted differences of the higher-resolution mesh vertices to obtain the detail coefficients, called *wavelet coefficients*. If we continue these filtering operations on each successive, lower-resolution mesh, we eventually obtain the coarsest possible mesh (called the *base mesh*), together with wavelet coefficients at multiple levels of detail. If we add back the wavelet coefficients to the corresponding mesh at each level, we can progressively refine the base mesh and thereby obtain a multiresolution representation of the original mesh. *Figure 1* illustrates a simple example of this process, where the mesh on the left is the input mesh, the mesh on the right is the base mesh, and the middle mesh is the model at an intermediate resolution level.

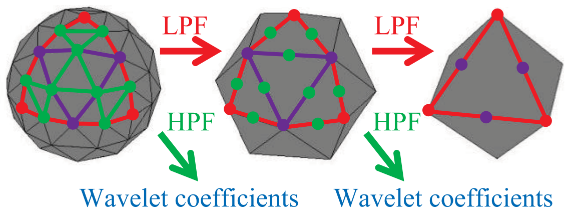


Figure 1: Illustration of wavelet decomposition. LPF=low-pass filter, HPF=high-pass filter. The reconstruction process proceeds in reverse. The purple and green circles represent vertices added at the midpoints of edges in the refinement process.

The reconstruction, or *synthesis*, process involves two more filtering operations: a refining operation where each triangular face of a lower-resolution mesh is subdivided into four sub-triangles by introducing new vertices at edge midpoints, and a perturbing operation where the new vertices are perturbed (displaced) to their new positions according to the wavelet coefficients. *Figure 1* highlights the refinement process of one face of the base mesh (outlined in red), by showing how the new vertices (purple) are added to the midpoints of that face and are connected together in the next higher resolution level (the middle mesh), in order to refine the one base mesh face into four smaller sub-faces. Similarly, in the middle mesh in *Figure 1*, the next set of vertices (green) is added at the midpoints of the edges of the new faces, and these vertices are connected together to produce 16 sub-faces in the next higher resolution level (leftmost mesh). Each new set of vertices is, in this case, projected onto the surface of a sphere (perturbing operation), which is how we get from an octahedron base mesh with flat faces (rightmost mesh in *Figure 1*) to an n-sided figure (leftmost mesh in *Figure 1*) that begins to approximate the shape of a sphere. Since the connectivity of the mesh at each resolution level is obtained by a common refinement process, we only need to transmit the base mesh and the set of wavelet coefficients in order to reconstruct the original mesh geometry. Due to the orthogonality property of the wavelets (see [Lou94] for an in-depth explanation), the mesh approximation at each resolution level is guaranteed to be the approximation at that level that is the closest to the next, finer-level mesh in a least-squares sense. The orthogonality property also means that the wavelet coefficients are a good indicator of where the approximation is not close to the finer-level mesh that it is approximating. For example, if a wavelet coefficient is zero, this means that the approximation is locally perfect as it indicates that there is no information (detail) missing in that region. If the wavelet coefficient is large, however, then it indicates that there is a large amount of information missing at that position and scale. Since the approximation at each resolution level is guaranteed to be the best possible approximation for that level, however, most of the wavelet coefficients are usually distributed around zero (not much information missing), so we can discard many of the (small) coefficients (thereby achieving compression) and still be able to reconstruct a close approximation of the original mesh.

In our implementation, we order the coefficients by magnitude and then select the largest wavelet coefficients at each level. The percentage of chosen coefficients at each level constitutes our *rate* of transmission. The wavelet coefficients are deliberately not quantized or entropy coded, in order to isolate the distortion effects (and compression) that are caused solely by eliminating wavelet coefficients. Our implementation also incorporates a *naïve* reconstruction, where every location on the mesh is refined in a uniform manner (every face is split into 4 sub-faces), regardless of whether or not additional detail is ever added to that location. This means that the reconstructed mesh always has the same number of triangles as the input mesh, but the quality of geometry reconstruction (the accuracy of the final vertex positions) depends on the number of wavelet coefficients that we use in the reconstruction.

2.2 Experimental Procedure

We tested the *rate-distortion* performance of the subdivision wavelet compression method on six different meshes with subdivision connectivity - shown in *Figure 2*, along with their associated base meshes. The base meshes are presented in faceted form, to demonstrate their different levels of complexity.

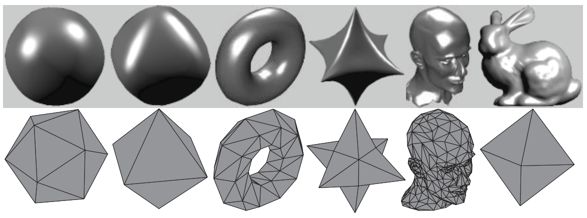


Figure 2: The subdivision models used for testing, and their associated base meshes (underneath each model). From left: Sphere (res. level 4), Rounded Octahedron (res. level 4), Torus (res. level 3), Star (res. level 4), Mannequin (res. level 3), and Bunny (res. level 6).

2.2.1 Choice and Preparation of Test Models

The models were chosen based on the amount of complexity, in terms of visual detail, that they possessed. We wished to investigate a range of models, from those that have very smooth surfaces (e.g., the Sphere) to those with quite a large amount of visual detail on the surface (e.g., Mannequin and Bunny). We also chose the models so that they would possess different *types* of details (e.g., smooth corners in the Rounded Octahedron, sharp points in the Star, sharp curves on the Bunny's ears and thighs) because we wished to see how the wavelet method would handle these different features as measured by the different distortion metrics. The associated base mesh for each model was chosen as the simplest (lowest-resolution) version of that mesh available in the model library. Each base mesh is therefore considered resolution level 1 in our experiments, and the resolution level for the corresponding input model is always considered relative to its base mesh. The resolution levels for each of the input models, relative to their base meshes, are stated in the caption of *Figure 2*.

2.2.2 Rate-Distortion Testing

Each of these test meshes was first decomposed (similar to *Figure 1*) to obtain its corresponding base mesh and wavelet coefficients. Then the reconstruction process was carried out at 11 different *rates*, where the *rate* is defined as the percentage of largest wavelet coefficients used at each resolution level, from 0% to 100% coefficients inclusive, in 10% steps. The mesh *distortion* at each of these rates was then measured separately with four different error metrics, which are discussed below.

3. STUDIED DISTORTION METRICS

We selected two traditional geometric error metrics - the Hausdorff distance (d_H) and the Root Mean Square Error (*RMSE*) - and two perceptual error metrics - the *MSDM2* metric and the subjective Mean Opinion Scores (*MOS*) obtained through a subjective experiment. The basic concepts behind each metric are discussed below.

3.1 Hausdorff Distance (d_H)

The Hausdorff distance represents the maximum distance between a point on one mesh and the surface of another mesh. It is formulated as follows:

The distance, $e(p,S)$, between a point p in 3D space and a mesh surface S, is defined as:

$$e(p,S) = \min_{p' \in S} d(p,p')$$

where $d(p,p')$ is the Euclidean distance between points p and p', and p' is a point on surface S. Then the one-sided (asymmetric) distance between two surfaces (or meshes), S_1 and S_2, is defined as:

$$E(S_1,S_2) = \max_{p \in S_1} e(p,S_2).$$

A two-sided (symmetric) distance, which is termed the Hausdorff distance, d_H, is defined as the maximum of $E(S_1,S_2)$ and $E(S_2,S_1)$:

$$d_H(S_1,S_2) = \max\{E(S_1,S_2), E(S_2,S_1)\}.$$

We used the *Metro* tool [CRS98] with the default settings, to compute the symmetric Hausdorff distance in all our experiments. The obtained d_H values were then normalized to fit into the range [0,1], to enable a comparison between the rate-distortion trends obtained from the different error metrics. The normalized Hausdorff distance, $d_H N^i$ for distorted model *i,* was computed as:

$$d_H N^i = \frac{(Current\ d_H{}^i - Minimum\ d_H{}^M)}{(Maximum\ d_H{}^M - Minimum\ d_H{}^M)}$$

Where *Current* $d_H{}^i$ is the current (un-normalized) d_H value for mesh *i*, and *Minimum* $d_H{}^M$ and *Maximum* $d_H{}^M$ are, respectively, the smallest and largest Hausdorff values produced for any of the distorted versions of mesh *M*.

3.2 Root Mean Square Error (RMSE)

The Root Mean Square Error *(RMSE)* measures how far, on average, the difference between the original and reconstructed vertex positions is from 0. The *RMSE* between two meshes, S_1 and S_2, is computed as:

$$RMSE(S_1,S_2) = \sqrt{\frac{\sum_{i=1}^{n}(v_i^{S_1} - v_i^{S_2})^2}{n}},$$

where n is the number of vertices in the meshes (both meshes must have the same number of vertices), and $v_i^{S_2}$ is the vertex in mesh S_2 corresponding to vertex $v_i^{S_1}$ in mesh S_1.

The *RMSE* was also normalized in our experiments to fit into the range [0,1], in a similar way to the Hausdorff normalization.

3.3 Mesh Structural Distortion Measure 2 (MSDM2)

The Mesh Structural Distortion Measure 2 (*MSDM2*), descendant of the earlier Mesh Structural Distortion Measure (*MSDM*) [LGDBE06], was recently introduced by Lavoué [Lav11] as a multiscale metric for objective visual quality assessment of a 3D mesh. The *MSDM2* is based on the 2D image metric, *SSIM* (Structural SIMilarity index), from Zhou et al. [ZBSS04], and works by measuring the differences in curvature statistics between two meshes, which are computed on local corresponding spherical neighbourhoods. These neighbourhoods vary in size according to the *scale*, *h*, which is related to the maximum length of the bounding box of the model. For example, for 3 scales, $h_i \in \{2\varepsilon, 3\varepsilon, 4\varepsilon\}$, where $\varepsilon = 0.5\%$ of the maximum length of the bounding box.

The symmetric *MSDM2* measure between a reference mesh, M_r, and a distorted mesh, M_d, is computed as the average of the two asymmetric *global multiscale distortion* (*GMD*) measures, $GMD_{M_d \to M_r}$ and $GMD_{M_r \to M_d}$, where $GMD_{M_d \to M_r}$ is defined as:

$$GMD_{M_d \to M_r} = \left(\frac{1}{|M_d|} \sum_{v \in M_d} MLD(v)^3 \right)^{1/3}.$$

$MLD(v)$ is the *multiscale local distortion* measure defined as:

$$MLD(v) = \frac{\sum_{i=1}^{n} LD^{h_i}(v)}{n}$$

which is the average of *local distortion* (*LD*) values at single scales. The *LD* at a given scale *h* is defined for each vertex *v* from M_d as:

$$LD^h(v) = \frac{\alpha L^h(v) + \beta C^h(v) + \gamma S^h(v)}{\alpha + \beta + \gamma}$$

where α, β and γ are set to 1, 1 and 0.5, respectively, and $L^h(v)$, $C^h(v)$ and $S^h(v)$ are, respectively, the 3D mesh equivalents of the luminance comparison function, the contrast comparison function, and the structure comparison function defined for images in [ZBSS04]. Lavoué defines these functions for meshes in [Lav11].

In our experiments, we obtained the symmetric *MSDM2* values and the associated distortion maps from the MEPP platform [LTD12], using 3 scales. These values are in the range [0,1], where 0 indicates that the reconstructed mesh is identical to the original, and values closer to 1 correspond to increasingly larger visual differences between the two meshes.

3.4 Mean Opinion Score *(*MOS*)*

The Mean Opinion Score (*MOS*) is a subjective measure of distortion, where a group of human observers is asked to give a score to some distorted objects, which reflects the observer's degree of perceived distortion on these objects, in relation to an original, undistorted object. The *MOS* for a distorted model, *i*, is computed as:

$$MOS_i = \frac{1}{n}\sum_{j=1}^{n} m_{ij}$$

where MOS_i is the mean opinion score of the i^{th} distorted model, *n* is the number of test observers, and m_{ij} is the distortion score given by the j^{th} observer to the i^{th} model.

The next section describes the subjective experiment that we conducted to obtain these *MOS* values.

4. SUBJECTIVE EXPERIMENT

Due to a lack of information in the literature about the perceptual effects of discarding different percentages of wavelet coefficients, and from a desire to evaluate the *MSDM2* in this context, we carried out our own perceptual distortion test on three of our test models. The details of this test are explained below.

4.1 Assessment Procedure

A group of 13 human observers were shown 3 different 3D models (the Torus, Star and Bunny – see *Figure 2*), where each original model was printed on paper together with its 11 distorted versions (reconstructions with 0%-100% wavelet coefficients), and the printouts were given to each individual observer. The original model was labelled, but the 11 distorted versions were arranged in random order around the original. The observers were asked to give a distortion score between 0 and 10 to each distorted model (using whole numbers only), which would reflect the degree of perceived distortion on this model in relation to the original. Participants were told that a score of 0 meant that, in their opinion, the distorted model was identical to the original (i.e., they could not perceive any distortion on this model) and a score of 10 was the worst case scenario (i.e., the distorted model was "very different" to the original). Participants were *not* told to assign a 10 to the worst model and then assign lower scores to all the other models; in fact, the observers were told that if they felt that no model "deserved" a 10, for example, they did not need to use the full range of distortion scores. The observers were also told to consider the distorted models together and give them relative scores. They were further asked, for all the models to which they gave a score greater than zero, to circle the area(s) on those models that they thought were the "worst distorted" areas. There was no specific time limit for the test, but the task took approximately 20 minutes for the whole group.

4.1.1 Choice and Preparation of Test Models

The Torus, Star and Bunny were chosen for the subjective experiment because these models have a range of interesting surface details: the smooth curves on the Torus, the sharp points on the Star, and the curves of varying degrees of sharpness on the Bunny. We wished to investigate how the *MSDM2* compares with the *MOS* for capturing such different types of surface detail. The viewing angle for each model (same viewing angle as in *Figure 2*) was chosen so as to portray that model in what we subjectively judged to be both its most discriminative angle and the angle that offered the most familiar viewpoint of the object. The lighting and surface reflectance conditions were also chosen specifically for each model, based on the selected viewpoint, so as to produce what we believed was the best visibility of the shape and surface detail of each model, for the test observers. For all three models we used interpolated shading and *Phong* face lighting, but the strengths of diffusion, ambient lighting and specular highlights were chosen individually for each model so that, for the chosen viewpoint, no surface details (or, as little as possible) would be hidden by the specular highlights or shadows. The printed models were made to be of similar size, so that 6 models could fit on one side of an A4 sheet of paper in landscape orientation, organized side by side in two rows and three columns. While this use of a fixed viewpoint and fixed viewing distance for the test models does limit the generality of the obtained results, this step was an attempt at reducing the amount of variance between the test subjects. If the observers were free to zoom in and out of the models and rotate them, it would have been difficult to ensure that all observers saw the same parts of each model; perhaps some people would have used more viewpoints than other people, or a wider range of viewpoints, to make their decision. Having a fixed viewing angle and distance ensured that this type of variation between the subjects was eliminated and so, since everyone was looking at exactly the same images, this made it easier to isolate the features of each model which were responsible for the different distortion scores given. This was ultimately the goal of the subjective experiment: to determine how the *MOS* compares to the *MSDM2* for different models with different types of features, distorted in the same way.

4.2 Normalizing the Distortion Scores

Before computing the *MOS* for each distorted model, the individual distortion scores were normalized to be in the range [0,1]. This was simply done by dividing each score by 10. Since the observers were instructed to only use the entire available scoring range (0-10) if they actually saw the need for it (i.e., if they could see enough difference between the different levels of distortion to require them to use the entire scoring range), they did not necessarily have to assign a 10 to the worst model and a 0 to the best model. Rather than scaling all the scores to fit into the 0-10 range, we were interested in the differences in the actual range of values that would be assigned for each model (a discussion of this is provided with *Table 1*, opposite). For this reason, it was not appropriate in our experiment to correct for the differences in gain and offset among the observers (as was done in [LC10], for example). The suitability of our experimental protocol was assessed by computing three Intraclass Correlation Coefficients (ICC), each of which measures the variation between the different observers in their subjective ratings of all the distorted versions of one of three test models (Torus, Star, and Bunny), respectively. The ICCs were computed from a two-way ANOVA test with no repetitions, and using the ICC equation related to *Model 2* in [SF79]. The ICC value for the Torus models was computed as 0.81, for the Star models as 0.79, and for the Bunny models as 0.94. Since ICC values close to 0 indicate poor agreement while values close to 1 indicate almost perfect agreement, the computed values show that there was strong agreement between the observers on the distortion scores they assigned to the Torus and Star models, and almost perfect agreement on the distortion values assigned to the Bunny models. These high levels of agreement indicate that our subjective experiment protocol was correct as it produced meaningful, consistent scores from the test observers.

5. RESULTS AND ANALYSIS

The rate-distortion curves for all six models were plotted and are displayed in *Figure* 3.

5.1 Variability between the Observers

While the ICC values reported above suggest strong agreement between the overall subjective distortion scores given to all three test models, looking at the distortion scores for the models at each rate of wavelet coefficients individually leads to some interesting observations. In particular, for the Torus model, only 4 out of 13 observers used the full distortion scoring range (0-10), for the Star 5 out of 13 used the full range, and for the Bunny 9 out of 13 used the full range. *Table 1* shows the distortion scores provided by the 13 observers for the Torus, Star and Bunny, at a rate of 0% wavelet coefficients, which is where the highest distortion scores were given.

Observer	Distortion score for **Torus**	Distortion score for **Star**	Distortion score for **Bunny**
1	6	7	10
2	10	10	10
3	10	10	10
4	8	7	10
5	7	8	10
6	8	8	10
7	8	9	9
8	5	8	10
9	7	6	10
10	10	10	10
11	10	10	10
12	10	10	10
13	9	10	10

Table 1: Subjective distortion scores for the Torus, Star and Bunny models reconstructed with 0% wavelet coefficients.

At a rate of 0% wavelet coefficients, the mesh reconstruction just produces the base mesh *shape* (but with the same number of triangles as the input mesh, due to the naïve reconstruction). Since the Bunny's base mesh is perceptually much more different to the input Bunny (see *Figure 2*) than the Torus and Star base meshes are to their respective originals, this explains why for a rate of 0%, the Bunny received the highest possible distortion score (10) from 12 out of 13 observers, whereas the Torus at this rate received a 10 from only 5 observers and the Star from 6 observers. This observation indicates that the choice of base mesh is a critical factor in the perceptual quality of the reconstructed mesh, especially at low rates of wavelet coefficients. Furthermore, the table above shows that the range of distortion scores for the Torus model at a rate of 0% coefficients is the largest (10-5=5), the Bunny the smallest (10-9=1), and the Star in between the two (10-6=4). A similar relationship holds for other low rates – from 10% to around 40% wavelet coefficients – but this is not shown in *Table 1*. The wider ranges for the Torus and Star indicate that the observers found it more difficult to decide on appropriate distortion scores for these models, especially at the lower rates of wavelet coefficients. The greater agreement between the scores for the Bunny in general (indicated by both the smaller range and the higher ICC value) implies that the different levels of distortion were perceptually more obvious on the Bunny than on the Torus or Star. In fact, when the 13 observers were asked which model they found it the easiest and hardest to judge different levels of distortions on, the Bunny was almost unanimously selected (by 12/13 observers) as the easiest, and the Torus and Star were both said to be equally difficult. The observers felt that this was because the Bunny is quite a familiar, natural object, and so it was easy to tell when the model looked 'wrong'.

5.2 Comparison of MOS and MSDM2

We were able to make several important comparisons between the *MOS* and *MSDM2* distortion results for each tested model.

5.2.1 Bunny

The most significant difference between the *MOS* and the *MSDM2* R-D curves for all three tested models is that the *MSDM2* has a more stable, almost linear rate of decrease with increasing percentages of wavelet coefficients, whereas the *MOS* curves are more irregular. The most obvious example of this is in the Bunny R-D plots, where the *MOS* curve indicates a sharp drop in perceived distortion between 40% and 50% wavelet coefficients, but the *MSDM2* curve does not reflect this as a large change. Indeed, the *MSDM2* curve seems to decrease at a nearly constant (slower) rate, from 0% right up to around 70% wavelet coefficients. If we compare the Bunny reconstructions with 40% and 50% wavelet coefficients (see *Figure 4*) and compare these to the original Bunny model (see *Figure 2*), we notice that the Bunny at 50% looks much more similar to the original model than the Bunny at 40% does. Perceptually, the worst distortion in the 40% model (as judged by the majority of the test observers) is the area circled in *Figure 4*, and this distortion is not present in the 50% model. The reason why the *MSDM2* does not perceive this as a large difference between the two models may be because the *MSDM2* is designed to capture differences in curvature, but the circled area in the 40% Bunny and the corresponding area in the 50% Bunny have almost the same curvatures (i.e., the Bunny 'bulges out' in nearly the same places). The perceptual difference, as reflected by the *MOS*, is purely geometric – the circled area bulges out much further in the 40% model than in the 50% model, which makes the Bunny look out of proportion and unnatural, whereas the 50% Bunny looks much closer to what we might imagine a bunny to look like and it is much closer, visually, to the original Bunny model.

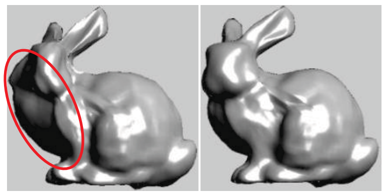


Figure 4: Bunny reconstructed with 40% wavelet coefficients (left) and 50% coefficients (right).

Interestingly, the opposite result occurs for the Bunny at 50%-60% wavelet coefficients: the *MOS* curve in this range changes very little (it is nearly flat), while the *MSDM2* has a sharper rate of decrease. Looking at *Figure 5,* which shows the reconstructed Bunny models with 50% and 60% wavelet coefficients, we can see that the most noticeable difference between them is the area circled in red (almost all observers circled this area as looking the most distorted). The *MSDM2* indicates this as a large distortion (see the area circled in red in the *MSDM2* distortion map in *Figure 5*, where "warmer colours" indicate higher distortion [Lav11]). This is because this 'bump' consists of rather a sharp curve compared to the smooth corresponding area in the 60% model (notice the corresponding area in the *MSDM2* distortion map for the 60% model, which indicates a considerably smaller error). However, the human observers did not perceive the removal of this bump in the 60% model as a very significant improvement, presumably because this was only a small 'glitch' on an otherwise good-looking bunny.

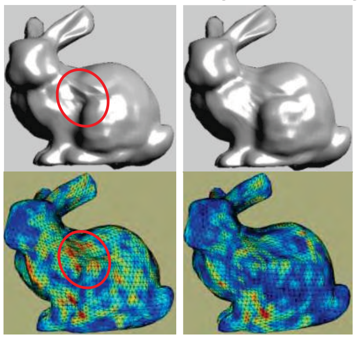


Figure 5: Bunny reconstructed with 50% wavelet coefficients (top left) and 60% wavelet coefficients (top right). Corresponding *MSDM2* distortion maps are beneath each model.

5.2.2 Star

In the case of the Star model, the *MOS* curve seems to follow the *MSDM2* curve more closely than in the Bunny's case. A good reason for this might be that the perceptual distortions in the Star are mainly due to the changes in curvature, which is what the *MSDM2* is designed to capture. For example, comparing the Star models reconstructed with 10%, 20% and 50% wavelet coefficients (see *Figure 6*), it appears that the perceived distortions on all these models are due to the differences in the sharpness of the Star's points and in the concavity of the Star's

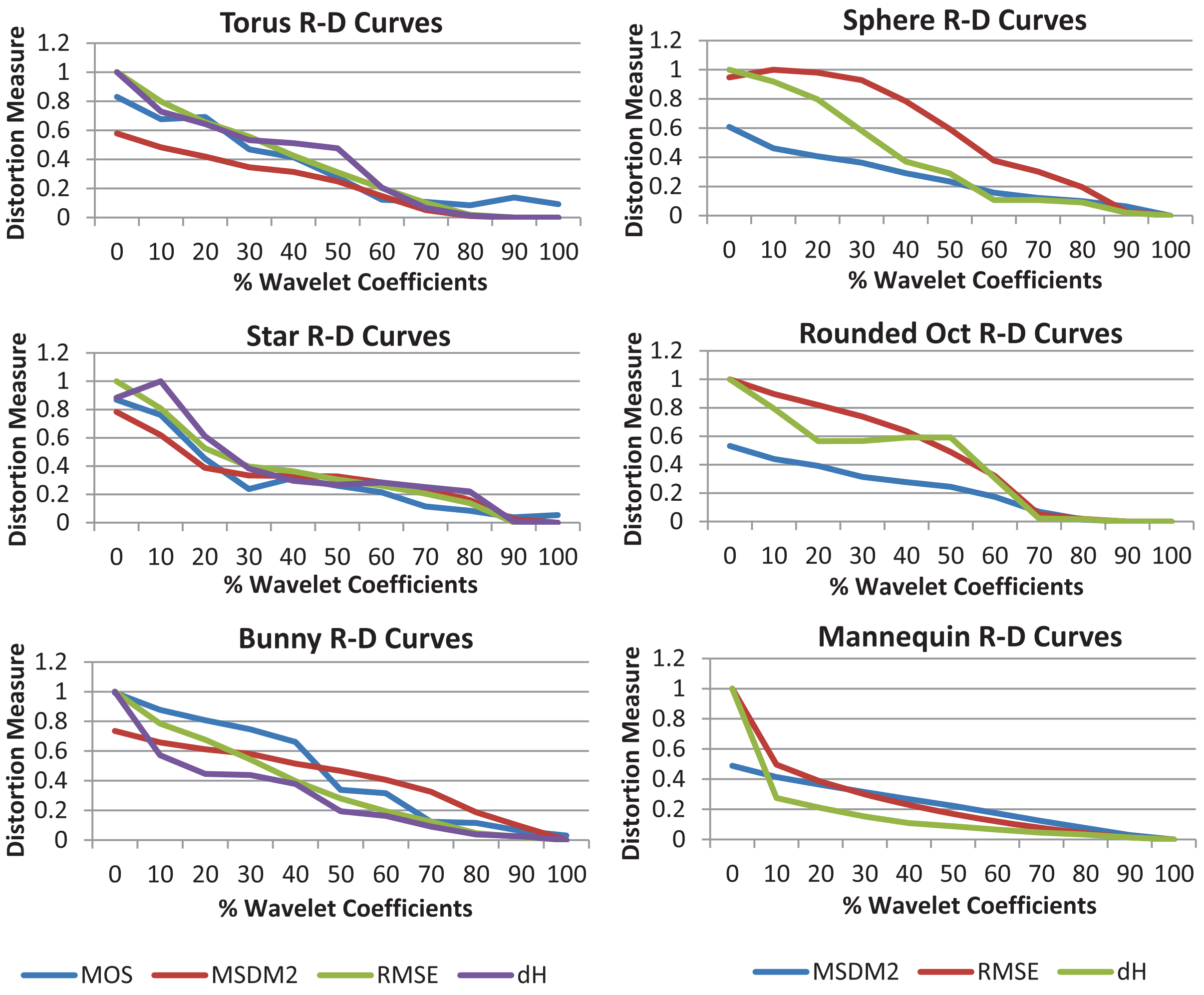


Figure 3: (Left) Rate-distortion curves for the 3 models used in the subjective test, and (Right) Rate-distortion curves for the remaining 3 models.

surface (circled on the 20% model, below, as an indication). These are the areas that the 13 observers circled as having the worst (most noticeable) distortion on most of the Star models. The corresponding *MSDM2* distortion maps indicate that the worst *MSDM2* distortions are in the same regions.

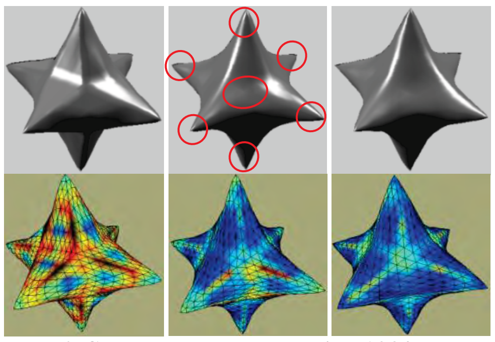


Figure 6: Star reconstructed with 10% wavelet coefficients (top left), 20% coefficients (top middle), and 50% coefficients (top right). Corresponding *MSDM2* distortion maps are beneath each model. Red circles indicate areas of worst distortion, as judged by the majority of the test observers.

5.2.3 Torus

The Torus model seems to have the largest disparity out of the three models tested, between the *MOS* and *MSDM2* rate-distortion curves, especially at the lower rates. A reason for this might be that, because the perceptual distortions in the Torus usually manifested themselves as small 'bumps' on the surface (for example, see the Torus reconstructed with 30% wavelet coefficients, in *Figure 7*), the *MSDM2* does not perceive these as very large differences in curvature to the original model, as these distortions are, in fact, not very sharp curves. This can be seen in the *MSDM2* distortion map in *Figure 7*, where the areas of the Torus with the worst perceived distortion (circled in red) do correspond to the areas of highest distortion in the *MSDM2* colour map, but these colours are mostly still in the lower (blue-green) range of distortion values, which, compared with the bright reds in the distortion map for the 10% Star model in *Figure 6,* are not very high. However, because the human eye expects a smooth, uniform shape, any small 'bumps' on the surface are readily visible and annoying as they interrupt the smooth flow of the surface. This observation was confirmed in the perceptual distortion test, where people seemed so aware of the little bumps that they even perceived the Torus models reconstructed with 100% wavelet coefficients as being different to the original (the two models are, in fact, geometrically identical). Only 5 observers out of 13 gave the Torus model at 100% a distortion score of 0, compared to the 10/13 zeros which were given to the 100% Star model and the 9/13 zeros which were given to the 100% Bunny model.

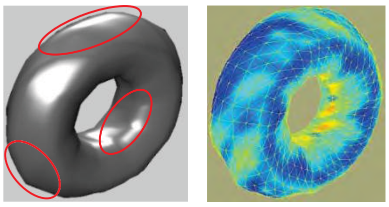


Figure 7: Torus reconstructed with 30% wavelet coefficients (left) and the corresponding *MSDM2* distortion map (right). Red circles indicate areas of worst distortion on this model, as judged by the majority of the test observers.

5.2.4 Overall Performance of MSDM2 vs MOS

Even though there are disparities between the *MOS* and *MSDM2* rate-distortion curves, the locations of various distortions on our test models and the general levels of distortion have been captured well by the *MSDM2*, compared to the human perception of these distortions. Evidence of this can be seen in *Figures 5, 6* and *7*, where the human-circled areas of "worst" distortion correspond to the highest error values on the *MSDM2* distortion maps, and in the R-D curves in *Figure 3*, where the *MSDM2* values, like the *MOS* values, gradually decrease as the percentage of wavelet coefficients increases.

In general, the largest differences between the *MOS* and *MSDM2* rate-distortion curves for all the three test models in our experiments seemed to occur at the lower rates (smaller percentages of wavelet coefficients). Because the largest wavelet coefficients are responsible for reconstructing the overall, global shape of a model, and the smaller wavelet coefficients are used to reconstruct the details, these results seem to suggest that the human eye is more critical of large-scale distortions (which affect the overall shape of a model) than the *MSDM2* predicts. On a low-detail model like the Torus, the human eye notices differences in local distortions more easily. This is to be expected, as this detail is not masked on a smooth surface whereas it might be masked on a highly-detailed surface. This last point agrees with the observations of the *visual masking effect* demonstrated by Lavoué [Lav11].

The disparities between the *MOS* and *MSDM2* for different models are an indication that human observers use more visual cues than just curvature differences to perceive visual distortion. For the type of distortion that we investigated, the removal of low- or medium-frequency wavelet coefficients produces large-scale localized distortions that have a high impact on the visual quality (see the left Bunny in *Figure 4*). However, the *MSDM2* metric fails to detect them; it underestimates these degradations, especially on very smooth, low-detail models like the Torus.

5.3 Using a Combination of Distortion Metrics for R-D Performance Evaluation

The performance of a wavelet-based mesh compression algorithm has not previously been reported with a combination of geometric and perceptual error metrics, and our investigation resulted in four main observations:

1. Both the geometric and perceptual distortion values decrease with increasing percentages of wavelet coefficients. This confirms that both the geometric and visual quality of a mesh improve with a greater number of wavelet coefficients.
2. The d_H and *RMSE* curves are quite closely correlated with the *MOS* curves for the Torus and Star models, but for the Bunny model the d_H drops more rapidly than the *MOS* at low rates. This suggests that large-scale distortions on detailed models like the Bunny affect perceptual quality more than geometric quality.
3. Across all six test models, the d_H curves generally seem less stable than the *RMSE* curves. This shows that the quality of maximum error induced by the surface reconstruction (measured by the d_H) is not always proportional to the average quality of vertex reconstruction (measured by the *RMSE*) as the number of wavelet coefficients increases.
4. The R-D performance of a wavelet-based mesh compression algorithm, and the suitability of a distortion metric to measure this performance, is dependent on the 3D model used as input.

More generally, we are able to conclude that, due to the disparities in distortion measurements produced by the different error metrics, it is not sufficient to evaluate the rate-distortion performance of a lossy mesh compression algorithm, such as the wavelet method, with a single error metric. The use of multiple distortion metrics for evaluation would benefit both the developers and users of mesh compression algorithms. The developers would benefit as it would be easier to accurately compare the performance of different algorithms, and the users would benefit because it would be easier to select the right compression method based on their application needs. For example, users that are only concerned with the *look* of the reconstructed model may choose a compression method with the best

MSDM2 or *MOS* performance, users that desire a close surface reconstruction (geometrically) within a certain tolerance regarding the original surface may be interested in the Hausdorff performance, while users that require the mesh vertices to be reconstructed exactly may consider the *RMSE* values. Our evaluation of the classic wavelet mesh compression method with several different error metrics aims to provide the first small step in this direction.

6. CONCLUSION

We evaluated the rate-distortion performance of the Lounsbery wavelet mesh compression scheme by using four different distortion metrics – the Hausdorff distance (d_H), the Root Mean Square Error (*RMSE*), the Mesh Structural Distortion Measure 2 (*MSDM2*), and the Mean Opinion Scores (*MOS*) obtained through a subjective experiment. We used the *MOS* to evaluate how well the *MSDM2* metric compares to the human perception of distortion caused by discarding different numbers of wavelet coefficients in the mesh reconstruction. The *MSDM2* has been found to correlate well with the *MOS* in terms of capturing the locations and general levels of these distortions, but has been shown to under-evaluate the perceptual effects of low-frequency (large-scale) shape distortions, especially on very smooth, low-detail models. We have further shown, through our evaluation of the wavelet compression method with different error metrics, that there exist disparities between the performance results produced by the existing distortion metrics, and for this reason it is important to measure and report the performance of a (lossy) mesh compression algorithm with several different distortion metrics. This would make it easier for developers and users of these compression algorithms to make more informed decisions about the applicability of those compression algorithms in different application areas and for different types of 3D models.

Promising future work in perceptual distortion metrics for wavelet-based mesh compression might include an investigation of the visual optimization tools used in JPEG 2000 [ZDL02], to determine whether some of the perceptual models used there to steer 2D image compression might be useful for 3D mesh compression.

7. ACKNOWLEDGEMENTS

The authors would like to thank Michael Lounsbery for the correspondence relating to his subdivision wavelet method, Gabriel Peyré and AIM@SHAPE for providing the subdivision models, and the anonymous reviewers whose comments helped us to improve the quality of this paper.

8. REFERENCES

[CRS98] Cignoni, P., Rocchini, C., Scopigno, R.: Metro: Measuring Error on Simplified Surfaces. In *Computer Graphics Forum* (1998). 17(2): pp. 167-174.

[HP05] Hoppe, H., Praun, E.: Shape Compression using Spherical Geometry Images. In *Advances in Multiresolution for Geometric Modelling*, N.A. Dodgson, M.S. Floater, and M.A. Sabin, Editors. 2005, Springer Berlin Heidelberg. pp. 27-46.

[KG02] Khodakovsky, A., Guskov, I.: Compression of Normal Meshes. In *Geometric Modeling for Scientific Visualization*, G. Brunnett et al., Editors. 2002, Springer Verlag. pp. 189–206.

[KSS00] Khodakovsky, A., Schröder, P., Sweldens, W.: Progressive geometry compression. In *Proc. 27th International Conference on Computer Graphics and Interactive Techniques*, New Orleans, LA, USA, July 2000, pp. 271-278, ACM/Addison-Wesley.

[Lav11] Lavoué, G.: A Multiscale Metric for 3D Mesh Visual Quality Assessment. In *Computer Graphics Forum* (2011). 30(5): pp. 1427-1437.

[LC10] Lavoué, G., Corsini, M.: A Comparison of Perceptually-Based Metrics for Objective Evaluation of Geometry Processing. In *Multimedia, IEEE Transactions on* (2010). 12(7): pp. 636-649.

[LGDBE06] Lavoué, G., Gelasca, E.D., Dupont, F., Baskurt, A., Ebrahimi, T.: Perceptually driven 3D distance metrics with application to watermarking. In *Proc. SPIE Applications of Digital Image Processing XXIX*, San Diego, CA, USA, August 2006, vol. 6312, pp. 63120L.1–63120L.12.

[Lou94] Lounsbery, J.M.: Multiresolution analysis for surfaces of arbitrary topological type. PhD Dissertation. Dept. Comput. Sci. and Engineering, University of Washington, Washington, USA, 1994.

[LTD12] Lavoué, G., Tola, M., Dupont, F.: MEPP - 3D MEsh Processing Platform. International Conference on Computer Graphics Theory and Applications, Rome, Italy, February 2012.

[SF79] Shrout, P.E., Fleiss, J.L.: Intraclass correlations: uses in assessing rater reliability. In *Psychological Bulletin* (1979). 86(2): pp. 420-428.

[SS95] Schröder, P., Sweldens, W.: Spherical wavelets: efficiently representing functions on the sphere. In *Proc. 22nd Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA, August 1995, pp. 161-172, ACM.

[VP04] Valette, S., Prost, R.: Wavelet-based Progressive Compression Scheme for Triangle Meshes: Wavemesh. In *Visualization and Computer Graphics, IEEE Transactions on* (2004). 10(2): pp. 123-129.

[ZBSS04] Zhou, W., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. In *Image Processing, IEEE Transactions on* (2004). 13(4): pp. 600-612.

[ZDL02] Zeng, W., Daly, S., Lei, S.: An overview of the visual optimization tools in JPEG 2000. In *Signal Processing: Image Communication* (2002). 17(1): pp. 85-104.

EmotiCon
Interactive emotion control for virtual characters

Bernhard Bittorf
CogVis/MMC, Faculty of Media,
Bauhaus-University Weimar
Bauhausstrasse 11
99423 Weimar, Germany
bernhard.bittorf@uni-weimar.de

Charles Wuethrich
CogVis/MMC, Faculty of Media,
Bauhaus-University Weimar
Bauhausstrasse 11
99423 Weimar, Germany
charles.wuethrich@uni-weimar.de

ABSTRACT

Emotional Expressions are an integral component of the reliability when animating virtual characters. But the sheer amount of possible emotions and the complexity in their selection makes it difficult to develop an intuitive way of controlling arbitrary facial expressions interactively in real-time. This work aims at finding a decent representation of emotions and their expressions, as well as their interactive control by suitable tools. It develops a valid cultural spreading system to classify emotions and thus control emotional expressions interactively. Different complexity stages are presented and evaluated for being able to satisfy different application scenarios.

Keywords
Facial Expressions, Emotion Control, Facial Animation, Virtual Humans, Knowledge-based Animation, Motion Synthesis, Emotion and Personality

1 INTRODUCTION

Realistic facial animation is one of the most difficult tasks for computer animators. The face is the main instrument for communication and defining a person's character. Animating human-like faces is typically done either by using keyframes or through Motion Capturing. Keyframing descends from 2D hand-drawn animation techniques. The usage of Motion Capturing and the evolution of modern graphics cards have led to the possibility of animating human-like faces in real-time. The sheer amount of possible motions requires new methods for controlling these animations in real-time. There are some approaches on defining scripting-languages or parameterized facial models but the field lacks an intuitive way of controlling arbitrary facial expressions interactively in real-time.

The main goal of this research is to find a reasonable model for controlling the state of a character rather than animating the face itself. We introduce a method for interactively controlling the emotions (and thereby the emotional expressions) of virtual characters in real-time which is modelled on the real-life interaction between director and actor. In order to achieve this, we have analyzed existing classifications of emotions and deduced methods and prototypes to control emotional expressions, interactively. Finally an experimental evaluation was performed.

2 RELATED WORK

Controlling human-like faces means basically controlling atomic movements. Such movements depend on the geometric model. The idea behind this work is to decrease the amount of possible movements or actions needed to be done by an animator by providing an abstraction layer which employs different interdependencies and metaphors.

The AMA-System [TMPT88] uses "Abstract Muscle Actions" to simulate muscle movements with single vertices that form the face. Those atomic expressions form so called tracks, which are a chronological sequence of keyframes. Tracks can be subsequently mixed like sound tracks in a recording studio.

Prem Kalra developed the SMILE-System at the University of Geneve. Different abstraction layers are defined to separate muscles, "Minimum Perceptible Actions" (MPAs), phonems and expressions as well as words and emotions from each other [KMMTT91, KMTM+98]. MPAs contain informations about the frame number where to start, the minimal action to be animated and the intensity to which the minimal action shall evolve. The System provides a High Level Script Scheduler (HLSS)

which allows the user to build emotions and words out of those MPAs and arrange the whole animation [MTM00]. The general syntax would be:
`while` *<duration>* `do` *<action>*.

Different approaches have been made to define such a scripting language which enables the animator to act as a director. But in this case a scripting language is not an interface for intuitive interactive work because the process of animating needs flexibility and responsiveness. However, the idea of creating an abstraction layer which enables the user to control a large number of parameters intuitively seems quite convenient though.

The MPEG-4 standard [Koe02] includes a set of Facial Definition Parameters (FDPs) describing the face and 68 associated Facial Animation Parameters (FAPs) which control rigid rotation of the head, eyeballs, eyelids and mandible. The standard also defines parameters that indicate the translation of corresponding feature points. FAPs are also used to describe the most common facial expressions and the visemes. In 2009 Rodrigues et al. developed a dynamic emotion model to facial expression generation using the MPEG-4 standard [RSV09] which was designed to work for the autmatic synthesis but not for interactive, intuitive control.

The Facial Action Coding System [EF78] is a widely used standard to systematically describe all observable facial actions that are independent from each other. Ekman and Friesen found 46 so called Actions Units (AUs).

The Facial Expression Repertoire [BW10] is a set of atomic face movements which is based on the Facial Action Coding System but extends it with asymmetric movement. The main contribution is the mapping of Basic Emotion to a list of Action Units.

In 2004 Helzle et al. [HBSL04] developed a system for mapping captured Motion Curves onto the vertices of a humanlike head. For each existing Action Unit there are 100 captured timesteps that describe the movement from a neutral position to the Action Unit at its extreme value. The Adaptable Facial Setup (AFS) is a a standardized facial rig, driven by a set of nonlinear deformations. In 2007 Schmidt developed a system for animating all possible Action Units via the Adaptable Facial Setup in realtime. [Sch07]

Motivation

Finding a reasonable model for controlling the emotional state of a character and therewith his emotional expressions interactively requires a reasonable classification of emotions. The sheer amount of possible atomic movements (46 in FACS, 68 in MPEG-4, 120 in AFS) - not to mention their intensities - yields the need for an abstraction layer. By controlling the emotions of a virtual character we provide the user with an intuitive abstraction. To satisfy different needs we used different models and developed different interfaces for working with them respectively.

The Adaptable Facial Setup works with 120 Facial Actions multiplied by 68 Influence Points which results in 8160 datasets (Motion Curves) with 100 timesteps each and provides us with the opportunity to render humanlike characters in real-time, and thus explore and evaluate different interfaces and interaction techniques.

3 EMOTICON

The idea of manipulating emotions instead of directly manipulating the face itself provides the user with an abstraction such that an intuitive real-time interaction is possible. But there are two open questions: How can the user control the emotional state of the figure in a way that as many emotional expressions as possible can be achieved? And how are these emotions mapped to emotional expressions or Facial Actions?

Classification

Basically there are two approaches to classify emotions. The *Dimensional Classification* was developed in 1896 by Wundt [Rei00] and describes an emotion by a set of coordinates in a n-dimensional space. In 1986 Russell developed a circumplex model [Rus80, Rus86, Rus02, Rus03], in which all possible emotions can be defined in terms of a pair of xy-coordinates on a plane described by the axes *arousal* and *pleasure* (see Figure 1). The origin marks a neutral state and an emotion results from mixing different states of arousal and pleasure.

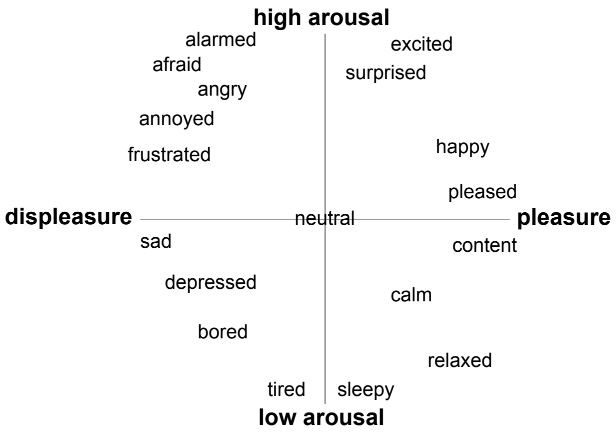


Figure 1: Classification according to James A. Russell

The *Categorical Classification* proposes some basic atomic emotions which can form more complex ones when they are mixed together. Different authors state different sets of Basic Emotions. But it can be shown that the field agrees in at least eight Basic Emotions. Table 1 shows the usage of the 14 most considered emotions in this scientific field whereas "‘+"’ indicates the usage and "‘-"’ indicates the rejection of the emotion respectively.

Emotion	McDOUGALL	PLUTCHIK	EKMAN
Fear	+	+	+
Anger	+	+	+
Disgust	+	+	+
Joy	+	+	+
Sadness	+	+	+
Interest	-	+	+
Surprise	+	+	+
Contentment	-	+	+
Compliance	+	-	-
Tenderness	+	-	-
Contempt	-	-	+
Amusement	-	-	+
Pride	-	-	+
Relief	-	-	+

Table 1: Different Basic Emotions according to different authors

We state that there is a more or less consensus in at least eight Basic Emotions: *Fear*, *Anger*, *Disgust*, *Joy*, *Sadness*, *Interest*, *Surprise* and *Trust*. Robert Plutchik used particularly these eight Basic Emotions for his classification of emotions [Plu62, Plu80]. Figure 2 shows a top view of Plutchiks model. The eight Basic Emotions are represented by eight petals which are subdivided into three different intensity levels respectively. *Surprise* for example increases to amazement and decreases to distraction. Related emotions are adjacent and polar ones are opposed. Mixing different Basic Emotions leads, depending on how far they are away from each other, to primary (*Trust* + *Joy* = *Love*), secondary (*Anger* + *Joy* = *Proudness*) or tertiary dyads (*Fear* + *Disgust* = *Shame*).

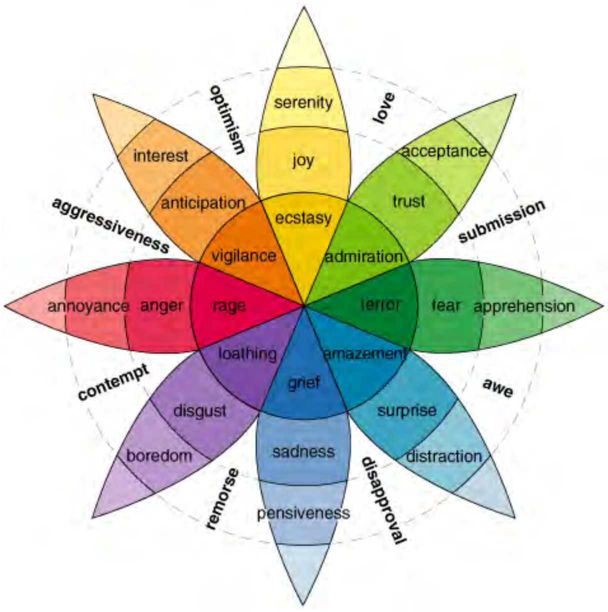


Figure 2: Classification according to Robert Plutchik

Interaction modes

On the basis of the *Categorical* and the *Dimensional Classification* of emotions and their inherent models we developed two interaction models. Because animating humanlike faces can be done in different ways and with different contexts we decided to implement different interaction modes based on different models.

Direct Control

Using Plutchiks classification of emotions as an interaction model allows the user to specify exactly which emotion the virtual character experiences at the moment. Mixing the eight Basic Emotions creates all other possible emotional states and therefore emotional expressions. We developed three interfaces for using this model.

There are basically two ways of controlling the eight Basic Emotions: either one chooses a graphical representation of the emotional space which provides the user with an overview and works as an interface or the input parameters are mapped onto a physical analogon (prop). Both methods were implemented.

The simplest way to interact with Plutchiks Classification is to use the flower-like graph as seen in Figure 2 as a GUI which can be controlled by the mouse. This leads to a very direct interaction but allows the user only to create primary dyads (see section Classification) and thus not all possible emotions. The reduction of possible emotions and the fact that related emotions are adjacent in the GUI leads to a reduction of the cognitive load for the user. The interface allows the animator to define vaguely the mood and its intensity by clicking a petal. The intensity increases to the outer sections of the petal. Thus the midpoint of the flower defines a neutral face which allows the user to fade between different (especially polar) emotions without any visual break.

Another approach is to map the Basic Emotions onto a physical analogon. We decided to implement a more artistic and a more technical prototype. We used one octave of a standard MIDI-keyboard as interface and mapped the eight Basic Emotions onto the eight white keys (as shown Figure 3). Keyboard sensitivity determines the intensity of the particular emotion. Extreme pressure on the keys thus means intenser emotions. This prototype allows the user to mix up to five different emotions per hand - each of them with one finger. But it is not possible to lock a certain emotion and its intensity for improving the other emotions at will because there is no way to lock the keys in a certain position. Furthermore refining the intensity values lacks accuracy.

By trying to combine the best properties of the two interfaces we used a sound mixer as a prop and mapped the eight Basic Emotions onto the eight sliders (see Figure 4). Each slider ranges from 0 to 255 whereas zero means a neutral state and 255 is the most emotional state. This allows the user to mix all eight Basic Emotions independently. At the same time the intensities can be locked and adjusted in a very accurate way.

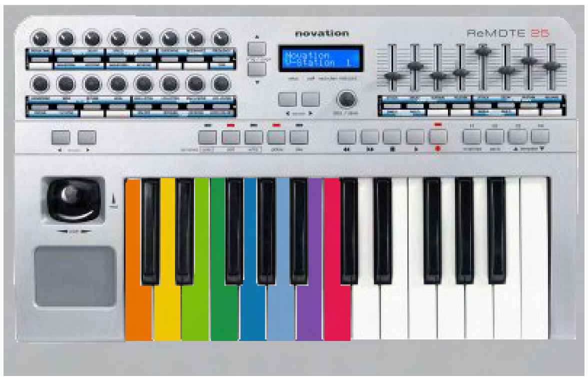


Figure 3: Analogon 1 - MIDI-keyboard

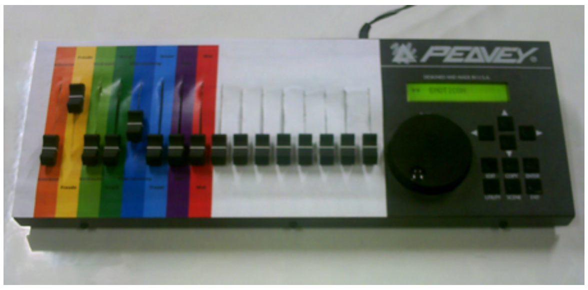


Figure 4: Analogon 2 - mixer

The Direct Control allows an arbitrary mix of all Basic Emotions whereas each single emotion can be controlled almost continuously. Although this model seems to be very intuitive the user has to be very clear about what he is doing which results in an high cognitive load.

Indirect Control

Using Russel's circumplex model as an interaction paradigm allows the user to specify vaguely which emotion the avatar experiences at the moment. Defining the states of arousal and pleasure without having to decide which exact emotion to choose leads to a reduction of cognitive load allowing to perform other tasks simultaneously.

According to the interfaces used for the Direct Control we used the graphical representation of the emotional space as a GUI and provided a prop for navigating.

When using the visualisation as a GUI which can be controlled by the mouse the user can pick a certain emotion and an emotional expression respectively by clicking on the corresponding point within the coordinate system. By holding down the mouse button one can navigate through the emotions which are mixed implicitly. Theoretically the systems enables the user to pick every possible emotion separately. Technically the system mixes the resulting emotions by averaging and normalizing the amplitudes of the individual Action Curves.

As a second approach we used the Spacemouse as a prop. This allows the user to navigate in six degrees of freedom whereas the device is mounted elastically and therefore always returns to a defined rest position automatically. This provides a haptical feedback about the position in the plane. Obviously we only use two degrees of freedom for navigating on the plane and controlling the emotion. This means that the character always starts in a neutral state. For accessing a state of high arousal the character has to go through all in-between states continuously. This leads to a very smooth flow of emotional expressions.

Constraining the input parameters to arousal and pleasure leads to a low cognitive load for the user. On the other hand it can be complicated to access a desired emotion directly and the system appears to be hardly predictable. The restriction on two parameters predestinates the concept of Indirect Control for being used not only for manual synthesis. It can also be used as basis for automatic synthesis of emotional expressions as it could be used for autonomous agents.

4 EVALUATION

All the proposed models and interfaces for controlling the emotional expressions of humanlike figures have their individual strengths and weaknesses. For a decent validation we performed an experimental evaluation. We wanted to know:

- *Which method can be used most intuitively?*
- *Are the interfaces useable?*
- *Do they behave according to the expectations of the user?*
- *Is the method easy to learn?*
- *Which is the most enjoyable to use?*

The experiment was designed as follows:
We asked 12 participants to control a virtual character interactively while a story was read to them. Afterwards they were asked to answer three qualitative paper-and-pencil questionnaires: a general one (age, expertise) and one for each control method. The respondent was presented with a continuous scale between 0 and 100.

- *Is the prototype usable?*
- *Is the usage intuitive?*
- *Is the prototype easy to learn?*
- *How much did the facial animation correlate to what you expected to see?*

The participants were between 20 and 50 years old and their expertise ranged from none to being professional puppeteers. Table 2 shows the results of the Direct Control. The percentage is a degree of satisfaction (0

Criteria	Mouse	Keyboard	Mixer
Ease of use	73%	56%	89%
Intuitivity	75%	74%	92%
Learning curve	84%	66%	92%
Expectations	67%	39%	90%
Overall score	66%	45%	93%

Table 2: Summary of evaluation results for Direct Control

means very unsatisfied and 100 means very satisfied). The variances were rather small (between 10 and 20)

Table 2 shows that using the Mixer as a prop works best for most of the participants (93% Overall). The mixer can be used right away is very intuitive and invites to play (92% Intuitivity). The reason why the keyboard performs so poorly (45% Overall) is that most users had strong difficulties using the keyboard sensitivity. So they were not able to gain the desired emotional intensities. Especially puppeteers had really fun with Direct Control and the mixer. All of them were performing the expected emotional expressions by themselves.

Criteria	Mouse	Spacemouse
Ease of use	60%	67%
Intuitivity	47%	50%
Learning curve	64%	70%
Expectations	54%	54%
Overall score	57%	69%

Table 3: Summary of evaluation results for Indirect Control

Table 3 shows that the Indirect Control paradigm was hard to understand for the users. This was due to two reasons. First, the model uses two abstraction layers. The users had to listen to the story, think about the resulting emotions (first abstraction) and map them to the axes arousal and pleasure (second abstraction). Second, the Spacemouse driver turned out to cause serious trouble. The fact that the Spacemouse performs significantly better than the mouse interface in spite of the driver problems shows that the Spacemouse seems to be a reasonable device for Indirect Control.

All in all the Direct Control performs much better and is more intuitive. The Indirect Control is hard to predict and needs a lot of training. The interviews after the test and the above results show that controlling the eight Basic Emotions with a Mixer is the most intuitive and most satisfying way to interactively control the emotions (and thereby the emotional expressions) of virtual characters in real-time.

5 LIMITATIONS

At the moment the system supports only emotional expressions - neither phonems or signs nor head and eye movement have been implemented.

6 CONCLUSIONS AND FUTURE WORK

We analyzed existing classifications of emotions and deduced reasonable classifications of emotions and their expressions for an intuitive usage and interactive control by suitable tools. We developed a valid cultural spreading system to classify emotions and thus control emotional expressions interactively. We presented two different models with different devices for controlling the emotional expressions of arbitrary virtual characters in realtime. Fig. 5 shows the eight Basic Emotions in their highest intensities according to our system. Our results can be used for the recognition and synthesis of facial expressions. The synthesis can be done automatically as for virtual agents or manually as for virtual characters in computer animated movies and puppetry.

Besides integrating phonems into the repertoire it would be interesting to personalize the emotions, especially for an automatic synthesis. This could be achieved by not only using the Facial Expression Repertoire as translator between Action Units and Basic Emotions but integrating some kind of Fuzzy Logic or parameterisation for example.

Another interesting way of using our system would be to use real faces as input. Either the emotions or the performed Action Units could be computed and mapped onto a virtual character interactively.

Different participants of the evaluation suggested to mix both existing systems. The idea is to navigate roughly with the mouse and then add more emotions using the mixer or the Spacemouse.

Last but not least there is an application which is being discussed with therapists. The framework could be used as a therapeutic tool for patients with emotional perception disturbances.

7 REFERENCES

[BW10] Filmakademie Baden-Württemberg. Facial expression repertoire. http://research.animationsinstitut.de/45.0.html, 2010.

[EF78] P. Ekman and W.V. Friesen. *Facial Action Coding System: Investigator's Guide*. Palo Alto: Consulting Psychologists Press, 1978.

[HBSL04] V. Helzle, C. Biehn, T. Schlömer, and F. Linner. Adaptable setup for performance driven facial animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, page 54, New York, NY, USA, 2004. ACM.

[KMMTT91] P. Kalra, A. Mangili, N. Magnenat-Thalmann, and D. Thalmann. SMILE:

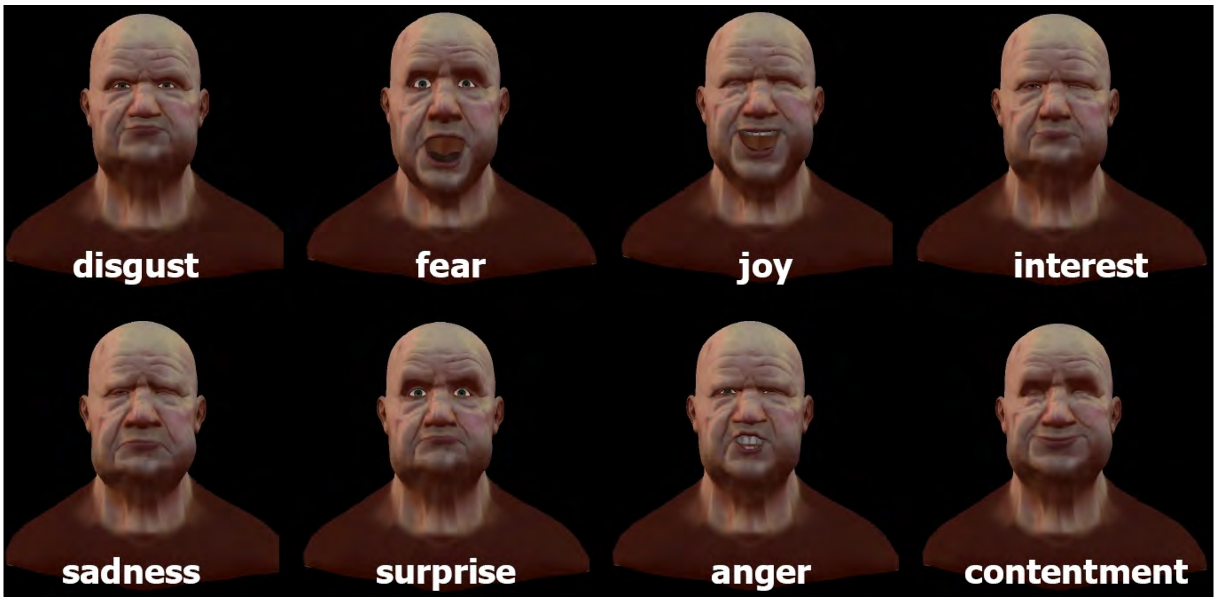


Figure 5: Eight Basic Emotions

a multilayered facial animation system. In *In T.L. Kunii, editor, Modeling in Computer Graphics*, pages 189–198. Springer-Verlag, 1991.

[KMTM+98] P. Kalra, N. Magnenat-Thalmann, L. Moccozet, G. Sannier, A. Aubel, and D. Thalmann. Real-time animation of realistic virtual humans. *Computer Graphics and Applications, IEEE*, 18(5):42–56, 1998.

[Koe02] R. Koenen. *Overview of the MPEG-4 Standard.* International Organization for Standardization, ISO/IEC JTC1/SC29/WG11 N2323, Coding of Moving Pictures and Audio, Geneva, Switzerland, 2002.

[MTM00] N. Magnenat-Thalmann and L. Moccozet. Virtual humans on stage, 2000.

[Plu62] R. Plutchik. *The Emotions: Facts, Theories and a New Model.* New York: Random House, 1962.

[Plu80] R. Plutchik. *Emotion: A Psychoevolutionary Synthesis*. Harper & Row, 1980.

[Rei00] Rainer Reisenzein. Wundt's three-dimensional theory of emotion. *Poznan Studies in the Philosophy of the Sciences and the Humanities*, 75:219–250, 2000.

[RSV09] P. Rodrigues, A. Sá, and L. Velho. Virtual emotion to expression: A comprehensive dynamic emotion model to facial expression generation using the mpeg-4 standard. In J. S. Wright and L. M. Hughes, editors, *Computer Animation*, chapter 6. Nova Science Publishers, 2009.

[Rus80] J. A. Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178, 1980.

[Rus86] M. Russell, J. A.; Bullock. On the dimensions preschoolers use to interpret facial expressions of emotion. *Developmental Psychology*, 22:97–102, 1986.

[Rus02] J. A. Russell. Reading emotions from and into faces: Resurrecting a dimensional-contextual perspective. In James A. Russell & Josi-Miguel Fernandez-Dols, editor, *The Psychology of Facial Expression*. Cambridge University Press, 2002.

[Rus03] J. A. Russell. Core affect and the psychological construction of emotion. *Psychological Review*, 110:145–172, 2003.

[Sch07] S. Schmidt. Glaubhafte gesichtsausdrücke und animationen für echtzeitanwendungen. Master's thesis, Bauhaus-University Weimar, 2007.

[TMPT88] N. Thalmann-Magnenat, N. Primeau, and D. Thalmann. Abstract muscle actions procedures for human face animation. *Visual Computer*, 3(5), 1988.

Performances Analysis of Underwater Image Preprocessing Techniques on the Repeatability of SIFT and SURF Descriptors

Amine Mahiddine, Julien Seinturier, Jean-Marc Boï, Pierre Drap, Djamal Merad
LSIS umr CNRS 7296
Centre National de la Recherche Scientifique
Marseille, France
surname@univ-amu.fr

ABSTRACT

ROV 3D project aims at developing innovative tools which link underwater photogrammetry and acoustic measurements from an active underwater sensor. The results will be 3D high resolution surveys of underwater sites. The new means and methods developed aim at reducing the investigation time *in situ*, and proposing comprehensive and non-intrusive measurement tools for the studied environment.

In this paper, we made an investigation to find at first a pre-processing method of underwater images that do not require a priori knowledge of the scene in order to increase the repeatability of SIFT and SURF descriptors and, in a second time, finding a method to compute distances which will be less costly in terms of execution time for finding corresponding points.

Keywords
Relative orientation, SIFT, SURF, K-nearest neighbour, Automatic Color Equalization, IACE, Correlation.

1. INTRODUCTION

ROV3D[1] project goal is to develop automated proceedings of 3D surveys, dedicated to underwater environment, using both acoustic and optic sensors. The acoustic sensor allows acquiring a great amount of low resolution data, whereas the optic sensor (close range photogrammetry) allows acquiring a low amount of high resolution data. In practice, a 3D acoustic scanner produces a range wide scan of the scene, and an optic system allows a high resolution restitution (larger scale) of different areas in the scene.

In underwater environment, the image quality is degraded by the significant changes undergone by the light. This is due to two factors: first water absorption by the suspended and dissolved materials, and diffusion, mainly due to particles that scatter the radiation [Pet08a] [Que04a].

When the light crosses the dioptre air / water, one part is reflected while the rest effectively penetrates into the water. However the amount of light that penetrates the water decreases with the height of the water column crossing because water molecules absorb a certain amount of light (which reduces its energy). As a first result, underwater images are becoming darker with increasing depth. Not only the amount of light is reduced with depth, but also the light undergoes a change color depending on the amount of water crossing. The wavelength corresponding to red disappears after a few meters (see Fig.1) and beyond 25m only blue remains [Sch10a].

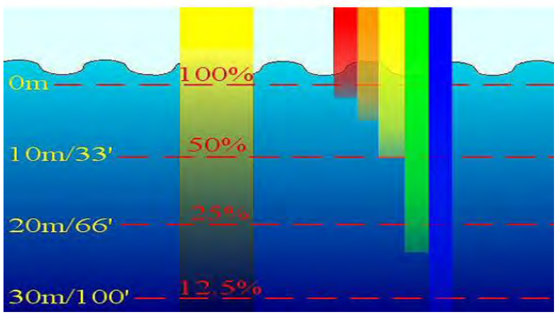


Figure 1. Colour appearance in underwater. [Iqb07a].

[1] http://www.rov3D.eu

Beyond the excessive amount of blue, underwater images therefore have a low brightness and contrast [Sch04a]. They are more affected by suspended particles called sometime « *Marine snow* ».

One of the most important issues in this work is to obtain an image quality for analysis, measurement, and extracting points of interest to optimize image processing and their orientation for the photogrammetric use.

2. UNDERWATER IMAGE PRE-PROCESSING

The underwater image pre-processing can be addressed from two different points of view: image restoration techniques or image enhancement methods.

Fan *et alii* proposed a restoration method based on blind deconvolution and the theory of Wells [Fan10a]. As a first step an arithmetic mean filter is used to perform image denoising, and then an iterative blind deconvolution using the filtered image is carried out. The calculation of the PSF of water is done using the following equations:

$$b = c\omega \qquad (1)$$

$$H_{medium}(\psi, R) = \exp\left\{-cR + bR\left[\frac{1-\exp(-2\pi\theta_0\psi)}{2\pi\theta_0\psi}\right]\right\} \qquad (2)$$

where θ_0 is referred to the median scattering angle, ψ is the spatial frequency in cycles per radian, R is distance between sensor and object, b scattering coefficient, c attenuation coefficient and albedo ω.

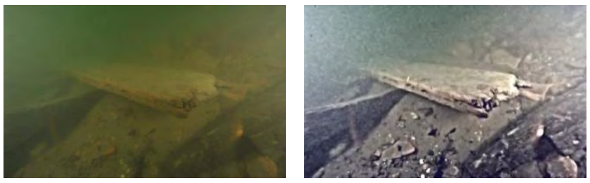
Image restoration techniques need some parameters such as attenuation coefficients, scattering coefficients and depth estimation of the object in a scene. For this reason in our works, the preprocessing of underwater image is devoted to image enhancement methods, which do not require *a priori* knowledge of the environment.

Bazeille *et alii* [Baz06a] proposed an algorithm to enhance underwater image, this algorithm is automatic and requires no parameter adjustment to correct defects such as non-uniform illumination, low contrast and muted colors.

In this algorithm which is based on the enhancement, each disturbance is corrected sequentially. The first step is to remove the *moiré* effect is not applied, because in our conditions this effect is not visible. Then, a homomorphic filter or frequency is applied to remove the defects of non-uniformity of illumination and to enhance the contrast in the image.

Regarding the acquisition noise, often present in images, they applied a wavelet denoising followed by anisotropic filtering to eliminate unwanted oscillations. To finalize the processing chain, a dynamic expansion is applied to increase contrast, and equalizing the average colors in the image is being implemented to mitigate the dominant color. Fig.2 shows the result of applying the algorithm Bazeille *et alii.*

To optimize the computation time, all treatments are applied on the component *Y* in *YCbCr* space. However the use of homomorphic filter changes the geometry, which will add errors on measures after the 3D reconstruction of the scene, so we decided not to use this algorithm.



(a) (b)

Figure 2. Images before (a) and after (b) the application of the algorithm proposed by Bazeille et alii. (Photo by Olivier Bianchimani on the Arle-Rhone 13 roman wreck in Arles, France)

Iqbal *et alii* have used slide stretching algorithm both on RGB and HIS color models to enhance underwater images [Iqb07a]. There are three steps in this algorithm (see Fig.3).

First of all, their method performs contrast stretching on *RGB* and then it converts the result from *RGB* to *HSI* color space. Finally, it deals with saturation and intensity stretching. The use of two stretching models helps to equalize the color contrast in the image and also addresses the problem of lighting.

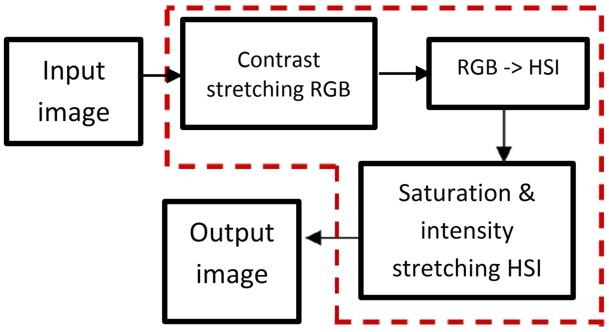


Figure 3. Algorithm proposed by Iqbal et alii .

Chambah *et alii* proposed a method of color correction based on the *ACE* model [Riz04a]. *ACE* “*Automatic Color Equalization”* is based on a new calculation approach, which combines the Gray World algorithm with the Patch white algorithm, taking into account the spatial distribution of information color. The ACE is inspired by human visual system, where is able to adapt to highly variable lighting conditions, and extract visual information from the environment [Cha04a].

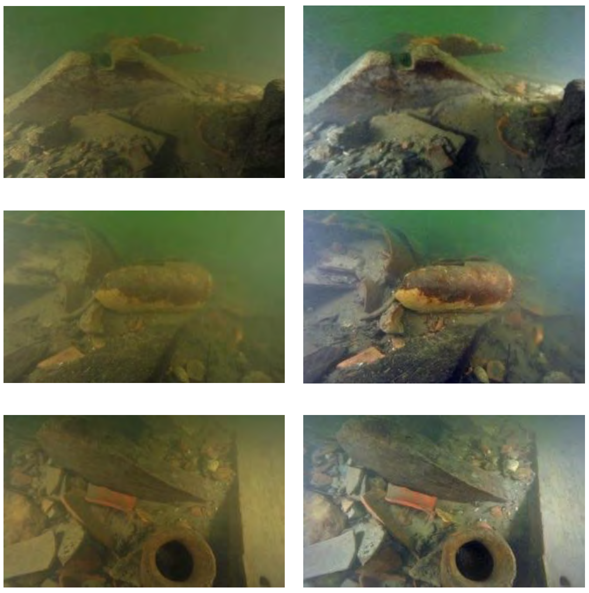
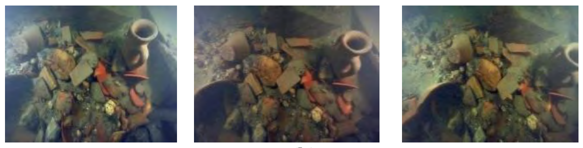

(a) *(b)*

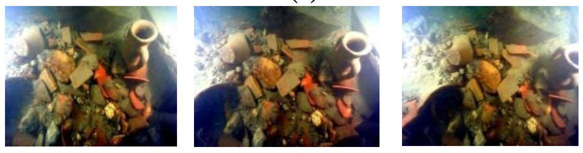
Figure 4. Phtographs of the wreck Arles-Rhône 13, before (a) and after (b) the enhancement by ACE method.[Cha04a].

This algorithm consists of two parts. The first one consists in adjusting the chromatic data where the pixels are processed with respect to the content of the image. The second part deals with the restoration and enhancement of colors in the output image [Pet10a]. The aim of improving the color is not only for better quality images, but also to see the effects of these methods on the *SIFT* or *SURF* in terms of their feature points detection. Three examples of images before and after restoration with *ACE* are shown in Fig.4.

Kalia et alii [Kal11a] investigated the effects of different image pre-processing techniques which can affect or improve the performance of the SURF detector [Bay08a]. And they proposed new method named IACE 'Image Adaptive Contrast Enhancement'. They modify this technique of contrast enhancement by adapting it according to the statistics of the image intensity levels.

(a)


(b)


(c)

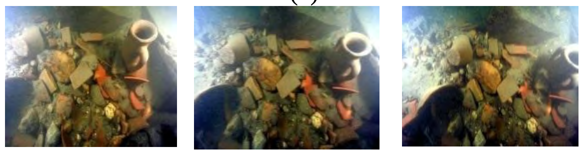

(d)

Figure 5. Phtographs of the wreck Arles-Rhône 13, (a) original images, (b) results by ACE method, (c) results by IACE method « Image Adaptative Contrast Enhancement », (d) results by the method proposed by Iqbal et alii.[Iqb07a].

If P_{in} is the intensity level of an image, it is possible to calculate the modified intensity level P_{out} with equation (3).

$$P_{out} = \frac{(P_{in} - c)}{(d - c)} \times (b - a) \qquad (3)$$

where a is the lowest intensity level in the image and equal to 0, b is its corresponding counterpart and equal to 255 and c is the lower threshold intensity level in the original image for which the number of pixels in the image is lower than 4% and d is the upper threshold intensity level for which the number of pixels is cumulatively more than 96%. These thresholds are used to eliminate the effect of outliers, and improve the intrinsic details in the image while keeping the relative contrast.

The results of this algorithm are very interesting. One can observe that the relative performance of IACE method is better than the method proposed by Iqbal *et alii* in terms of time taken for the complete detection and matching process

3. FEATURE EXTRACTION AND MATCHING

The purpose of preprocessing is improving the quality of images to enhance the detection of interest points. Thereafter, these points of interest will be matched and used for 3D reconstruction of the scene.

There are several methods for extracting interest points such as Edge detector, Corner detector [Guo09a]. Juan *et alii* [Jua09a] made a comparison between *SIFT*, *PCA-SIFT* and *SURF*.

In our work, we decided to use two methods most robust in terms of invariance to the transformation and distortion of images: Scale Invariant Feature Transform "*SIFT*" and speeded-Up Robust Features "*SURF*".

Scale-invariant feature transform "*SIFT*" is a detector and descriptor at the same time proposed by Lowe [Low04a]. it is a method of extracting points of interest that are invariant to changes during image acquisition, these points of interest are local maxima or minima of the difference of Gaussians.

Each point has detected a descriptor vector which is the norm and direction of the gradient in the region around the point of interest. [Lin09a]

Speeded-Up Robust Features "*SURF*" proposed by [Bay08a] is a descriptor invariant to change of scale, rotation and image, this method is divided into two parts, the first part is devoted to the detection of points of interest, where in each scaling the local maxima are calculated using the Hessian matrix. From these local maxima, we choose the candidate points that are above a given threshold which will subsequently be invariant to scaling.

The purpose of the second part of this algorithm is to find a descriptor that will make the points detected invariant to rotation, the *SURF* descriptor is much faster but less robust than *SIFT* and can therefore be used in applications for real time processing.

After using *SIFT* and *SURF* to extract features from images, we implemented some methods for measuring distances, (see Table 1). These methods are often used to compute the similarity between points in a source image and points in a target image. Functions *SSD*, and *SAD, NSSD* calculate the level of dissimilarity between two points where the best result corresponds to the minimum value obtained after the computation.

We also added the method proposed by Lowe, this method is based on the K-Nearest Neighbour algorithm (*KNN)* with a modified search using the *kd-tree* to find corresponding points using Euclidean distance and optimize the calculation time.

Distance	Definition
Sum of Squared Differences **SSD**	$\sum_u (f_i(u) - f_j(u))^2$
Normalise Sum of Squared Differences **NSSD**	$\sum_u \left(\frac{(f_i(u) - \overline{f_i})}{\sqrt{\sum_u ((f_i(u) - \overline{f_i}))^2}} - \frac{(f_j(u) - \overline{f_j})}{\sqrt{\sum_u ((f_j(u) - \overline{f_j}))^2}} \right)^2$
Sum of Absolute Distances **SAD**	$\sum_u \left\| f_i(u) - f_j(u) \right\|$
Euclidean Distance	$\sqrt{\sum_{u=1}^{n} (f_i(u) - f_j(u))^2}$

Table 1 Distance measurements.

4. EXPERIMENTS

In our experiments, we took two sets of 14 images taken by photographer Olivier BIANCHIMANI in July 2012 on the *Arles-Rhone 13* roman wreck in the Rhodano river, south of France. We reduced the resolution of these photographs to 639 x 425 pixels in order to reduce computation time. These two sets are taken in two different situations. We choose for the first, a scene where we can see in Fig.2 the presence of wood pieces and in the second (see Fig.5), a scene with amphora and stones. The choice of these situations was to work on a real underwater scene and test the robustness of detectors and descriptors in the different conditions that we can cross in a marine environment.

The implementation was run on an *Intel Core i7* CPU *980* at *3.33* GHZ with *12*GB of *RAM* under Windows 7 operating system. We studied the effects of different methods that can affect or improve the performance of repeatability of a descriptor. Initially, we noticed improvements in color quality and we also see that the algorithm proposed by *Iqbal et alii* gives the best visual results.

Our approach is to detect points of interest on all images using *SIFT* or *SURF* descriptors. Subsequently, images are matched two by two with one of methods of distance measurement mentioned above in Table 1. For each matched pair of images, the relative orientation is computed using the *5* points algorithm proposed by [Ste06a].

From these orientations, an approximate value of orientations and coordinates of object points are calculated. Then a bundle adjustment is applied for optimal estimation of orientation parameters and 3D coordinates as illustrated in Fig.14.

We cannot give the results for all tests because of the space limitations. In the Table 2 and Table 3, we present some results obtained after several tests. These tables summarize the tests performed with SURF and SIFT descriptors on the original images and preprocessed images, the purpose of these tests as a first step is to find the best preprocessing algorithm in terms of color correction and preprocessing time and which mainly increases the repeatability of descriptors. In a second step, we seek to find the most appropriate method for calculating distances with the type of images that we used in our work which will give more points matched and remove outliers.

We judge the quality of these descriptors according to the number of image pairs oriented, the number of corresponding points and the reprojection error calculated both with the Root Mean Square (*RMS)* and the average error methods. We found that the *RMS* is always less than 0.5 then we have focused only on the average error (see Fig.7 and Fig.9).

The results obtained with images from the three preprocessing methods are better in comparison to results obtained with original images. However, the *ACE* took an hour and *35* seconds, for the same image *IACE* took *0.13* seconds and the method proposed by Iqbal *et alii* took *0.15* seconds almost the same time as the *IACE* method.

Before choosing the best method of preprocessing to be used in our future work, we started first by the choice of method of measuring distances where it was found that the method used by D. Lowe which is based on the algorithm *KNN* performed best in terms of points matches and computation time (see Fig.6, Fig.8, Fig.10, Fig.12), otherwise the *SSD* method and its normalized version *NSSD* also produce good results in terms of matched points and the number of pairs oriented but requires more time for the computation.

Finally after several tests, we found that the *IACE* and the method proposed by Iqbal *et alii* are quite efficient in terms of preprocessing time and number of matched points. However we cannot make a choice between these methods because the results depend on image quality and nature of objects which are located in the scene. In Table 2 and Table 3 we presented the results obtained in two different situations of a marine environment, where the *IACE* method with *SIFT* and *SUFT* descriptors gave the best result for the first situation. However, the second situation the method proposed by Iqbal *et alii* with *SURF* descriptor gave *671* matched points against *413* matched points with *IACE* and the same descriptor.

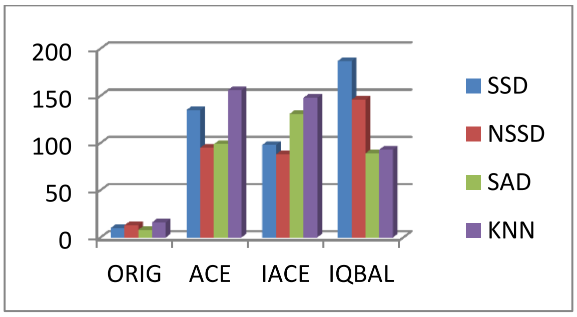


Figure 6 Matching points obtained with SURF descriptors in the first scene with presence of wood pieces.

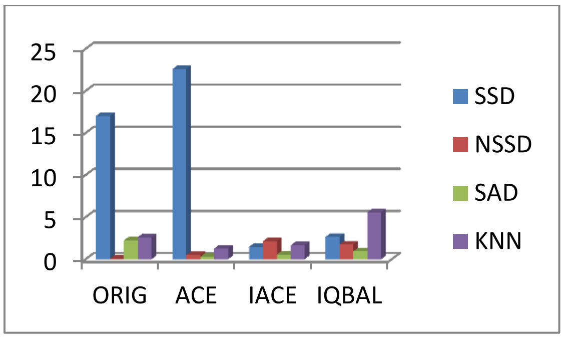


Figure 7 Average error.

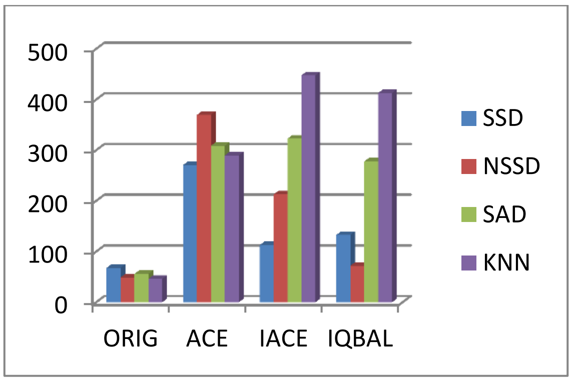


Figure 8 Matching points obtained with SIFT descriptors in the first scene with presence of wood pieces.

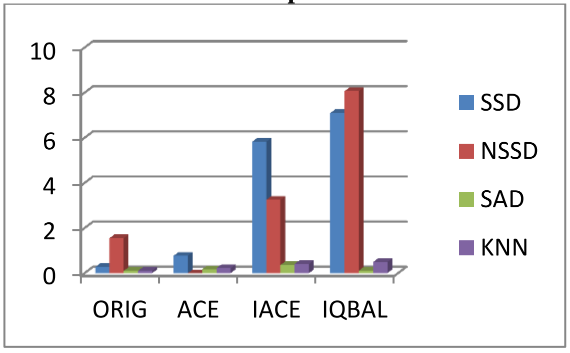


Figure 9 Average error.

SURF		SSD	NSSD	SAD	KNN
ORIG 630x425 pixels	Pairs /91	2	2	2	3
	Time (s)	22	22	20	21
	RMS	0.05	0.06	0.01	0.05
ACE 1h35	Pairs /91	14	10	7	9
	Time (s)	67	122	36	49
	RMS	0.05	0.04	0.35	0.036
IACE 0.13s	Pairs /91	12	13	9	8
	Time (s)	85	184	44	66
	RMS	0.07	0.07	0.03	0.03
IQBAL 0.15s	Pairs /91	14	18	7	8
	Time (s)	78	134	42	52
	RMS	0.04	0.05	0.03	0.05

(a)

SIFT		SSD	NSSD	SAD	KNN
ORIG 630x425 pixels	Pairs /91	4	5	4	4
	Time (s)	103	113	96	99
	RMS	0.03	0.05	0.01	0.011
ACE 1h35	Pairs /91	16	25	9	11
	Time (s)	253	253	178	238
	RMS	0.04	0.04	0.12	0.01
IACE 0.13s	Pairs /91	17	29	13	11
	Time (s)	268	1493	201	275
	RMS	0.11	0.07	0.01	0.009
IQBAL 0.15s	Pairs /91	15	11	12	12
	Time (s)	311	1413	195	264
	RMS	0.09	0.11	0.01	0.01

(b)

Table 2 Test on a set of photographs of a scene with wood pieces, (a) results with SIFT (b) results with SURF.

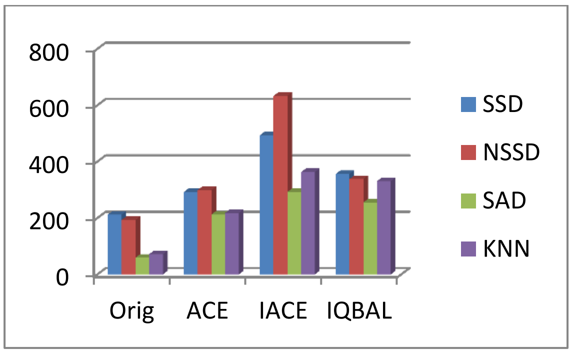


Figure 10 Matching points obtained with SURF descriptor in the second scene with presence of amphora and stones.

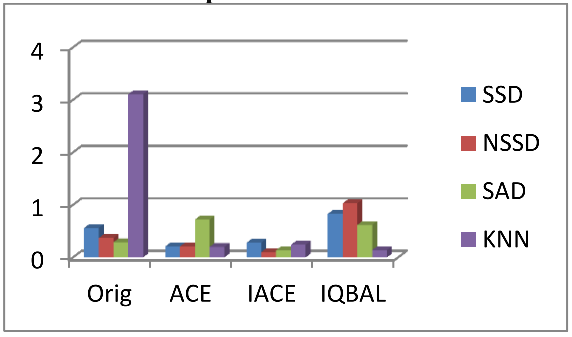


Figure 11 Average error.

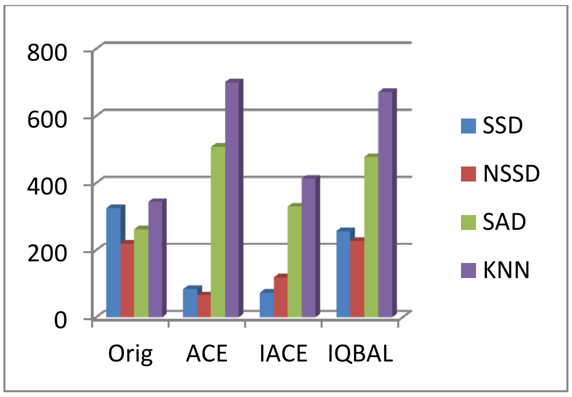


Figure 12 Matching points obtained with SIFT descriptor in the second scene with presence of amphora and stones.

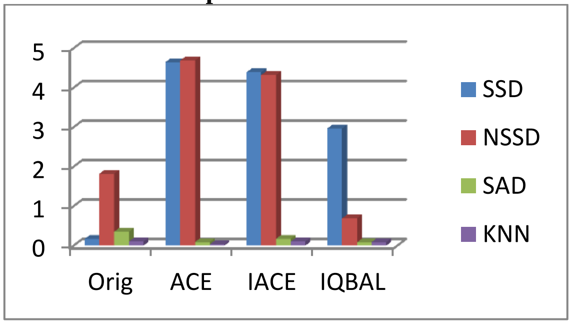


Figure 13 Average error.

SURF		SSD	NSSD	SAD	KNN
ORIG 630x425 pixels	Pairs /91	17	15	6	7
	Time (s)	51	47	26	32
	RMS	0.06	0.05	0.05	0.04
ACE 1h35	Pairs /91	11	11	12	14
	Time (s)	83	193	50	75
	RMS	0.02	0.02	0.03	0.03
IACE 0.13s	Pairs /91	11	12	12	15
	Time (s)	106	356	69	104
	RMS	0.01	0.01	0.02	0.02
IQBAL 0.15s	Pairs /91	14	13	11	14
	Time (s)	99	274	64	96
	RMS	0.03	0.02	0.02	0.02

(a)

SIFT		SSD	NSSD	SAD	KNN
ORIG 630x425 pixels	Pairs /91	15	11	11	11
	Time (s)	152	192	114	134
	RMS	0.02	0.02	0.01	0.01
ACE 1h35	Pairs /91	12	11	15	16
	Time (s)	218	742	163	214
	RMS	0.09	0.11	0.01	0.007

IACE 0.13s	Pairs /91	11	17	11	15
	Time (s)	273	1359	198	269
	RMS	0.07	0.06	0.01	0.01
IQBAL 0.15s	Pairs /91	20	16	12	16
	Time (s)	259	1216	188	260
	RMS	0.04	0.04	0.009	0.009

(b)

Table 3 Test on a set of photographs of a scene with amphora and stones, (a) results with SIFT (b) results with SURF.

As we said earlier, the number of matched points is not a sufficient factor to judge the repeatability of descriptors, for this we calculated the reprojection error to estimate the accuracy of our calculations. To minimize the reprojection error, we thought of an improvement in the quality of matched points, the idea is to correlate in a search window centered on each point matched, if the score of the correlation is less than a threshold then the point is kept if it is the nearest or it is replaced by the nearest point. Otherwise if the score is higher than threshold, the two points are deleted from the list [Kra97a].

Table 4, presents the results of a test applied to a pair of images where we find *200* matched points and a reprojection error of *1.79*. After applying the correlation, *160* points have moved and *40* points are deleted and the new reprojection error is reduced to *0.74* which means that the correlation corrects the quality of matched points. The disadvantage of this improvement is that it can be applied only on a pair of images with very weak relative rotation and a slight change of scale which is the case in our experiments and in the field of stereovision.

Features in left	Features in right	Matched points	RMS	Shifted points	Removed points	RMS minimized
749	696	200	1.79	160	40	0.74

Table 4 Table showing the result after application of correlation.

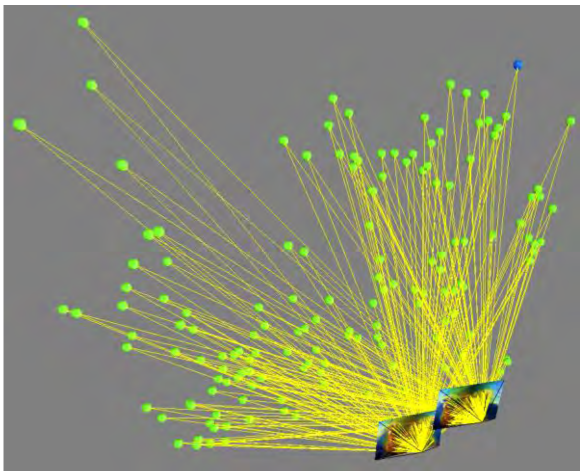


Figure 14 Example of the orientation of a pair of images.

5. CONCLUSION & FUTURE WORK

In this paper, we studied three preprocessing methods whose purpose was to improve color and contrast of underwater images and increase repeatability of descriptors compared to original images. We have also presented some methods for measuring distances where we found that the *IACE* method and the method proposed by Iqbal *et alii* give almost the same results in terms of computation time and repeatability of *SIFT* and *SURF* descriptors.

The use of one of these methods as an initial method of preprocessing with the *KNN* method for distance measurements gives good results in terms of computation time and the reprojection error compared to results obtained with images without preprocessing. Nevertheless, the *ACE* method is very slow in terms of preprocessing time, however we observed an improvement of color contrast and a brightness correction. For this reason, we plan to use the images obtained as texture after the full 3D reconstruction of the underwater scene.

We also showed in this paper, the usefulness of the correlation to minimize the reprojection error in the case of a small rotation between images. The future work is to improve the algorithm of SIFT and test Earth Mover's Distance to find the corresponding points.

6. REFERENCES

[Bay08a] Herbert Bay, Andreas Ess, Tinne Tuytelaars, & Luc Van Gool. (2008). Speeded-Up Robust Features (SURF). Comput. Vis. Image Underst., Vol.:110, pp.346-359, isbn/issn:1077-3142.

[Baz06a] Stéphane Bazeille, Isabelle Quidu, Luc Jaulin, & Jean-Phillipe Malkasse. (2006, 16 - 19 Octobre 2006). Automatic Underwater Image Pre-

Processing. Paper presented at the CMM’06 - CARACTERISATION DU MILIEU MARIN

[Cha04a] Majed Chambah, Dahbia Semani, Arnaud Renouf, Pierre Courtellemont, & Alessandro Rizzi. (2004). Underwater Color Constancy : Enhancement of Automatic Live Fish Recognition. Paper presented at the 16th Annual symposium on electronic imaging,, United States.

[Fan10a] Fan Fan, Yang Kecheng, Fu Bo, Xia Min, & Zhang Wei. (2010, 9-11 April 2010). Application of blind deconvolution approach with image quality metric in underwater image restoration. Paper presented at the Image Analysis and Signal Processing (IASP), 2010 International Conference on. pp.236-239.

[Guo09a] Chenguang Guo, Xianglong Li, Linfeng Zhong, & Xiang Luo. (2009, 21-22 Nov. 2009). A Fast and Accurate Corner Detector Based on Harris Algorithm. Paper presented at the Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on. Vol.:2, pp.49-52.

[Iqb07a]) Kashif Iqbal, Rosalina Abdul Salam, Azam Osman, & Abdullah Zawawi Talib. (2007). Underwater Image Enhancement Using an Integrated Colour Model. IAENG International Journal of Computer Science, Vol.:34.

[Jua09a] Luo Juan, & Oubong Gwon. (2009). A Comparison of SIFT, PCA-SIFT and SURF. International Journal of Image Processing (IJIP), Vol.:3, pp.143-152.

[Kal11a] Robin Kalia, Keun-Dong Lee, Samir B.V.R., Sung-Kwan Je, & Weon-Geun Oh. (2011, 9-11 Fev.). An analysis of the effect of different image preprocessing techniques on the performance of SURF: Speeded Up Robust Feature. Paper presented at the 17th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV). pp.1-6.

[Kra97a] Karl Kraus. (1997). Photogrammetry vol 1 & 2 (Stewardson Peter, Trans. Dummlerbush ed. Vol. 2). Bonn, Germany.

[Lin09a] Andrea Lingua, Davide Marenchino, & Francesco Nex. (2009). Performance Analysis of the SIFT Operator for Automatic Feature Extraction and Matching in Photogrammetric Applications. Sensors, Vol.:9, pp.3745-3766, isbn/issn:1424-8220.

[Low04a]David Lowe. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, Vol.:60, pp.91-110.

[Pet08] Frédéric Petit, Philippe Blasi, Anne-Sophie Capelle-Laizé, & Jean-Christophe Burie. (2008). Underwater Images Enhancement by Light Propagation Model Reversion. Paper presented at the IS&T Fourth european conference on Color in Graphics Imaging and Vision, Terrassa : Espagne.

[Pet10] Frédéric Petit. (2010). Traitement et analyse d’images couleur sous-marines : modèles physiques et représentation quaternionique. Doctorat, Sciences et Ingénierie pour l'Information, Poitier.

[Que04a]Jose P. Queiroz-Neto, Rodrigo Carceroni, Wagner Barros, & Mario Campos. (2004). Underwater Stereo. (Conférencier invité) Proceedings of the Computer Graphics and Image Processing, XVII Brazilian Symposium.

[Riz04a] Alessandro Rizzi, & Carlo Gatta. (2004). From Retinex to Automatic Color Equalization: issues in developing a new algorithm for unsupervised color equalization. Journal of Electronic Imaging, Vol.:13, pp.75-84.

[Sch04a] Yoav Y. Schechner, & Nir Karpel. (2004). Clear Underwater Vision. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04), Vol.:1, pp.536-543.

[Sch10a] Raimondo Schettini, & Silvia Corchs. (2010). Underwater Image Processing: State of the Art of Restoration and Image Enhancement Methods. EURASIP Journal on Advances in Signal Processing, Vol.:2010, pp.14.

[Ste06a] Henrik Stewenius, C. Engels, & David Nister. (2006). Recent developments on direct relative orientation. Isprs Journal of Photogrammetry and Remote Sensing, Vol.:60, pp.284-294, isbn/issn:0924-2716.

Unsupervised Perception-based Image Restoration of Semi-transparent Degradation using Lie Group Transformations

V. Bruni, E. Rossi, D. Vitulano
Dept. SBAI University of Roma Sapienza, Via A. Scarpa 16,00161 Rome, Italy
Istituto per le Applicazioni del Calcolo, C.N.R., Via dei Taurini 19, 00185 Rome, Italy
bruni@dmmm.uniroma1.it {e.rossi,d.vitulano}@iac.cnr.it

ABSTRACT

This paper presents a generalized model for the removal of semi-transparent defects from images of historical or artistic value. Its main feature is the combination of Lie group transformations with human perception rules that makes restoration more flexible and adaptive to defects having different physical or mechanical causes. Specifically, Lie groups allow to define a redundant set of transformations from which it is possible to automatically select the ones that better invert the physical formation of the defect. Hence, the restoration process consists of an iterative procedure whose main goal is to reduce defect visual perception. The proposed restoration method has been successfully tested on original movies and photographs, affected by line-scratches and semi-transparent blotches.

Keywords: Image Restoration, Lie Groups, Human Visual System, Semi-transparent Defects.

1 INTRODUCTION

In the last years, there has been an increasing demand for the fruition of archived material thanks to the growing development of digital devices. Hence, a lot of research effort has been devoted to propose novel and adaptive digital restoration methods able to deal with image defects like noise, line-scratches, tear, moire, blotches, shake and flicker — see [1]-[12]. If on the one hand, the variety of degradations has led to the definition of specific detection and restoration models in order to guarantee a better precision and adaptivity to different scenarios; on the other hand, the advent of new devices and the up-to-date digital technology has led to the need of a general framework able to simultaneously and globally manage different kinds of degradation. This entails the definition of a new restoration paradigm that puts human eye as the final image consumer and judge, independently of the specific processing method. That is why human perception is gaining a significant role in image processing techniques [13]-[16]. In this perspective, the main goal of digital restoration must be the perceived quality of the restored images. In [17] the authors tried to formalize a global detection/restoration framework based on both physical and visual characteristics of the class of analysed defects. It was mainly based on the following observation. Image defects are detected by human eye 'at first glance' even in complicated contexts as they represent 'anomalies' in natural images. Hence, the reduction of the visual contrast of the degraded region (visual anomaly) should decrease the visual contribution of the degraded area without creating new artifacts. The model basically simulates the Human Visual System (HVS) response to the presence of the defect by projecting the degraded image J into a new space where the defect becomes the most visible object of the scene. This space depends on both the physical cause of the defect and the resolution at which the defect shows its greatest visibility and it allows to define appropriate and automatic detection and restoration operators. They still depend on the physical cause of the defect that, in turn, gives the 'a priori' information that makes the restoration process somewhat independent of the specific image. The deeper the knowledge about degradation, the lower the dependence of restoration on the original image.

However, it is not always possible to have detailed information about the degradation under exam. Moreover, though their different original causes, many defects can show the same features on the image, such as color, shape, etc.. For example, water blotches or foxing often have the same visual appearance on the image, as well as cracks and scratches, or yellowing and fading. Anyway, independently of their different nature, these defects preserve the perception at first sight as common and primary feature.

The main goal of this paper is then the definition of a general restoration framework that aims at being almost independent of a priori specific assumptions on the degradation under exam. The framework is required to select suitable transformations from a redundant set only accounting for reduced 'a priori' information about the defect under exam, as, for example, the semi-transparency of the degraded region. In this way, the same restoration algorithm can be used for a very wide class of image degradations. This kind of framework could really be of interest in several applications since it promotes and facilitates its use by non expert people and, at the same time, it avoids the development of integrated softwares able to deal with specific kinds of degradation. To this aim the role of HVS has to be emphasized in the whole restoration process by introducing proper mathematical concepts and tools. This paper represents a first contribution to this challenging purpose. It will use Lie groups theory combined with HVS rules for defining continuos transformations that also include the ones that better correlate with the degradation process. In fact, a local contrast-based restoration process that embeds transformations in a Lie group gives us the opportunity of defining a redundant set of transformations that also contains the inversion of the unknown degradation process. Furthermore, it allows to develop a restoration algorithm that automatically selects the more suitable transformations for points having the same visual contrast. In fact, infinitesimal operations in Lie algebras and their integration in global transform in Lie groups are able to model some human visual phenomena, as deeply investigated in [18] and [19]. In this paper Lie groups are of great importance since the final solution (original not degraded image) is not known in advance as well as the exact degraded process. The only assumption is the knowledge of the degradation map. However, the flexibility of the proposed model allows this map to be not precise as it will be discussed later. Hence, the proposed model has the following advantages: *i)* the combination of HVS and Lie algebra allows the restoration model to have not a precise target to converge. The model is only required to force the contrast of the final solution to be in a suitable range of values according to typical contrasts of the surrounding clean image — degradation has to be invisible. *ii)* the variability of Lie group transformations and their combination allows a more flexible model for the degraded image — they also include the simpler and widely used translation and shrinking operations [9, 4, 2].

The remainder of the paper is the following. Next section gives a brief review of Lie Algebra and Lie group transformations. Section 3 presents the proposed restoration methodology and its refinement for two kinds of semi-transparent defects: line-scratches and blotches. Finally, Section 4 presents some experimental results and concluding remarks.

2 LIE ALGEBRAS AND LIE GROUPS TRANSFORMATIONS

In the following, we give few mathematical details about Lie algebra and groups useful to understand the proposed approach. For a complete treatment of this topic see, for instance, [20] and [21].

A finite Lie group G is both a multiplicative group and a differentiable manifold, that is G is a group locally diffeomorphic to R^n, if n is its dimension. As a result, a Lie group G has got both algebraic and geometric properties, thanks to the group structure and the differentiable structure respectively, and they are deeply related. Finally, every Lie group of finite dimension can always be viewed as a matrix group.

Since a Lie group is a manifold, it has a tangent space at the identity element e, called its Lie algebra, namely $\mathfrak{g}$, which is a vector space of the same dimension of G. The exponential map $\exp : \mathfrak{g} \to G$ gives a natural way to move from the Lie algebra $\mathfrak{g}$ (vector space) to the group G (manifold) and, in the case of finite matrix group, it has a very simple form since it corresponds to matrix exponential: if $X \in \mathfrak{g}$, i.e. X is a tangent vector at e in G, then $\exp(X) = \sum_{n=0}^{\infty} \frac{X^n}{n!}$.

Most of the matrix Lie groups can be used to describe transformations in the plane or in the space. The dimension of the group is the number of free parameters needed to describe the transformations; its Lie algebra elements are tangent vectors at the identity and represent infinitesimal transformations of the points. In this paper we are interested in projective transformations that can be described as a group matrix, P_n, acting on points of $\mathbf{R}^n$ expressed in homogeneous coordinates, with the convention that the $(n+1)-$th value in the coordinates is always scaled back to 1. Projective transformations are characterized by $m = n(n+2)$ parameters (dimension of P_n), described by the elements $G_1, G_2, \ldots, G_m$ of a Lie algebra basis. For instance, for $n = 2$

$$G_1 = \begin{pmatrix} 0&0&1\\0&0&0\\0&0&0 \end{pmatrix} \quad G_2 = \begin{pmatrix} 0&0&0\\0&0&1\\0&0&0 \end{pmatrix} \qquad \textit{translations}$$

$$G_3 = \begin{pmatrix} 1&0&0\\0&0&0\\0&0&0 \end{pmatrix} \quad G_4 = \begin{pmatrix} 0&0&0\\0&1&0\\0&0&0 \end{pmatrix} \qquad \textit{scaling}$$

$$G_5 = \begin{pmatrix} 1&0&0\\0&-1&0\\0&0&0 \end{pmatrix} \textit{rotation} \quad G_6 = \begin{pmatrix} 0&1&0\\1&0&0\\0&0&0 \end{pmatrix} \textit{shear}$$

$$G_7 = \begin{pmatrix} 0&0&0\\0&0&0\\1&0&0 \end{pmatrix} \quad G_8 = \begin{pmatrix} 0&0&0\\0&0&0\\0&1&0 \end{pmatrix} \quad \textit{projections.}$$

Hence, every real linear combination of $G_1, ..., G_m$ is an infinitesimal projective transformation in the vector space that corresponds to a transformation of the group P_n thanks to the exponential map. The infinitesimal transformation of a generic point $p \in \mathbf{R}^n$ is $\tilde{L}_j = G_j\tilde{p}$, where $\tilde{p}$ is the point p expressed in homogeneous coordinates. We denote by L_j the corresponding affine coordinates of $\tilde{L}_j$, $\quad j = 1, \ldots, m$. Hence, for $n = 2$, we have

$$L_1 = \begin{pmatrix} 1\\0 \end{pmatrix} \quad L_2 = \begin{pmatrix} 0\\1 \end{pmatrix} \quad L_3 = \begin{pmatrix} x\\y \end{pmatrix} \quad L_4 = \begin{pmatrix} x\\-y \end{pmatrix}$$

$$L_5 = \begin{pmatrix} y\\-x \end{pmatrix} \; L_6 = \begin{pmatrix} y\\x \end{pmatrix} \; L_7 = \begin{pmatrix} xy\\y^2 \end{pmatrix} \; L_8 = \begin{pmatrix} x^2\\xy \end{pmatrix}.$$

3 THE PROPOSED RESTORATION MODEL

The degraded image J at the point $p = (x, y)^T$ can be modeled as

$$J(p) = \mathcal{T}(I(p)),$$

where $\mathcal{T}$ is the unknown degradation transformation and I is the original image. The goal should be to find the inverse of $\mathcal{T}$, namely $\mathcal{T}^{-1}$, in order to reconstruct the original image I. The key point is that $\mathcal{T}$ is unknown. The proposed model set a suitable group of eventually redundant transformations where automatically select $\mathcal{T}^{-1}$. In particular, the selected group is the group of projective transformations (in the plane or in the space as we will explain later): it contains translation and shrinking operations commonly used in restoration models but also rotations, shears and projections. Hence, we fix a group of transformations and the iterative procedure that selects their best composition, rather than a specific type of transformation. Moreover the projective group is a Lie group and its geometrical structure is exploited in order to define a simple iterative procedure to select $\mathcal{T}^{-1}$.

3.1 Distance minimization

The iterative procedure presented in [22] has been used. It mainly exploits the relation between Lie algebras and Lie groups to map a given submanifold S_1 of $\mathbb{R}^n$ to another one, S_2, through a suitable composition of transformations of the group, minimizing their distance. More precisely, let $p \in S_1$, n_p the unit normal at S_1 in p and d_p the distance between p and S_2 along n_p. So $d = \sum_{p\in S_1} d_p$ is the global distance between S_1 and S_2. The Lie group structure allows to look for an infinitesimal transformation inside the Lie algebra (that is a vector space), instead of a global transformation of the group, and then to move it to the group by the exponential map. So the problem is linear and the goal is just to find $\alpha_1, ..., \alpha_m \in \mathbb{R}$, such that the corresponding infinitesimal action on p, that is $\sum_{j=1}^m \alpha_j L_j^p$, projected onto the normal direction n_p minimizes d, that is

$$(\alpha_1, \ldots, \alpha_m) =$$
$$= arg \min_{\alpha_j} \sum_{p\in S_1} \left(d_p - \sum_{j=1}^m \alpha_j \left(L_j^p \cdot n_p \right) \right)^2 . \quad (1)$$

Therefore, $\vec{\alpha} = (\alpha_1 \ldots \alpha_m)^T$ is such that $\vec{\alpha} = \vec{A}^{-1}\vec{b}$, where $\vec{A}$ is the matrix whose elements A_{jh} are defined as follows

$$A_{jh} = \sum_{p\in S_1} \left(L_j^p \cdot n_p \right) \left(L_h^p \cdot n_p \right)$$

while $\vec{b}$ is a column vector whose elements are

$$b_h = \sum_{p\in S_1} d_p \left(L_h^p \cdot n_p \right).$$

Hence, $t = \sum_{j=1}^m \alpha_j G_j$ is the infinitesimal transformation in the Lie algebra that minimizes the distance between S_1 and S_2 and $T = \exp(t)$ its corresponding element in the group. S_1 is updated applying T to its points and the minimization algorithm is applied to the new couple $\left(S_1^{(1)}, S_2 \right)$,where $S_1^{(1)} = T(S_1)$, and so on until the distance between S_1 and S_2 is small enough. For the numerical computation of $\exp(t)$ applied to generic point p, a 4th order Runge Kutta algorithm can be used — see [22] for details. It is equivalent to cut the 4th order series expansion of the matrix exponential and apply it to the point p, that is

$$T \approx I + t + \frac{1}{2}t^2 + \frac{1}{6}t^3 + \frac{1}{24}t^4,$$

but it directly manages affine coordinates.

It is worth stressing that this procedure has much in common with the basic concepts of convex projections for restoration, described in [23]. However, in this case we do not use neither convex sets nor orthogonal projections, but only iterated projective transformations in $\mathbb{R}^n$ and their algebraic and geometrical properties.

3.2 HVS embedded in the minimization algorithm

We would like to apply the iterative procedure described in the previous section to the damaged image by modeling it as a suitable submanifold S_1 of $\mathbb{R}^n$, in order to select the transformations that move S_1 towards the clean image S_2. It is worth observing that for blotches restoration, $n = 3$ and

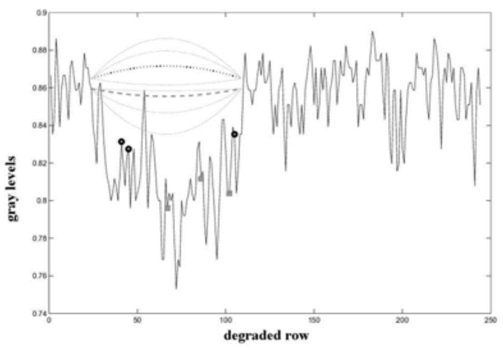


Figure 1: Parabolas are the target curves used in the iterative procedure. Markers are in correspondence to two different groups of points selected with the SMQT transform. Dashed and dotted parabolas are their corresponding target curves.

the whole degraded area is modeled as a surface in $\mathbb{R}^3$ while for scratch restoration $n = 2$ and each degraded row is modeled as a curve in $\mathbb{R}^2$. Unfortunately, in case of digital restoration the final clean image is unknown so that the aforementioned procedure would be unfeasible.

To take advantage of the aforementioned method and to preserve the original image information, HVS perception mechanisms can be embedded in the restoration process. They allow us to define a suitable range of admissible intensity values for the damaged area to be not visible with respect to its neighborhood. In other words, to be invisible, the degraded region must be contained in a certain **cone of visibility**, that depends on the global intensity value of the analysed image; at the same time, abrupt changes in the final solution are not allowed in order to avoid artifacts in correspondence to the frontier of the degraded region. The range of admissible values for the final solution cannot exceed the one of the surrounding information, in terms of visibility bounds, in order to be perceived as a natural scene component. The **cone of visibility** for $n = 2$ can be then defined as in Fig. 1, where the upper and lower parabola curves respectively reach the greatest and least allowed values, namely r_2 and r_1, and the initial point of the **cone of visibility** is in the range of invisible luminance value with respect to the average of the surrounding information [25]. To obtain it, the visual contrast of pixels in the area around the initial points, namely R, must satisfy Weber's law, i.e. $\frac{I_R - I_B}{I_R} < \tau$, where I_R and I_B are the luminance of the region R and its background B, while τ is the just noticeable threshold [24]. For $n = 3$, the cone is defined in the same way replacing parabolas with paraboloids. The cone of visibility is then the target S_2. Hence, the degraded image is moved inside the cone by the distance minimization iterative procedure in order to make the damage invisible.

However, a further consideration has to be made for semi-transparent degradation. Despite the wide flexibility of Lie transformations, the minimization process in eq. (1) is global. In fact, at each step the parameters $\{\alpha_j\}_{j=1,\ldots,m}$ are the same for each point of the degraded area. Hence, if on the one hand global transformations preserve the original information contained in the degraded region, on the other hand they forget that pixels may have been subjected to a different amount of degradation. In order to find a tradeoff between preservation of original information and model flexibility, it is necessary to classify damaged pixels accounting for their visual feature and restore them accordingly. We aim at processing in the same way points that are equally perceived by human eye i.e., points having the same visual contrast have to converge to the same target sub-manifold. In order to classify pixels with the same visual contrast, the Successive Mean Quantization Transform (SMQT) [26] is used. It groups pixels having the same visual features. More precisely SMQT builds a binary tree using the following rule: given a set of data D and a real parameter L (number of levels), split D into two subsets, $D_0 = \{x \in D | D(x) \leq \overline{D}\}$ and $D_1 = \{x \in D | D(x) > \overline{D}\}$, where $\overline{D}$ is the mean value of D. D_0 and D_1 are the first level of the SMQT. The same procedure is recursively applied to D_0 and D_1 and so on until the L^{th} level, that is composed by 2^L subsets. Each group belongs to a sub-manifold (defined by interpolation). The target sub-manifold of the i-th group is defined as a paraboloid (parabola) cut by the plane $z = M + \Delta$ (line $y = M + \Delta$) for $n = 3$ ($n = 2$), where M is the mean value of the intensity surrounding values and Δ accounts for the global visibility of the degraded area with respect to the external one, and whose vertex is proportional to the mean value of the group. Hence, each group converges to the corresponding sub-manifold, as shown in Fig. 1, according to the minimization in eq. (1). The iterative minimization process stops when the target sub-manifold has been reached, in agreement with visibility bounds. More precisely, if $S_1^{(K)}$ is the solution at the K-th iteration for the $i-$th group and S_2 the corresponding target sub-manifold, then $S_1^{(K)}$ is an acceptable solution if

$$\frac{\sum_p |S_1^{(K)}(p) - S_2(p)|}{\sum_p S_2(p)} \leq \tau \ , \qquad (2)$$

where the first member is the Weber's contrast [24, 27] evaluated at the points of the analysed sub-manifolds, while τ is the just noticeable detection threshold for that visual contrast [24, 27].

Summing up, HVS is exploited for 1) classifying groups of pixels according to their visual contrast; 2) defining the cone of visibility and the target sub-manifolds inside it that have to be reached by each group of pixels; 3) defining the stopping criteria for the iterative procedure.

In the next sub-section the whole restoration algorithm is presented. Afterward, two well known semi-transparent defects are briefly presented: line scratches and blotches. They represent an interesting case study. In fact, the physical formation of semi-transparent defects can be complex and can depend on several conditions and events that cannot be known or reproduced in real applications. They can be caused by dirt or moisture on archived material as well as mechanical stress of the support. Hence, they often partially obscure image regions (see Fig. 2) and appear as more or less irregular regions with variable shape and size, having a slightly different color from the original one [7, 9]. They can be then easily confused with scene components since they do not completely hide the underlying original information, that must be retained (after restoration) for its historical and/or artistic value.

3.3 The Algorithm

In the following, the whole restoration algorithm is briefly summarized.

1. Estimate the extrema of the **cone of visibility** r_1, r_2 from the surrounding information. They respectively are the minimum and maximum values of a neighborhood of the degraded area;
2. Apply SMQT to the degraded area;
3. Compute the target curves whose vertices are set according to the mean amplitudes of the groups computed in step 2 and the output of step 1;
4. For each group in step 2, apply the iterative procedure in Section 3.2, where the targets are the ones of step 3, until eq. (2) is satisfied;
5. Perform masking refinement.

The last step is different according to the degradation kind. For semitransparent blotches it corresponds to the study of the contrast properties of pixels in order to understand if degraded pixel are already masked by the original image; for line-scratches it corresponds to apply the visibility-based weights in eq. (5), as next section shows.

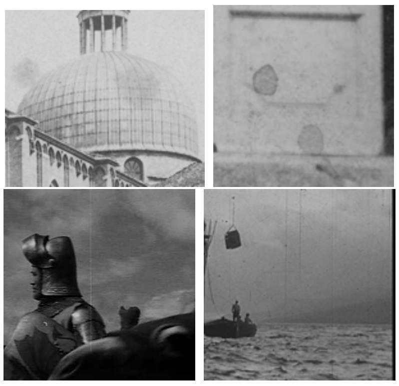


Figure 2: Examples of semi-transparent defects in real photographs and movies.

3.4 Line scratches

Line scratches are common defects on old film sequences. They appear as straight lines spreading over much of the vertical extent of an image frame, as shown in Fig. 2. They can have a different color and are of a limited width [7]. They are often caused by mechanical stress during the projection of a film and occupy the same or a similar location in subsequent frames. The works [1] and [28] provided a physical model for the observed scratches. It has been proved that they are the result of light diffraction effect that occurs during the projection and/or the scanning process. In fact, a scratch is a thin slit on the film material and it is crossed by the light in the projection process. Since the slit (width and depth) is not uniform as a different amount of the original information is removed in the degradation process, the damaged area can be modeled as a partially missing data region and it is well represented by the following equation

$$J(x,y) = (1-(1-\gamma)e^{\frac{-2}{\omega_p}|y-c_p|})I(x,y)+(1-\gamma)L_x(y) \quad (3)$$

where (x, y) are the coordinates of image pixels, $L_x(y)$ is the 1D function model for the scratch, i.e.

$$L_x(y) = b_p sinc^2\left(\frac{y-c_p}{\omega_p}\right), \quad (4)$$

with b_p, c_p and ω_p respectively the maximum brightness, the location (column number) and the horizontal width of the scratch on the image. γ is a normalization parameter that measures the global visibility of the scratch in the degraded image, while $e^{\frac{-2}{\omega_p}|y-c_p|}$ approximates the positive decay of the scratch contribution from its central part toward its end. γ compares the average energy of the peaks of the horizontal cross-section of the image with the one of the scratch and it is in the range $[0, 1]$; hence, the smaller γ the more perceptible the scratch.

Taking into account the reduced horizontal width of a line-scratch, n is set equal to 2, the minimization algorithm is applied row by row and the limiting curves of the **cone of visibility**, as in Fig. 1, are the ones to which the iterative procedure has to converge. However, accounting for the impulsive nature of the defect, as in the equation model (4), a refinement of the final solution is required, according to the scratch visibility with respect to its local context. More precisely, for each point of the scratched area the following weight is applied to the output of the minimization process in eq. (1)

$$\bar{S}_1^{(K)}(x,y) = w(y) S_1^{(K)}(x,y), \qquad (5)$$

where $w(y) = \gamma e^{\frac{-2}{\omega_p}|y-c_p|}$, according to the degradation model in eq. (3).

3.5 Semi-transparent blotches

Such blotches are caused by water penetration into paper or chemical reactions whose final visual effect is a darker region on the document with variable shape, color and intensity. Unfortunately, the lack of distinctive features, like shape and color, does not allow the definition of a precise model for the degraded image. However, part of the original information still survives after the degradation process. Due to the larger physical dimension of the blotch with respect to the line scratch, n is set equal to 3 and the distance minimization algorithm is applied to the whole degraded area modeled as a surface in $\mathbf{R}^3$. The group of projective transformations in the space has dimension 15 and it contains the same types of transformations as for $n = 2$. The restoration process is exactly the one described above. Anyway, due to the high semi-transparency of this kind of defects and its variable dimension, a different amplitude of the cone of visibility has to be used for the darkest and brightest points of the degraded area. Hence, a pre-processing step to separate darkest and brightest region of the damage is required. Moreover, in the masking refinement, the study of contrast properties allows to understand if pixels are already masked by original image information: in this case it may be more convenient to leave them unchanged to avoid the creation of annoying artifacts, as it is described in detail in [29].

4 EXPERIMENTAL RESULTS

The proposed approach has been tested on selected images from the photographic Alinari Archive in Florence, affected by semi-transparent defects and on several real sequences (digitized copies of actual damaged films) having different subjects and of 1-2 minutes length (1500- 3000 frames). Some results are shown in Figs. 3 and 4. It is important to test the method on real damages. Since the precise physical process and the corresponding real transformation are unknown, artificial defects are not representative of real applications on archived material.

In all tests, the size of the neighbouring area of the degraded region is three times the one of the degraded area, while the number of groups of the SMQT in step 2 has been set equal to 8 for semi-transparent blotches and 2 for line scratches. The visual quality of the restored images is very satisfactory. Textures are well preserved as well as details of the original image information, while annoying artifacts, like spikes, halo effects or over-smoothed regions, do not appear. In fact, the different processing of pixels having different contrast value contributes to the flexibility of the restoration model. The convergence process is different for each group of points so that it could happen that some groups converge after one or two iterations while others require longer convergence time. In that way, the restoration process has two main advantages: halo effects at the borders of the degraded region and over-smoothing of the restored pixels are missing. In addition, the refinement procedure allows to successfully deal with some delicate cases, as the intersection between the degraded area and a darker region of the image as the third example in Fig. 3 and the Knight shoulder in Fig. 4, and also to make the restoration process more independent of detection results. Finally, it is worth emphasizing two additional advantages of the proposed method. Even though it involves iterative procedures to converge to the final solution, it uses simple and fast operations so that just 4/5 iterations on average are required for convergence. The time of each iteration depends on both the dimensions of the image and the degraded area. For instance, in the case of the blotch in Fig. 5, each iteration takes 45 seconds on average; in the case of the scratch in Fig. 6, each iteration for each row takes just 1/2 seconds. In addition, the algorithm does not require user's interaction, since it automatically adapts each of its steps to the analyzed image.

For the sake of completeness, the restoration results have been compared with some recent restoration methods of semi-transparent defects [9, 4, 2, 28]. Since the clean image is not available in real applications, quantitative measures or metrics to determine the goodness of the restoration are not convenient. Comparison is then based on the perceived visual quality: some results are shown in Figs. 5 and 6. As it can be observed, the proposed restoration procedure gives high quality im-

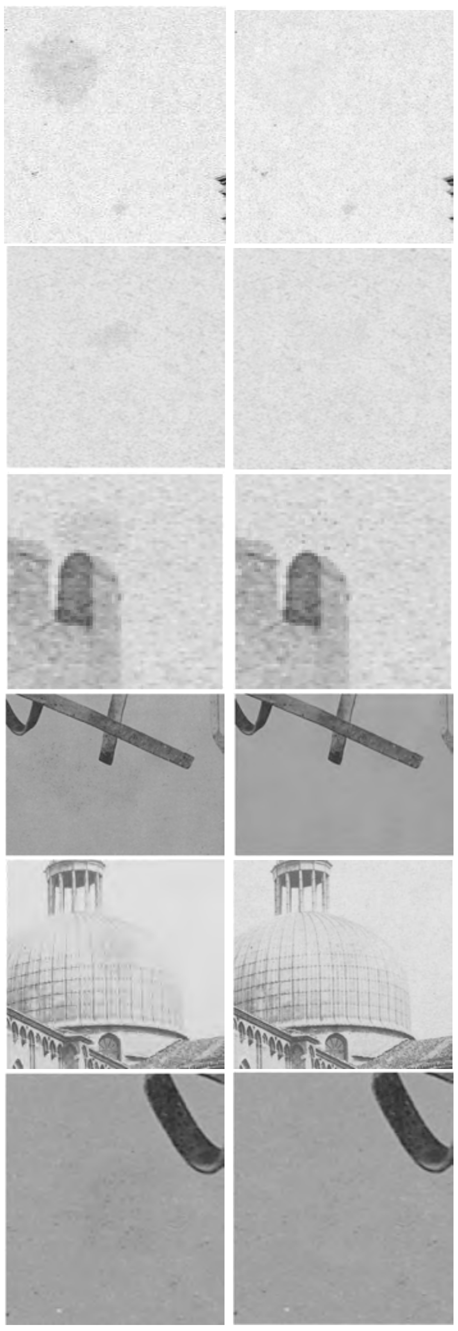


Figure 3: Semi-transparent blotches: Original (*Left*) and restored (*Right*) images.

ages even though it is based on less assumptions about the considered degradation.

All the aforementioned features make the proposed method a valid and promising attempt to the definition of a user's friendly and global restoration framework. Future research will be then oriented to further generalize the proposed restoration framework to make it more flexible and adaptive to a wider class of degradation kinds.

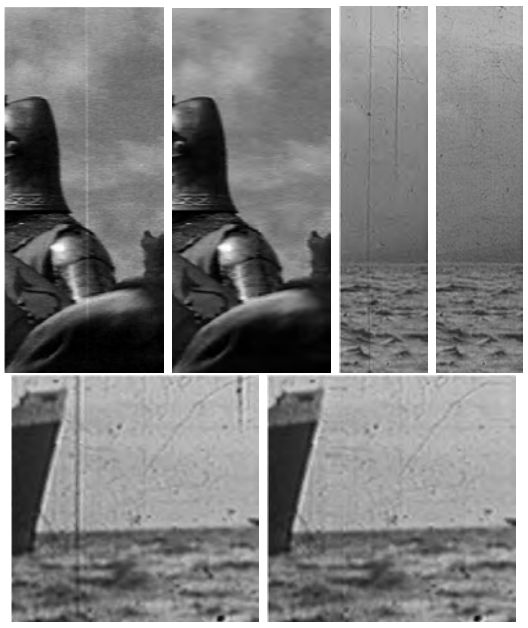


Figure 4: Line-scratches: Original (*left*) and corresponding restored images (*right*).

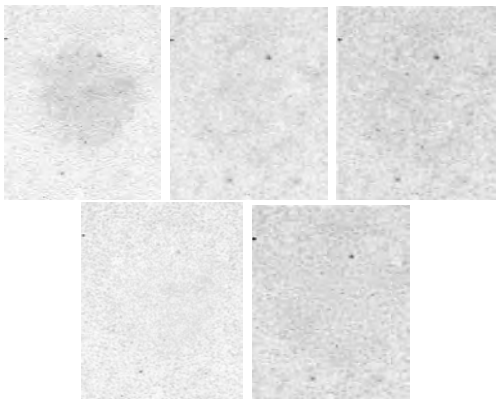


Figure 5: *Clockwise order* Original image, restored using methods in [9, 4, 2], and the proposed one.

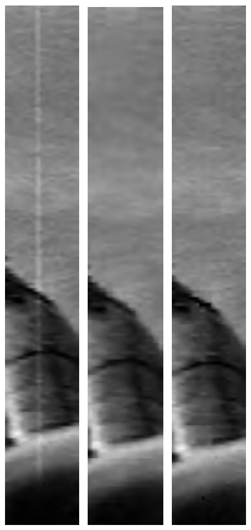


Figure 6: *Left to right* Original image and restored using the methods in [28] and the proposed one.

REFERENCES

[1] V. Bruni, D. Vitulano. A generalized model for scratch detection. IEEE Trans. on Image Proc., Vol. 13, No.1, 44-50, Jan. 2004.

[2] V. Bruni, A. Crawford, A. Kokaram, D. Vitulano, Semi-transparent blotches removal from sepia images exploiting visibility laws, Sig. Image and Video Proc., Springer, 2011.

[3] D. Corrigan, A. Kokaram, Automatic treatment of film tear in degraded archived media. Proc. of ICIP '04, Singapore, Oct. 2004.

[4] A. Greenblatt, S. Agaian, K. Panetta, Restoration of images damaged by semi-transparent water blotches using localized image enhancement, Proc. of SPIE 2008, 2008.

[5] M. Gulu, O. Urhan, S. Erturk, Scratch detection via temporal coherency analysis and removal using edge priority based interpolation, Proc. of IEEE Int. Symp. on Circ. and Syst., 2006, May 2006.

[6] A. Kokaram. On missing data treatment for degraded video and film archives: a survey and a new bayesian approach. IEEE Trans. on Image Proc., Vol 13, No. 3, 397-415, Mar. 2004.

[7] A. Kokaram. Motion Picture Restoration: Digital Algorithms for Artefact Suppression in Degraded Motion Picture Film and Video, Springer Verlag, 1998.

[8] P. van Roosmalen, R. Lagendijk, J. Biemond. Correction of intensity flicker in old film sequences. IEEE Trans. on Circ. and Syst. for Video Tech., Vol. 9, No.7, 1013-1019, 1999.

[9] F. Stanco, L. Tenze, G. Ramponi, Virtual restoration of vintage photographic prints affected by foxing and water blotches, Journal of Electronic Imaging, Vol. 14, No. 4, 2005.

[10] L. Tenze, G. Ramponi, Line scratch removal in vintage film based on an additive/multiplicative model, Proc. of IEEE-EURASIP NSIP-03, Grado, Italy, June 2003.

[11] S. Tilie, I. Bloch, L. Laborelli, Fusion of complementary detectors for improving blotch detection in digitized films, Pattern Recognition Letters, Vol. 28, 1735-1746, May 2007.

[12] T. Vlachos, Flicker Correction for Archived Film Sequences, IEEE Trans. on Circ. and Syst. for Video Tech., Vol. 14, 508-516, Apr. 2004.

[13] S. Agaian, B. Silver, K. Panetta, Transform Coefficient Histogram Based Image Enhancement Algorithms Using Contrast Entropy, IEEE Trans. on Image Proc., Vol. 16, No. 3, Mar. 2007.

[14] H. Sheikh, A. Bovik, Image Information and Visual Quality, IEEE Trans. on Image Proc., Vol. 15, No. 2, Feb. 2006.

[15] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Trans. on Image Proc., Vol. 13, No. 4, 600-612, Apr. 2004.

[16] Z. Wang, Q. Li, Information Content Weighting for Perceptual Image Quality Assessment, IEEE Trans. on Image Proc., Vol. 20, No. 5, 1185-1198, May 2011.

[17] V. Bruni, A. Crawford, A. Kokaram, D. Vitulano, Visual perception of semitransparent blotches: detection and restoration, I-Tech Book: Brain, Vision and AI, chapter 1, 2008.

[18] W. Hoffman, The Lie algebra of visual perception, J. of Math. Psycho., Vol. 3, No. 1, 65-98, Febr. 1966.

[19] P.C Dodwell, The Lie transformation group model of visual perception, Perception and Psychophysics, Vol.34, No. 1, 1-16, 1983.

[20] V. Varadarajan, Lie groups, Lie algebras and their representations, Prentice-Hall Series in Modern Analysis, Englewood Cliffs, N.J., 1974.

[21] S. Helgason, Differential geometry and symmetric spaces, Pure and Appl. Math., Vol. XII. Academic Press, New York-London, 1962.

[22] T. Drummond, R. Cipolla, Application of Lie algebras to visual servoing, Int. J. of Computer Vision, Vol. 37, No. 1, 21-41, 2000.

[23] D. Youla, H. Webb, Image Restoration by the Method of Convex Projections: Part 1 - Theory,IEEE Trans. on Med. Imag., Vol. 1, No. 2, 81 -94, 1982.

[24] S. Winkler, Digital Video Quality. Vision Models and Metrics, Wiley 2005.

[25] R. Gonzalez, R. E. Woods, Digital Image Processing, Prentice Hall, 2nd Edition, 2002.

[26] M. Nilsson, M. Dahl, I. Claesson, The successive mean quantization transform, Proc. of ICASSP, pp. 429-432, 2005.

[27] T. Pappas, R. Safranek, J. Chen, Perceptual criteria for image quality evaluation, Handbook of Image and Video Processing, pp. 939-959, Academic Press, 2005.

[28] V. Bruni, A. Kokaram,D. Vitulano, Fast removal of line scratches in old movies, Proc. of ICPR'04, Vol. 4, pp. 827-830, Aug. 2004.

[29] V. Bruni, E. Rossi, D. Vitulano, Image Restoration via Human Perception and Lie Groups, Proc. of VISAPP 2012, Vol. 1, SciTePress, Feb. 2012.

GPU Based Computation of the Structural Tensor for Real-Time Figure Detection

Marcin Bugaj
AGH University of Science and Technology
Al. Mickiewicza 30
30-059 Kraków, Poland
mm.bugaj@gmail.com

Bogusław Cyganek
AGH University of Science and Technology
Al. Mickiewicza 30
30-059 Kraków, Poland
cyganek@agh.edu.pl

ABSTRACT
In this paper we present a real-time realization of the method of detection of local structures in images of predefined orientation. The method is based on an analysis of the structural tensor computed in monochrome and color images. Thanks to the GPU implementation of the low-level feature detection an order-of-magnitude speed-up was achieved compared to the software implementation. The method can be used for real-time detection of solid objects in HDTV streams as shown by many examples.

Keywords
Structural tensor, corner and edge detection, graphics card, real-time low level feature detection

1. INTRODUCTION
An analysis of low-level features in images constitutes a front-end in many computer vision systems. Thus, there is still an ongoing research on their fast and precise computation methods. One of the very promising methodologies relies on computation of the so called structural tensor (ST) which conveys information on multi-dimensional and multi-scale local neighborhoods of pixels in 2D, 3D and higher dimensional images [1][3].

In this paper we present a method of detection of local features in images of specific orientation. The method relies on prior computation of the ST which is proposed to be done in the graphic card (GPU). For this purpose the HIL library, provided by Cyganek *et al*. [3], as well as the CUDA development environment by nVidia® were used [10][11][12]. The presented method fits well to the concept of smart cameras which are able to precompute and transfer low-level features defined by a user. Thanks to this an order-of-magnitude speed-up was achieved which allows processing of the HDTV streams in real-time, as shown by experiments.

The paper is organized as follows. In section 2 we present an overview of figure detection based on multi-channel and multi-scale structural tensor. In section 3 the architecture of the processing platform is presented. Section 4 contains experimental results. The paper ends with conclusions provided in section 5.

2. FIGURE DETECTION WITH THE STRUCTURAL TENSOR
The structural tensor was first proposed by Bigün et al. [1], and then used by many authors in context of different applications [7]. With the structural tensor each pixel neighborhood can be investigated for their strength and orientations in terms of local intensity or color signal gradients. Having a local neighborhood of pixels Ω, centered at a point $\mathbf{x}_0$, the main idea is to determine a dominating directional vector **w** which is as close as possible to all gradients $\mathbf{q}_i$ in this neighborhood. This is depicted in Figure 1.

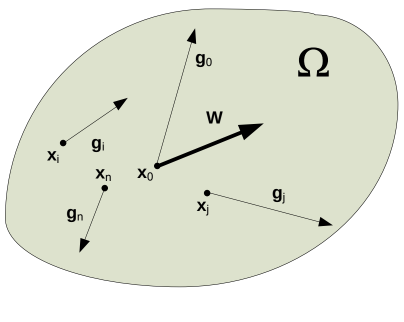


Figure 1. Computation of the dominating orientation vector w in a pixel neighborhood Ω based on signal gradients $\mathbf{q}_i$.

In other words, if such **w** can be determined, then the whole Ω can be represented solely by **w**. Moreover, its parameters, such as phase and magnitude provide us important information on a type of a neighborhood. As will be shown, this can be used to determine corners or local structures with specific orientations. In consequence, ST can be used to detect characteristic figures in images. In our application it was used to detect e.g. road signs in real time, as will be shown in experimental section.

More specifically, to compute **w** of Ω one needs to compute gradient vectors $\mathbf{q}_i$ for all points $\mathbf{x}_i \in \Omega$. For comparison of vectors their inner product is used. Thus, the vector **w** at a point x_0 is an estimator of an average orientation in Ω that fulfills the following optimization problem

$$\arg\max_{\mathbf{w}} \left(\int_{\Omega} \left[\mathbf{q}(\mathbf{x}) \mathbf{w}^{\mathrm{T}}(\mathbf{x}) \right]^2 d\mathbf{x} \right) = \arg\max_{\mathbf{w}} (\mathbf{w}^{\mathrm{T}} \mathbf{T} \mathbf{w}), \qquad \mathbf{(1)}$$

where **q** and **w** are column vectors, whereas **T** denotes the structural tensor, which is defined as follows

$$\mathbf{T}(\mathbf{x}) = \int_{\Omega} \mathbf{q}(\mathbf{x}) \mathbf{q}^T(\mathbf{x}) d\mathbf{x} . \qquad \mathbf{(2)}$$

The square of the inner product in (1) fulfils the invariant assumption on rotation of π radians. Otherwise parallel and anti-parallel configurations of vectors would cancel out. On the other hand, the outer product $\mathbf{qq}^T$ in (2) conveys dimension of the tensor **T**.

In a case of multi-channel color images the gradient vector **q** can be defined as proposed by Di Zenzo [5]. In this approach summation of the partial gradient components in image channels is assumed. To find the ST for images with *M* channels we employ this idea to (2), as follows [3]

$$\mathbf{T} = \int_{\Omega} \sum_{k=1}^{M} \left(\mathbf{q}_k(\mathbf{x}) \mathbf{q}_k(\mathbf{x}) \right) d\mathbf{x} = \sum_{k=1}^{M} \int_{\Omega} \left(\mathbf{q}_k(\mathbf{x}) \mathbf{q}_k(\mathbf{x}) \right) d\mathbf{x} = \sum_{k=1}^{M} \mathbf{T}_k . \qquad \mathbf{(3)}$$

Thus the summation in (3) follows all gradient fields, each computed independently for each color channel in an image. Finally, let us observe that there are two dimensions involved in (1) and (3). The first directly follows dimension of the gradients, i.e. it is 2D for single image or 3D for video sequences, and so on. The second dimension comes from a number of image channels *M* in (3). Discrete realization of the structural tensor was analyzed by Hauβecker *et al.* [7]. This is given as follows

$$\hat{T}_{ij}(\rho, \xi) = F_{\rho}(R_i^{(\xi)} R_j^{(\xi)}) , \qquad \mathbf{(4)}$$

where $R_i^{(\xi)}$ a ξ-tap discrete directional operator and F_ρ is a smoothing kernel at scale ρ [3]. For $R_i^{(\xi)}$ we used the directional filters provided by Farid *et al.* [6].

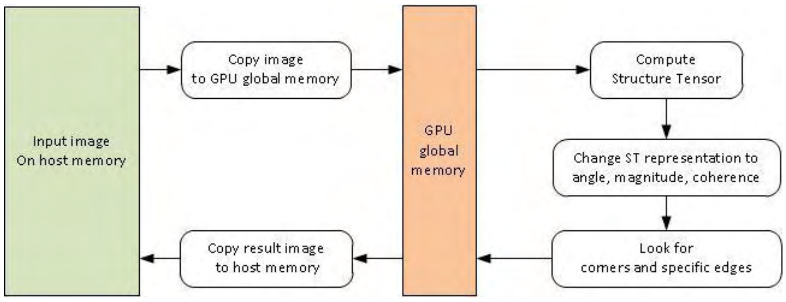


Figure 2. Processing path for corners and edges detection.

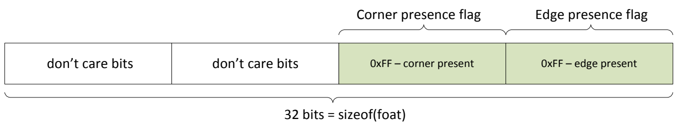


Figure 3. Pixel format of the output image.

In image processing the structural tensor is very useful in a slightly different form, i.e. computing phase and magnitude of the vector **w**. The phase of the vector **w** in (1), which corresponds to an eigenvector of the greatest eigenvalue of **T**, can be found analytically from the following representation [7]

$$\mathbf{w} = \begin{bmatrix} w_1 & w_2 \end{bmatrix}^T = \begin{bmatrix} T_{xx} - T_{yy} & 2T_{xy} \end{bmatrix}^T . \qquad \mathbf{(5)}$$

From the above the following is easily obtained

$$\varphi = \operatorname{atan2}\left(\frac{2T_{xy}}{T_{xx} - T_{yy}} \right). \qquad \mathbf{(6)}$$

This formula is directly used to find local structures in images with requested phase. In this case we need to additionally check trace of **T** which should be greater than a predefined threshold τ, that is

$$t = T_{xx} + T_{yy} > \tau . \qquad (7)$$

For corner detection the eigenvalues of **T** should be computed, as follows

$$\lambda_{1,2} = \tfrac{1}{2}\left(\left(T_{xx} + T_{yy}\right) \pm \sqrt{\left(T_{xx} - T_{yy}\right)^2 + 4T_{xy}^{\ 2}} \right). \qquad \mathbf{(8)}$$

It can be shown [7] that the analysis of a type of local pixel neighborhoods can be based on the analysis of the local eigenvalues (8). However, the two eigenvalues can be joined together in a form of the following coherence component [1]

$$c = \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \right)^2 . \qquad \mathbf{(9)}$$

The above coefficient takes on 0 for ideal isotropic areas or structures with constant intensity value, and reaches 1 for ideally directional structure.

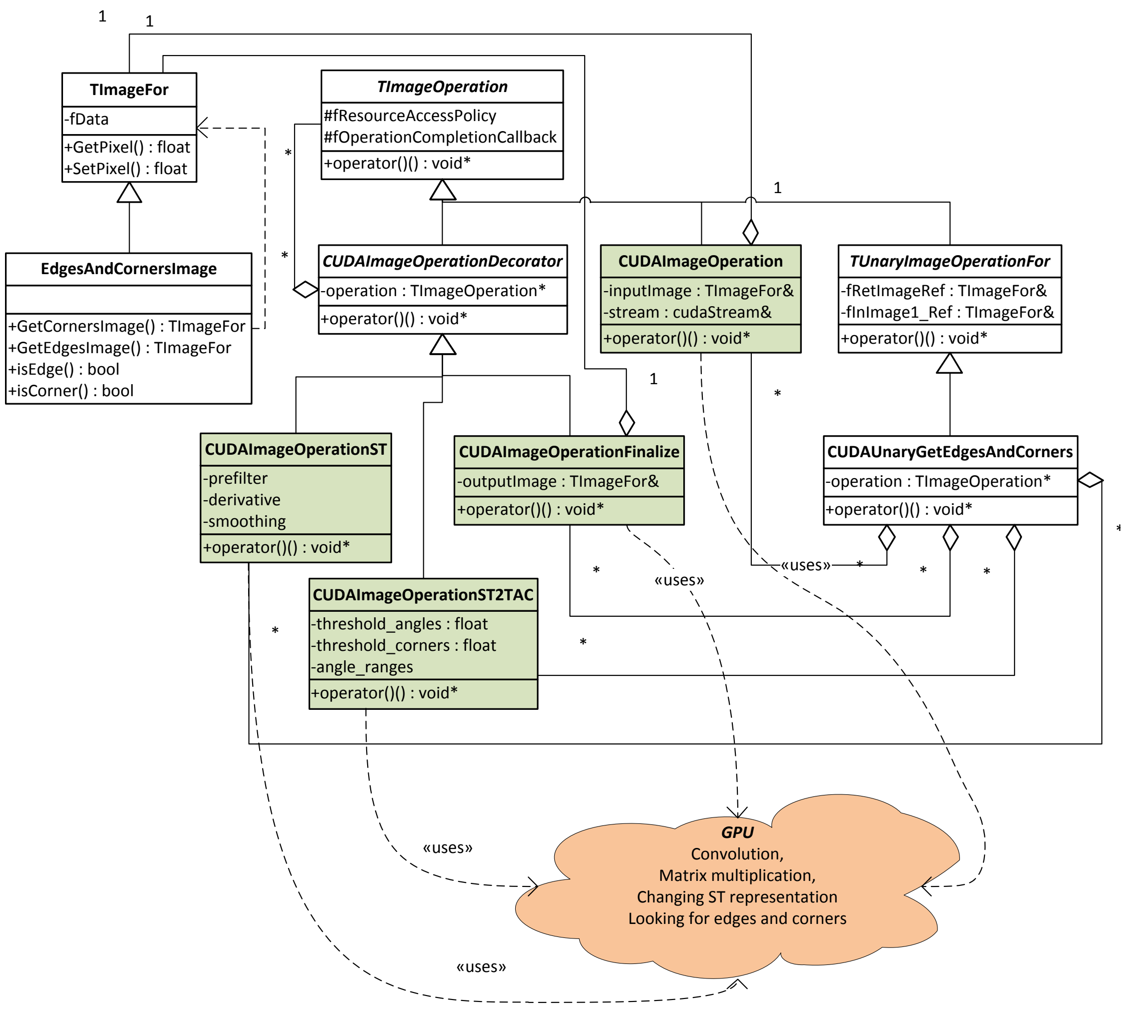


Figure 4. Architecture of the class hierarchy.

3. ARCHITECTURE OF THE IMPLEMENTATION PLATFORM

In implementation of the oriented structures in images (e.g. corners, edges, etc.) based on the aforementioned structural tensor, the CUDA graphics cards were used. The code was compiled with `nvcc` - enhanced-C compiler. HIL library was utilized as a tool for image and image operation representation [3]. Figure 2 shows the data flow for an operation of searching corners and edges. The image is first copied to the GPU global memory. Structure tensor is calculated, which requires pre-filtering of images, calculation of their derivatives, point-by-point images multiplication and their smoothing, as outlined in formula (4). In the present implementation the maximum length of each of the filters is 11.

The matrix representation of the structure tensor obtained at this stage is not too useful, so we go on a vector representation (**5**). Now following parameters are given explicitly: the angle indicating the direction of maximum change in image (**6**), the tensor trace (7) and coherence level *c* (**9**) for each pixel of the image. We are looking for image points for which the direction of maximum change imposed is within the desired range of angles, provided that the coherence is greater than the set threshold value. Minimum value of coherence *c* is particularly important because it significantly affects the accuracy. Corner detection is to find points for which the minimum eigenvalue of the structural tensor exceeds a given threshold value (**8**). The result of these operations is an image in which each pixel contains two flags. The first one indicates the presence of a corner, while the other the presence of an edge having desired pitch. The output pixel format is illustrated in Figure 3. Image in this form is copied from the GPU global memory to host memory.

In this implementation we put special emphasis on efficient GPU memory management. Indeed, it appears that the time of allocation and release of GPU memory is long and comparable with the duration of structure tensor computation. One also cannot neglect the transfer time from the CPU memory to the GPU memory. The first of the dilemmas was resolved by transferring the responsibility for memory allocation for cudaStream class instance. This class was created in our framework for the purpose of effective memory and CUDA streams management. Once the memory was allocated by `cudaStream` it is destroyed by the same stream object. During the detection of corners and edges the memory transfer takes place only twice - when copying the original image to the GPU memory and copying back the result image to the CPU memory. All operations that happen during processing allocate GPU memory using a `cudaStream` class object. Number of GPU-CPU memory transfers has been limited to a necessary minimum. The Figure 4 shows a UML diagram of the classes needed to create the corners and edges detectors. To make any calculations using the GPU one has to create an object of the class `CUDAImageOperation`. Then decorate it with `CUDAImageOperationDecorator` and its derived classes. The last decorator must be `CUDAImageOperationFinalize` class instance. Since the operation of searching corners and edges is often needed, `CUDAUnaryGetEdgesAndCorners` class was defined which provides a convenient façade for above function classes. The following code listing shows exemplary usage of this class:

```
TImageOperation * operation = new
CUDAUnaryGetEdgesAndCorners
(ProcessedImage, OriginalImage,
AngleRanges, *stream);
(*operation)();  // perform detection
```

`CUDAImageOperation` class and the classes derived from `CUDAImageOperationDecorator` use elementary GPU functions - the CUDA kernels. These are the following functors: one-dimensional convolution, matrix point-by-point multiplication, the tensor representation transformation, search engine for edges and corners, etc. The method `operator()` of these classes calls them asynchronously, instructing CUDA driver to carry out them in the near future, but not waiting for them to complete. The time between the completion of execution of the kernel function and the end of their commission is a time where the CPU is in a 'sleep' state. The potential of unused CPU computational power is supposed to be utilized in further development.

4. EXPERIMENTAL RESULTS

Performance of edges and corers detection operation has been investigated on artificial and real images. Tests were performed on two machines: a laptop Core2Duo 1.8GHz, 2GB of RAM with the graphics card GeForce8400M GS and PC computer i7 3.0GHz endowed with the graphics card GeForce Quadro FX 3800M. Achieved performance differs significantly which results from different capabilities of the graphics cards – especially number of processing cores and their clock. Throughput was obtained by measuring the time of 16 times repeated detection operation. Time was measured by CUDA timers.

The plots in Figure 5 compare the throughput of detection operations. Performing calculations using the graphics card results in at least tenfold improvement in efficiency, comparing with bare CPU computations as well as with the OpenMP parallelized version of HIL [4]. This gain increases slightly with a resolution of the input image, since operating on large images increases the profit resulting from the small CPU-GPU transfer to parallel operations (convolution, matrix multiplication) time ratio. Figure 6 is a diagram of operations performed on the graphics card while testing the edges and corners detector obtained using `computeprof v4.0`. The diagram includes a function performed on the graphics card as well as their duration. GPU work breaks during single detection are rare and short. This is a desired effect, because GPU time is not wasted. This is made possible through the use of asynchronous kernel functions and sparse synchronization of the CPU and GPU. The synchronization function is invoked only in necessary moments - when the CPU needs to use the results of calculations of the GPU.

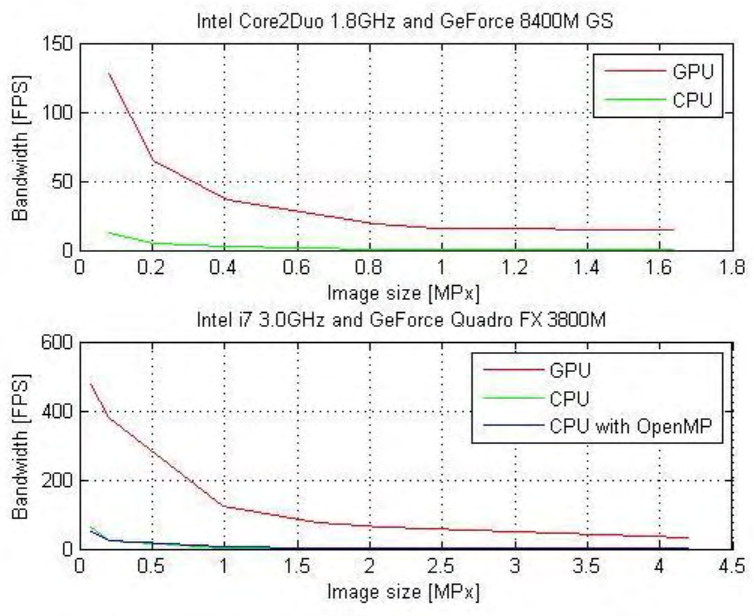


Figure 5 Achieved performance on different machines

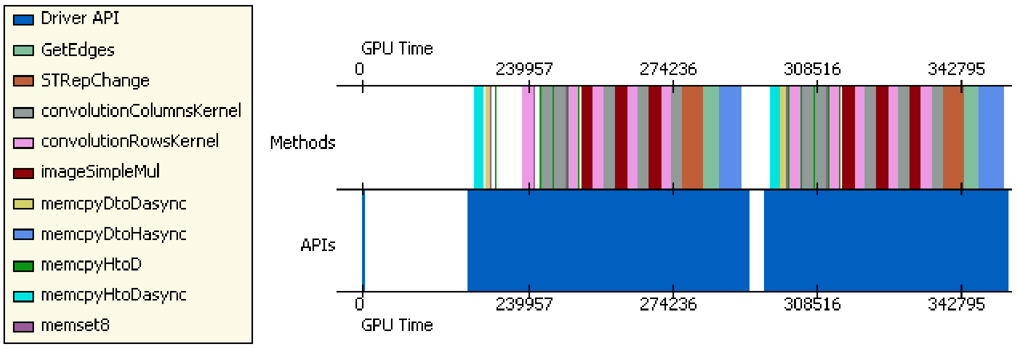


Figure 6 GPU load timing during edges and corner detection

It is worth noting that the performance of these operations should depend on length of the applied filters. In the above implementation always 11-tap filter is used. In the case of smaller length filters redundant coefficients are supplemented by zeros. Figure 7, Figure 8 and Figure 9 present detected corners and edges in computer generated test patterns as well as in a real image containing several road signs. The 3-tap filters (Simoncelli, Binomial) were used during operations. Filter details in [3].

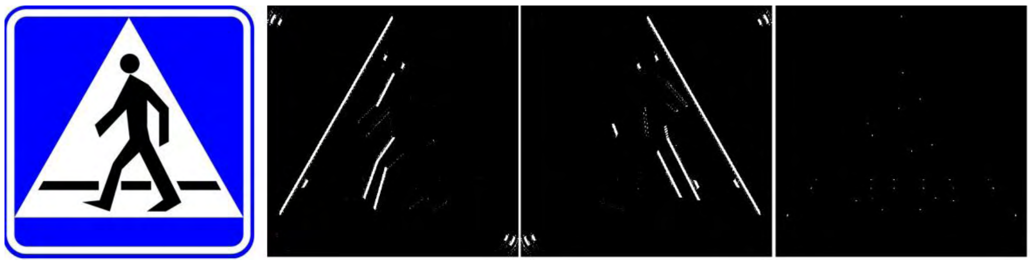


Figure 7. a) Original image b) angles in range (50, 70) c) angle in range (-50, -70) d) corners.

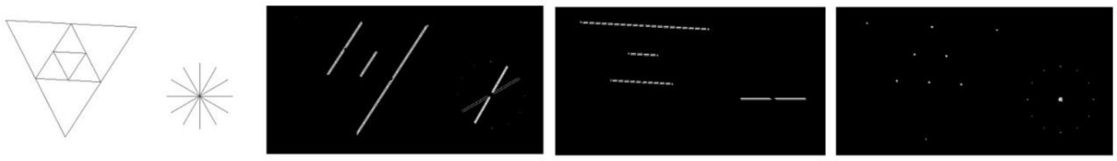


Figure 8. a) Original test image b) angles in range (50, 70) c) angles in range (0, -10) d) corners.

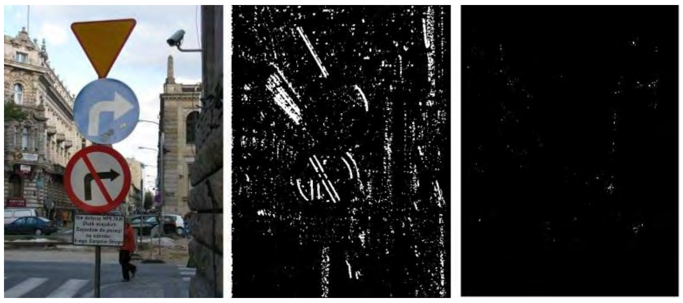


Figure 9. a) Real image b) angles in range (-50, -90) c) corners.

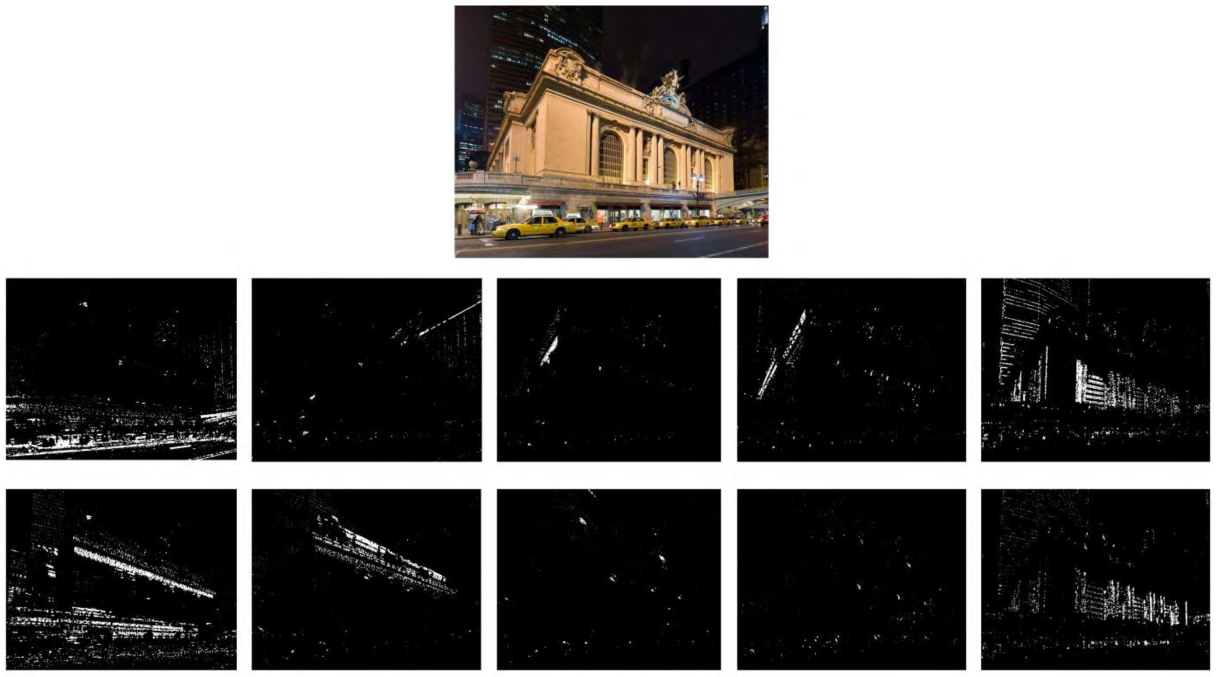


Figure 10. Pattern image and edges detection in full angle ranges

Detection in ranges (0, 20), (20, 40), (40, 80), (80, 10) in first row

Detection in ranges (-20, 0), (-40, -20), (-60, -80), (-100, -80) in second row

5. CONCLUSIONS

In this paper we present a method of detection of low-level features of specific orientation in images. These are corners and linear structures of predefined phase which allow detection of rigid objects, such as road signs, cars, etc. First the structural tensor is computed with help of the HIL library augmented with the graphic card with the CUDA environment. The presented method allows computations which in the worst case are at least an order-of-magnitude faster that an equivalent serial and multi-core software realization. This, in turn, allowed real-time operation on HDTV streams which is shown in the provided experimental results. Finally, the presented implementation was made available from the Internet and can be downloaded at [9].

6. ACKNOWLEDGMENTS

Financial support of the Polish funds for scientific research is greatly acknowledged.

7. REFERENCE

[1] Bigün, J., Granlund, G.H., Wiklund, J., Multidimensional Orientation Estimation with Applications to Texture Analysis and Optical Flow. IEEE PAMI **13**(8), (1991) 775-790

[2] Carson, C., Belonge, S., whichGreenspan, H., Malik, J., Blobworld: Image Segmentation Using Expectation-Maximization. IEEE PAMI **24**(8), (2002) 1026-1038

[3] Cyganek, B., Siebert J.P.: An Introduction to 3D Computer Vision Techniques and Algorithms, Wiley (2009)

[4] Cyganek, B.: Adding Parallelism to the Hybrid Image Processing Library in Multi-Threading and Multi-Core Systems. 2nd IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA 2011), Perth, Australia, 2011

[5] Di Zenzo S., A note on the gradient of a multi-image. Computer Vision, Graphics and Image Processing, **33**: (1986) 116-125

[6] Farid, H., Simoncelli, E.P., Differentiation of discrete multidimensional signals. IEEE Trans. Image Proc. **13**(4) (2004) 496-508

[7] Hauβecker, H., Jähne, B., A Tensor Approach for Local Structure Analysis in Multi-Dimensional Images. Technical Report, University of Heidelberg (1998)

[8] http://www.wiley.com/legacy/wileychi/cyganek3dcomputer/supp/HIL_Manual_01.pdf

[9] http://student.agh.edu.pl/~mbugaj/Detector/Detector.rar

[10] NVIDIA, CUDA C Programming Guide (2011)

[11] Sanders, J., Kandrot, E., CUDA by example: an introduction to general-purpose GPU Programming, Addison-Wesley (2011)

[12] NVIDIA, CUDA Toolkit Reference Manual, (2011)

Detecting and Removing Islands in Graphics-Rendering-Based Computations of Lower Envelopes of Plane Slabs

Kamran Safdar
Fachbereich Computerwissenschaften
Universität Salzburg, A-5020 Salzburg, Austria
ksafdar@cosy.sbg.ac.at

ABSTRACT

Geometric algorithms which make use of graphics rendering often require manipulation and adaption of the pixel map of the lower envelope of plane slabs. The complex interaction of the slab geometries may give rise to isolated portions in the pixel map ("islands") which need to be discarded and patched appropriately. Such problems may occur, for instance, when attempting to compute multiplicatively-weighted Voronoi diagrams or straight skeletons in 2D by means of graphics rendering of the lower envelope of plane slabs in 3D. This paper presents general algorithms for detection, labeling, and removal of islands in an input pixel map. Removal, here, means recovery of the correct portions of the pixel map in lieu of the islands. The presented island detection algorithm requires only a constant number of passes over the input pixel map without any dependence on the number of input sites being processed by the geometric algorithm. This paper introduces the concept of black lists for the removal of islands and explains how the presented approach can cope with stacked islands, with no need for looping over the stack of islands for recovery of the correct pixel map underneath it. The discussion is concluded by experimental results obtained with the implementation of the presented algorithms.

Keywords: black list, FILM, GPGPU, islands, lower envelope, pixel map, plane slabs, rendering-based computation, SIR, STIR, straight skeleton, weighted Voronoi diagram.

1 INTRODUCTION

1.1 Motivation

Rendering specially crafted distance functions or slab geometries and projecting their lower envelope is a well-known approach for obtaining graphics-rendering-based solutions of geometric problems. For instance, the use of graphics hardware for computing discretized versions of Voronoi diagrams of point sites in 2D was shown already several years ago in early versions of the OpenGL Programming Guide [WND97]. Roughly, the Voronoi diagram of a set of point sites in 2D is a partition of the plane into individual regions, the so-called Voronoi regions, with exactly one region per site, such that every region is given by the loci of points closer to its defining site than to any of the other point sites. At every input point p an upright circular cone is constructed above the xy-plane such that its rotation axis is parallel to the z-axis and such that its apex coincides with the input point. If the envelope of the cone forms an angle of 45^o with the xy-plane then the set of all points with distance r from p can be identified with the intersection of the cone with the plane $z = r$. If every cone is colored uniquely then a discretized Voronoi diagram of the input points can be obtained by rendering the lower envelope of all cones via a parallel projection in the direction of $+z$, which is readily accomplished on a modern graphics processing unit (GPU).

Hoff et al. [HCK+99] extended this concept to discretized Voronoi diagrams of more general primitives in 2D and 3D. We give reference to [Hae90, Den03b, Den03a, Yam05b, Yam05a, CT05, VR05, FG06, LZC09] for other illustrative examples of how the fast rendering and interpolation capabilities of a GPU have been employed to solve various problems of a geometric nature.

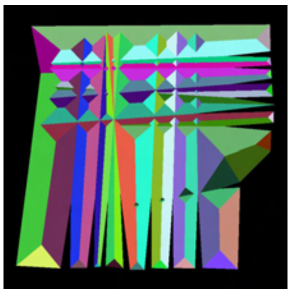
Generally speaking, in all such rendering-based solutions every input site is assigned a unique color identity which is used as the color of the slab drawn for this site. Of-course, the geometries are designed in a way that they leave no portion of the scene uncovered thus constituting a correct diagram of the geometric structure being discretized. Moreover, all such solutions rely on the parallel projection of the lower envelope of these slab geometries. The mutual interactions of the geometries often is very complex as the slabs may penetrate each other at multiple locations. From a computational point of view a problem occurs when a slab geometry reappears, in the projected image of the scene, after penetrating somewhere. These reappearing portions of the slab geometries project to areas in the pixel map which we call *islands*. In simple terms, an island is a set of pixels which bear the same color identity as some site but lie isolated from that site in the image. Formally, we define an island pixel as a pixel from which the projection of the corresponding site cannot be reached by traversing only identically colored neighbors.

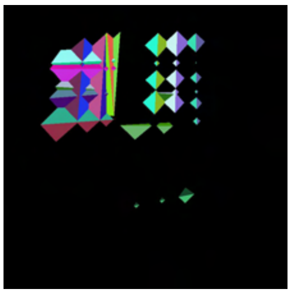
While the standard Voronoi diagram of a set of point sites in 2D is indeed given precisely by the lower envelope of a set of cones in 3D and, thus, can be obtained easily by graphics rendering, this approach is not directly applicable to multiplicatively-weighted Voronoi diagrams: For multiplicatively-weighted Voronoi diagrams one has to employ cones with different inclinations, which implies that the Voronoi diagram may no longer corresponds to the projection of the lower envelope of the cones to the xy-plane. Rather, the complex interaction of the cone geometries may give rise to isolated portions in the pixel map ("islands") that may need to be taken care of appropriately. Similar problems arise when one attempts to compute the straight skeleton of a simple polygon by means of graphics rendering of the lower envelope of certain plane slabs.

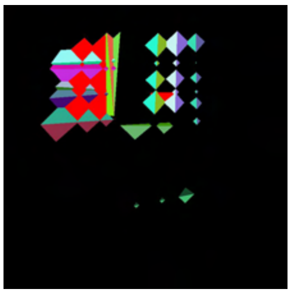
It is important to know here that a few of the isolated portions may not actually be islands, but true features of the geometric structure being approximated. Such islands are referred to as *false islands*. Existence of false islands is understandable as for example, a multiplicatively-weighted Voronoi diagram can actually have disjoint Voronoi regions. Or, while approximating a straight skeleton, false islands may appear in the pixel map because of certain true islands isolating them from their corresponding slabs and all of them collectively forming a large complex island. Removal of the "island" tags from such false-islands purely depends on the geometric structure being approximated.

The false islands which appear while computation of the discrete straight skeleton are efficiently dealt by the proposed "stir" algorithm as it utilizes the outline of an island for recovery of the correct portions of the diagram underneath it. And, in case of a false island, the site corresponding to the false island participates in forming the outline of the complex island containing this false island. The reason for this is that a false island is actually a portion of the correct diagram and, therefore, its leading part exists in the outline of the complex island which contains it. The removal of this complex island is carried out in a way that the false island portions remain intact and, hence, the actual diagram is recovered. Figure 1 illustrates an example of such false islands (highlighted in red) while sketching the straight skeleton of a simple polygon.

In case of a multiplicatively-weighted Voronoi diagram, a false island pixel may be dealt by appropriately comparing its distance to the corresponding site and its distance to the sites which form the outline of the complex island containing this false island. The depth value of each pixel, here, is proportional to the distance of this pixel to its corresponding site in terms of its weight.


(a) Original pixel map.


(b) Detected islands.


(c) False islands (red).

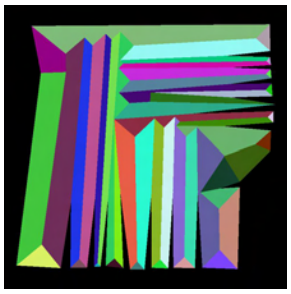

(d) Recovered sketch of straight skeleton.

Figure 1: False islands encountered while approximating straight skeleton of a simple polygon.

1.2 Main Contribution

The identification and, more importantly, removal of islands is a prerequisite for being able to extend rendering-based computations to geometric structures where the normal projection of the lower envelope of a suitably defined set of plane slabs does not suffice to reveal a discretized approximation of the structure. This paper highlights this problem and presents remedies for its resolution.

The main contributions of the presented work are:

1. A robust algorithm for the detection of islands in the rendering of a lower envelope of plane slabs.
2. Introduction of the concepts of black lists, and outlines of islands for recovering correct portions of the geometric structure hidden underneath the islands.
3. An efficient approach to deal with multiply stacked islands.

To put the problem of island detection and removal in such a pixel map in an actual geometric context, we briefly discuss the construction of a multiplicatively-weighted discretized Voronoi diagram of point sites using the cones method (by Hoff et al. [HCK+99]), and our own graphics-rendering-based technique for computing a discretized straight skeleton of a simple polygon.

1.3 Witnessing islands while constructing weighted discrete Voronoi diagrams

This discussion refers to the well-known cones method [HCK+99] for the construction of discretized Voronoi diagrams in 2D; the authors claimed that it is easily extendable to multiplicatively-weighted Voronoi diagrams. For simplicity, let us suppose that the input contains only point sites. The distance functions, cones in this case, are multiplicatively weighted [OBS92] and thus have variable angles. Furthermore, consider the case as illustrated in Figure 2 in which sites a, b, and d, have the same weight, however, site c has a higher weight and thus a higher base angle of the corresponding cone drawn over it. Due to this wider angle, this cone at site c (shown in green color in figs. 2 to 4) is responsible for three islands in the cross-sectional area under consideration. These islands are illustrated by dash-dotted-green lines in fig. 2. The dashed lines represent the non-visible sides of the cones. The dotted-red, dotted-blue, and dotted-purple portions, in the same figure, represent the actually correct but hidden parts of the diagram due to islands.

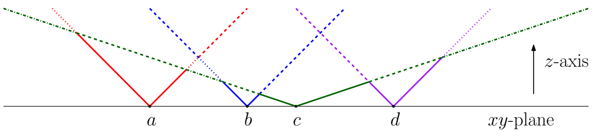


Figure 2: Islands arising due to site c.

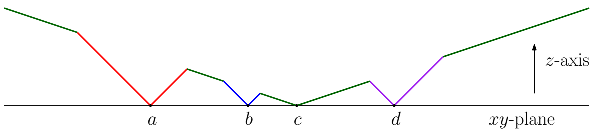


Figure 3: Visible lower envelope containing islands.

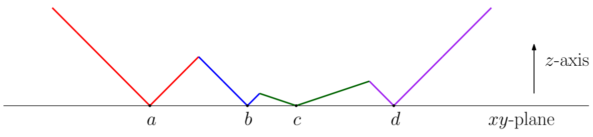
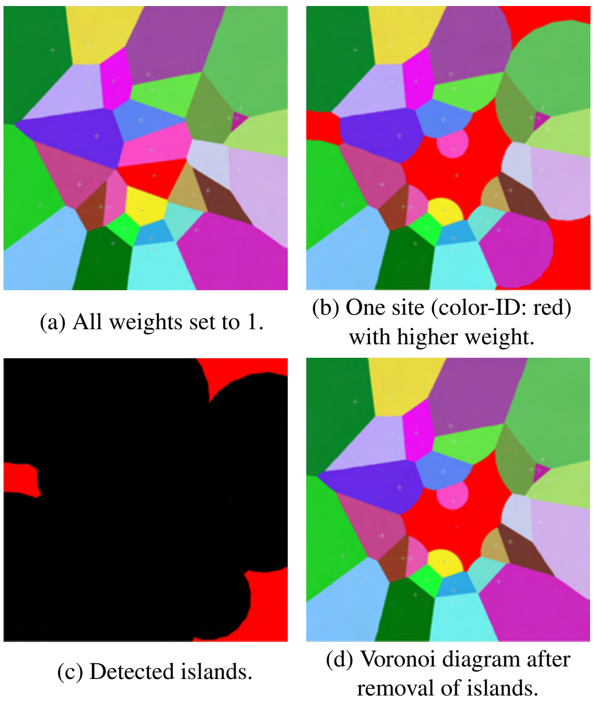


Figure 4: Lower envelope after removal of islands.

Similarly, if various sites carry different weights many islands may pop-up in the rendered pixel map which need to be taken care of. Figure 5 illustrates another example of islands experienced in a pixel map while computing a multiplicatively weighted discrete Voronoi diagram.



(a) All weights set to 1.

(b) One site (color-ID: red) with higher weight.

(c) Detected islands.

(d) Voronoi diagram after removal of islands.

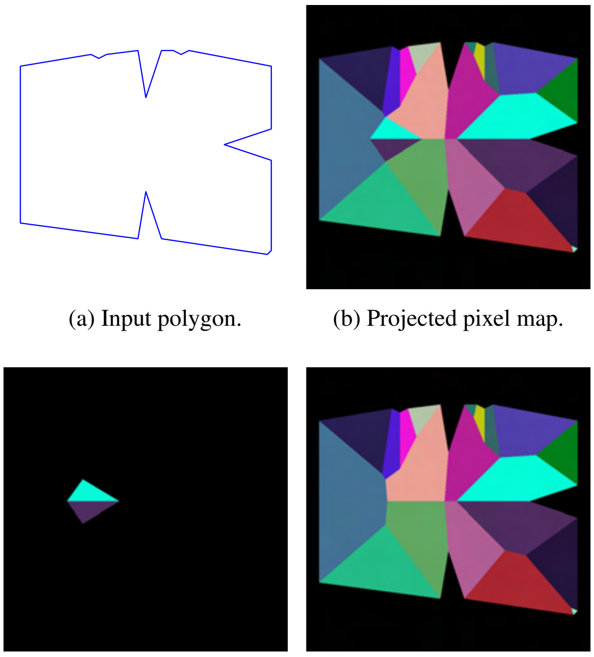
Figure 5: An example of islands appearing in the projection of the lower envelope of weighted distance functions while computation of discrete Voronoi diagram.

1.4 Experiencing islands while approximating straight skeleton of a polygon

Skeletons, being important structures for representation of 2D objects based on their topological characteristics, are of much interest in many geometry applications. The term “straight skeleton” was first tossed by Aichholzer et al. in 1995 [AAAG95], who are also the first ones to give an algorithm for computing the straight skeletons for interior of simple polygons. The straight skeleton of a polygon is defined by the set of lines traced out by its vertices during a shrinking process in which the edges of the polygon are moved inwards towards its interior, such that, at any instance of time, every shrinking edge is parallel to the original edge and the orthogonal distance between every shrinking and the corresponding original edge is the same. While this shrinking process, if any vertex passes over a non-adjacent edge, the polygon is split into two and the shrinking process is continued in each individually. The lines traced are known as *arcs* and their endpoints which are not vertices of the polygon are known as *nodes*. The arcs, the nodes, and the vertices of the polygon collectively define a graph embedded in the interior of the polygon which is its *straight skeleton*.

A straight skeleton is composed of only straight line segments and is closely related to, but not same as, the Voronoi-based medial axis offsetting scheme as the latter may contain circular arcs in addition to the straight lines. And, this fact makes straight skeletons advantageous over any other skeleton type as processing straight lines turns out to be less complex than handling some curved constructs.

Suppose we are given, as input, the n vertices of a simple polygon $\mathscr{P}$. Assuming $\mathscr{P}$ to lie on the $z = 0$ plane of the 3D euclidean space and using the general convention of assuming height of a point to be its z-value, certain slabs are drawn over each vertex. Each of these n vertices is assigned a unique color identity which is used as the color for its corresponding slab. The exterior of $\mathscr{P}$ contains background color of the scene since all n geometries grow from the edges towards the interior of $\mathscr{P}$. Moreover, the mutual interaction of these slabs is extremely complex as they penetrate each other at multiple locations. Hence, islands may exist in the parallel projection of the lower envelope of these slabs. These islands must be removed to achieve an accurate sketch of the straight skeleton of $\mathscr{P}$. Figure 6 shows an example island occurring due to the interaction of slabs constructed over the vertices of an input polygon.



(a) Input polygon. (b) Projected pixel map.

(c) Detected island. (d) Straight skeleton sketch.

Figure 6: An example of island in the projection of the lower envelope of slab geometries.

2 DETECTION AND REMOVAL OF ISLANDS – A GENERAL APPROACH

The approach is to process the parallel projection of the lower envelope of slabs, which are drawn by the employed geometric structure approximation technique, by detecting, uniquely identifying, and removing all islands in it. The projection is saved as a color pixel map and island detection is carried out on it as the first step. If any islands are encountered then the island labeling algorithm is invoked which labels every island making it uniquely identifiable. Subsequent to this, the pixel map is fed to the island removal procedure which generates a correct diagram patch for every island, and appropriately stitches these patches in the pixel map in place of the corresponding islands. If no islands are detected in the first step then this indicates that the output of the employed geometric technique is already correct and requires no further processing.

2.1 Detecting the islands

To detect islands in the projected pixel map, an island-boolean-map having resolution the same as that of the pixel map is maintained. The approach is to mark all pixels as island pixels first and then start changing their boolean flags based on the colors and flags of their neighborhood. Here, it is assumed that the input sites are projected adjacent to the background color of the scene (as in case of a polygon with background color on its exterior). Or, if this is not the case then the sites are projected in background color while still keeping the color identities of their corresponding slabs intact. In the latter case, the projections of sites are recovered back to their original colors later after island detection.

A pixel is marked as a non-island pixel only if its color is the background color of the scene, or if any of its neighboring pixels has the background color, or if any of its neighboring pixels has a color same as the pixel under question and that neighbor has already been marked as a non-island pixel. This leads to the island detection algorithm which iterates over the whole pixel map from one end to the other updating the pixel flags and leaving behind the actual islands. This continues until no pixel flag is updated during a pass of the pixel map.

The assumption of sites being projected in, or adjacent to, background color not only guarantees the generality of island detection but also ensures the island detection to be fail-safe. This is because, the islands are assumed as not to have any connectivity to their corresponding sites via their identically colored neighboring pixels and that the islands are detected based on their relationship to the background of the scene. Moreover, islands may be classified into two kinds based on their relationship to the sites responsible for their occurrence. If an island is due to only one site, it is called a *simple island*. And, if it is due to more than one sites then it is referred as a *complex island*. Islands due to different sites but lying immediately adjacent to each other are regarded as a single complex island. The sites responsible for existence of islands are called *bad sites*. In other words, a bad site is the one a portion of whose corresponding slab is projected as an island in the pixel map.

2.1.1 *Four-neighbors versus Eight-neighbors testing*

Defining the term *neighbor* is of crucial importance for accurate island detection. Two candidate schemes, namely four-neighbors and eight-neighbors are the natural candidates for adoption. Considering only four neighbors (left, right, top and bottom) of a pixel to be its actual "neighbors" wins over the the other alternate in cases where some pixels of an island have diagonal connectivity to the non-island portions of their respective slabs. These island pixels are actually part of the island but are missed by the eight-neighbor-testing scheme as it assumes them as connected to their corresponding slabs and marks them as non-islands. However, the four-neighbor-testing is able to detect such pixels correctly and marks them as islands since it does not perform any diagonal connectivity checks. For an illustration of this fact, see Figure 7 which is taken from one of the experiments approximating the straight skeleton of a polygon.

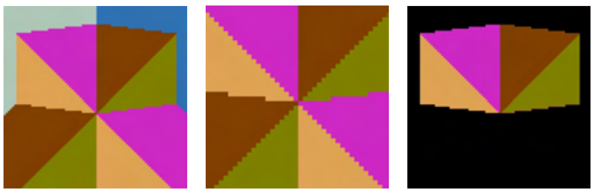

(a) Portion of pixel map. (b) A zoomed portion of fig(a). (c) Detected island.

Figure 7: An island detected using four-neighbor connectivity testing.

Thus, eight-neighbors testing is not a good choice for adoption regardless of its early connectivity detection as compared to its counterpart. This leaves us with the four-neighbors testing scheme which works perfectly as it is capable of handling the cases which the eight-neighbors testing scheme fails to take care of.

The four-neighbors testing mechanism, as a drawback, requires an extra pass over the input pixel map for removal of single pixel thick areas which are actually not islands but are marked as islands due to the lack of diagonal-connectivity testing. These areas may occur as diagonally connected strips of single pixels. For an example, see Figure 8 which is taken from one of the experiments approximating the straight skeleton of a polygon.

2.1.2 *Orders of processing the pixel map*

The order of processing the pixel map in subsequent iterations of the island detection algorithm is extremely important as it can greatly effect the overall number of iterations required to accurately detect the islands.

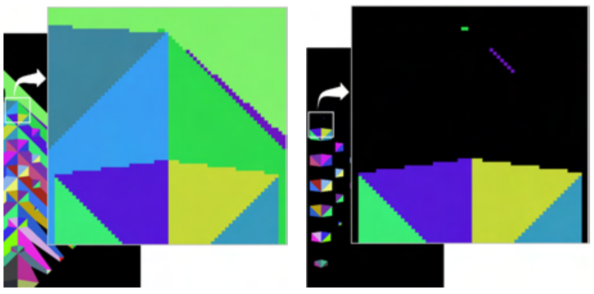

(a) Portion of the original pixel map. (b) Detected islands.

Figure 8: Single pixel thick islands detected using the four-neighbor testing scheme without an extra pass.

Moreover, since immediate neighbors of every pixel are to be analyzed, the island detection algorithm should process the pixel map within the bounding box (second pixel in the second row –to– second-last pixel in the second-last row).

Let us consider the processing of a pixel map starting from the lower left corner of the box area inside it and moving row by row towards the top right corner. All non-island pixels which are direct neighbors of the background color, or ones which have a connectivity to it via their left or bottom neighbors will be correctly marked in the very first pass of the pixel map. For each subsequent pass, a one-pixel thick layer of the not yet marked non-island pixels, adjacent to already marked non-island pixels, will pass the non-island-pixel test, thus, leaving behind the actual island pixels. Similarly, if the pixel map is processed from the top right corner to the bottom left corner, a large number of non-island pixels having connectivities to the background color via their top or right neighbors pass the non-island test in the very first iteration.

Thus, altering the order of processing the pixel map at every subsequent iteration greatly boosts the detection of islands. To achieve maximum performance, we apply a four-way processing of the pixel map. That is, processing in the following order: First Iteration – bottom-left-to-top-right, Second Iteration – top-right-to-bottom-left, Third Iteration – bottom-right-to-top-left, Fourth Iteration – top-left-to-bottom-right; and then repeating this order for any subsequent iterations.

Experiments on various datasets using different processing orders (one-way, two-way, four-way) prove the four-way-four-neighbor testing to be the best choice. For details, please see section 3. Two sample runs of the island detection algorithm using four-way-four-neighbor testing are illustrated in Figure 9.

2.2 Labeling the islands

One of the island removal techniques presented in this paper requires unique identification of every island in

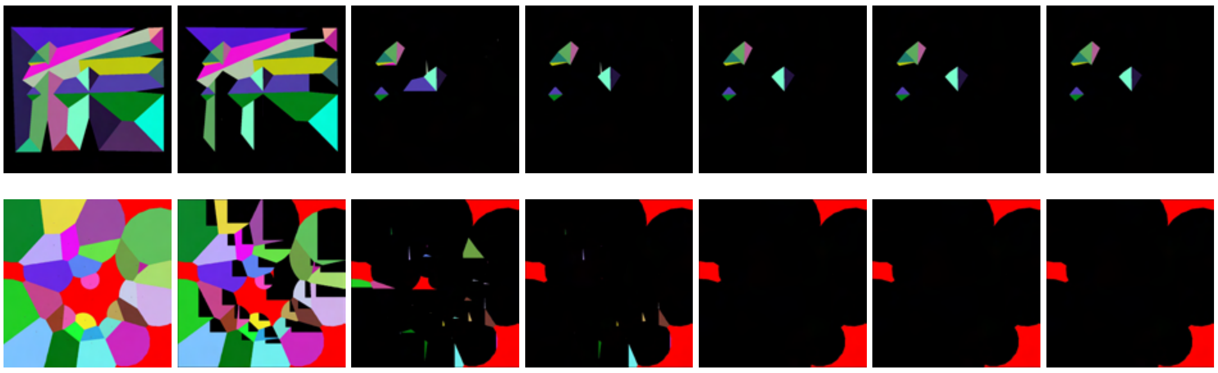
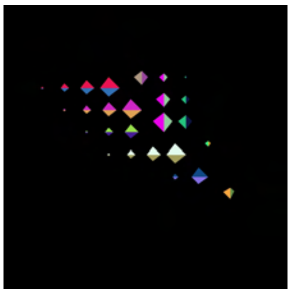


Figure 9: Progress of each iteration (left-to-right) of the four-way-four-neighbor testing for island detection in example pixel maps (left most in both rows) projected during approximation of the straight skeleton of a simple polygon having twenty vertices (first row), and the weighted Voronoi diagram of thirty randomly generated point sites (second row). The right most image in both rows contains the detected islands.

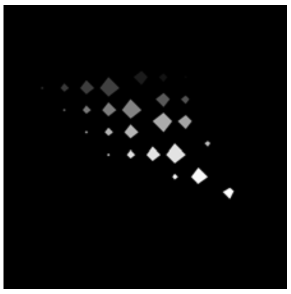
the pixel map thus laying the basis of the basic requirement for an island labeling algorithm. Moreover, in some later parts of the geometric solution being computed, the overall technique may also require information relating some particular island(s) for which we must be able to uniquely identify and distinguish it/them.

Let us assume that island detection has already been applied to the input pixel map and some islands do exist which need to be labeled. The island labeling algorithm maintains an island-label-map having resolution the same as that of the pixel map. It begins by initializing the island-label-map to all zeros – zero is regarded as the non-island label. The algorithm parses the island-boolean-map and considers only island pixels. For every such pixel, it generates a new label and assigns it to this pixel by recording this label at the pixel's position in the island-label-map. Along with this, it compares the labels of the neighboring pixels. The eight-neighbor scheme is adopted here for defining the term "neighbor". This boosts the performance as the connectivity information does not effect a pixel's island flag and, rather, lets the determination of labels of the diagonal neighbors beforehand. Now, if a neighbor of the current pixel has a non-zero label, and this label is smaller than that of the current pixel, then the algorithm sets the current pixel's label same as that of the neighbor pixel just tested.

Looping the island-boolean-map and the island-label-map with this simple testing scheme assigns unique labels to all islands. Thus, the underlying idea is similar to the standard connected component labeling techniques, that is, to assign a temporary label to each island pixel and update it depending on whether its immediate neighbor pixels also belong to the same island. As an example of the labeling output, Figure 10 illustrates an island map and the corresponding island-label-map with labels converted to the levels of gray.



(a) Sample island-map.



(b) The corresponding island-label-map.

Figure 10: An example grid of island labels with labels converted to the levels of gray.

Similar to the island detection, different orders of processing of the island-label-map can be adopted for labeling the islands. Application of a two-way processing of the pixel map ensures a generally speedy approach, since the eight neighbor scheme is employed here.

2.3 Removing the islands

Removal of islands is the most important step towards computation of the geometric structure being approximated. This section presents two island removal algorithms which can be adopted depending upon the types of the islands detected, and the requirements of the problem being addressed. If no complex island is detected in the pixel map then the first algorithm is used which operates per bad site and hence has a worst case complexity of the number of input sites (n) times the resolution of the input pixel map. This algorithm is named *sir* abbreviating "Simple Island Removal".

In worst scenarios, if more than one bad sites exist, it is not unlikely to have islands which are stacked over each other in a way that the top islands hide the ones underneath them. In such cases, removal of the visible islands in the pixel map may result in popping up of new islands in their places thus leaving the problem of island removal unresolved. The second island removal algorithm is capable of handling such stacked islands without requiring any extra looping over each stack of islands for their removal, hence, keeping the overall worst case complexity under controlled bounds. This variant of island removal is referred to as *stir* abbreviating "Stacked Island Removal". Thus, if removal of stacked islands is essential, or if any complex islands are encountered in the pixel map, then "stir" is used which operates per island and has a higher theoretical worst case complexity than "sir".

Both algorithms, "sir" and "stir", rely on a simple data structure which is called a *black list*. As the name suggests, it is a list of sites whose corresponding geometries are temporarily banned from being displayed in the scene based on certain criteria. The criteria, of-course, relates to involvement in creation of islands. In short, a black list is a sorted list of unique barred bad-sites. Moreover, both of these algorithms rely on the basic philosophy of patching the correct portions of the diagram in place of the islands. Temporarily black listing a bad site makes its geometry to be shutoff thus also shutting-off the islands due to it, and making the diagram underneath them visible on the lower envelope. These correct portions are patched in place of the islands in pixel map. The patch operations have been implemented as a parse of the pixel map while updating pixels relating to the island being processed. One could restrict the processing of entire pixel map in both "sir" and "stir" to bounding boxes of the islands being removed. This, however, does not really pay-off in real practice because of the overheads involved in computing and saving these bounding boxes.

2.3.1 Simple island removal – sir

The "sir" simply iterates on all bad sites, shutting-off their corresponding slabs one-by-one, and recovering the portions of diagram underneath the islands due to each of them. The worst case complexity of "sir" is thus $O(r \cdot w \cdot h)$, where r is the number of bad sites ($r < n$), w is the width, and h is the height of the pixel map. Considering $m = w \cdot h$ to be the resolution of the pixel map, the worst case time complexity bounds to $O(r \cdot m)$. The pseudo-code of "sir" is outlined in algorithm 1.

2.3.2 Stacked island removal – stir

It is straightforward to see that the slabs projected immediately adjacent to an island are the ones which actually constitute the correct portion of the diagram underneath it. Thus, black listing all bad sites except those participating in forming the *outline* of an island removes any possibility of more islands being stacked underneath it. The "stir" relies on this concept and utilizes the outline information for providing a general approach to remove stacked and complex islands. The pseudo-code of "stir" is outlined in algorithm 2.

```
Require: $pixelMap[\ ]$ = the pixel map to process;
    $islandBoolMap[i] = true$ if pixel $i$ is an island pixel;
    $noOfIslands$ = total number of islands encountered;
    $noOfBadSites$ = total number of bad sites.
Ensure: $pixelMap[\ ]$ = correct pixel map with no islands
1:  if ( $noOfIslands > 0$ ) then
2:      for $badSite = 1 \to noOfBadSites$ do
3:          $blackList \leftarrow badSite$, and re-render the scene
4:          $tempPixelMap \leftarrow framebuffer$
5:          for $y = 1 \to imageHeight$ do
6:              for $x = 1 \to imageWidth$ do
7:                  $pix = (x, y)$
8:                  if ( $(islandBoolMap[pix] = true)$ AND
                    $(pixelMap[pix] = colorID(badSite))$ ) then
9:                      $pixelMap[pix] \leftarrow tempPixelMap[pix]$
10:                     $islandBoolMap[pix] \leftarrow false$
11:         $blackList \leftarrow NULL$
```
Algorithm 1: The core *sir* algorithm

```
Require: $pixelMap[\ ]$ = the pixel map to process;
    $islandBoolMap[i] = true$ if pixel $i$ is an island pixel;
    $islandLabelMap[i] > 0$ if pixel $i \in$ island "$islandLabelMap[i]$";
    $noOfIslands$ = total number of islands encountered;
    $noOfBadSites$ = total number of bad sites;
    $island.outline$ = set of bad sites present in the outline of $island$;
    $island.label$ = the unique label of $island$;
Ensure: $pixelMap[\ ]$ = correct pixel map with no islands
1:  if ( $noOfIslands > 0$ ) then
2:      for $island = 1 \to noOfIslands$ do
3:          for $badSite = 1 \to noOfBadSites$ do
4:              if ( $badSite \notin island.outline$ ) then
5:                  $blackList \leftarrow badSite$
6:          re-render the scene
7:          $tempPixelMap \leftarrow framebuffer$
8:          for $y = 1 \to imageHeight$ do
9:              for $x = 1 \to imageWidth$ do
10:                 $pix = (x, y)$
11:                 if ( $islandLabelMap[pix] = island.label$ ) then
12:                     $pixelMap[pix] \leftarrow tempPixelMap[pix]$
13:                     $islandBoolMap[pix] \leftarrow false$
14:                     $islandLabelMap[pix] \leftarrow 0$
15:         $blackList \leftarrow NULL$
```
Algorithm 2: The core *stir* algorithm

The worst case complexity of "stir" is $O(i \cdot (r + w \cdot h))$, where i is the number of islands processed, r is the number of bad sites ($r < n$), w is the width, and h is the height of the pixel map. Considering $m = w \cdot h$ to be the resolution of the pixel map, the worst case time complexity bounds to $O(i \cdot (r + m))$.

Experiments prove that adopting "stir" in place of "sir" actually boosts performance when the number of input sites is fairly large. This happens because the graphics card has to render lesser number of geometries in each iteration as compared to rendering all geometries except one every time. Hence, the algorithm which appears theoretically worse, performs actually better on large datasets. Additionally, this adoption ensures removal of stacked islands as a positive side effect.

2.3.3 Handling dependency chains of islands

It is interesting to note that an island may exist due to the existence of another island in the pixel map. And this may also be true for that source island. Thus, it is possible for a pixel map to contain chains of dependencies of islands. Hence, it may become necessary to remove some other islands before a certain island is removed. An example of a dependency chain of islands is illustrated in Figure 11. Some islands in such a chain are actually the true features ("false" islands) of the diagram being computed and appear isolated because of the other "true" islands in the chain which isolate them from their corresponding slabs. A straightforward application of "sir" in such cases may not give the desired results. Moreover, the "stir" algorithm may require more than one runs in such cases. As in the first run, it will remove the root of this chain, and in the subsequent runs the reset of the chain will be processed. Furthermore, if the dependency chain forms a cycle then the "stir" requires a run of the "sir" algorithm, prior to its application, to remove one of the islands in the chain in order to break the cycle.

Such situations can be avoided by marking the "false" islands, which form links in these chains, as non-islands prior to the application of any island removal algorithm. We name this process as *film* abbreviating "False Island Marking". Hence, the application of "film" before removing islands lets us efficiently handle the dependency chains of islands. Here we use the term *participants* to refer to the individual islands constituting the complex island which is a dependency chain.

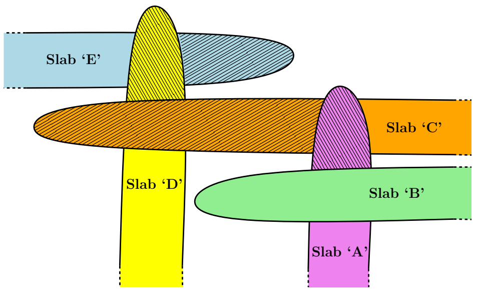


Figure 11: An illustration of a dependency chain of islands with true islands shown in falling tiling pattern, and false islands highlighted in rising tiling pattern.

False island marking – *film* All participants of a complex island are, by default, assumed to be "true" islands. The algorithm processes all complex islands in the pixel map and classifies their participants as *possibly-false*, *false*, and *true*. The classification is performed based on the relationship of a participant to the outline of the complex island being processed. First, all participants of the complex island being processed are analyzed. If the color of a participant is same as that of an outline constituent then that participant is marked as "possibly-false". As a next step, the outline adjacent to every "true" participant is checked for existence of a constituent having color same as a "possibly-false" participant of this complex island. If found then that "possibly-false" participant is marked as a "false" participant. This is done because when this "true" participant will be removed then the participant which is now marked as "false" will actually get connected to its corresponding slab thus forming a portion of the diagram being processed (for instance, see the false island corresponding to slab 'C' in Figure 11). Following this, the outline information of this newly marked "false" participant is checked for existence of a constituent having color same as that of any other "possibly-false" participant of this complex island. If found then that participant is marked as a "true" participant. This is because, as this newly marked "false" participant is to stay in the diagram, it will isolate this newly marked "true" participant from its corresponding slab (for example, see the island portion corresponding to slab 'D' in Figure 11). If all participants of a complex island are marked as "false" then all of them are actually "true" participants and must be handled accordingly. Finally, the boolean flags and labels of all "false" participant pixels are appropriately updated.

The application of "film" provides three benefits: First, it breaks the cycles in the dependency chains of islands. Second, it ensures the accuracy and integrity of the diagram. And third, it reduces the count of participants constituting the complex island, hence, reducing the number of pixels to be processed for its removal.

2.4 Salient features of the algorithms

Generality For detection and labeling of islands, no assumption is made with respect to the input primitives as these algorithms purely operate on a discrete grid which does not grow or shrink depending upon any factors. The island detection and removal approach is generally applicable to all graphics rendering based algorithms as islands are one of the major challenges towards their completeness, and rendering slabs and projecting their lower envelope is a basic feature of all such techniques.

Simplicity These algorithms can be easily implemented on the available graphics systems as from/to GPU data transfer functionalities are vastly supported. Moreover, these techniques do not have any special case arising which requires special handling.

Robustness Our algorithms use the features of the projected pixel map to efficiently handle highly sophisticated problems due to islands. For instance, the "stir" algorithm is capable of handling stacked islands without requiring any extra looping over each stack thus giving highly efficient throughput.

3 RESULTS

The statistics discussed in this section have been recorded while conduct of our experiments using a standard PC with Intel Core i7-2600 CPU clocked at

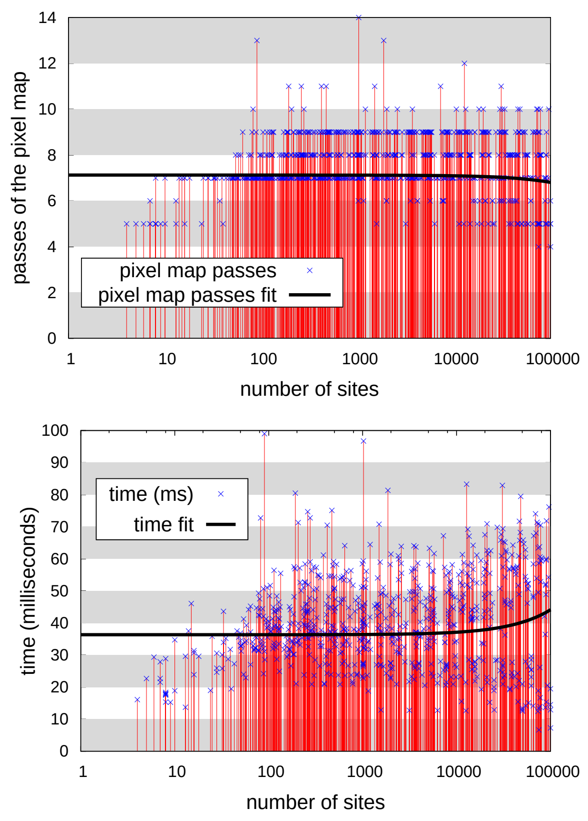


Figure 12: Plots of data relating sample runs of island detection using the four-way-four-neighbor testing scheme.

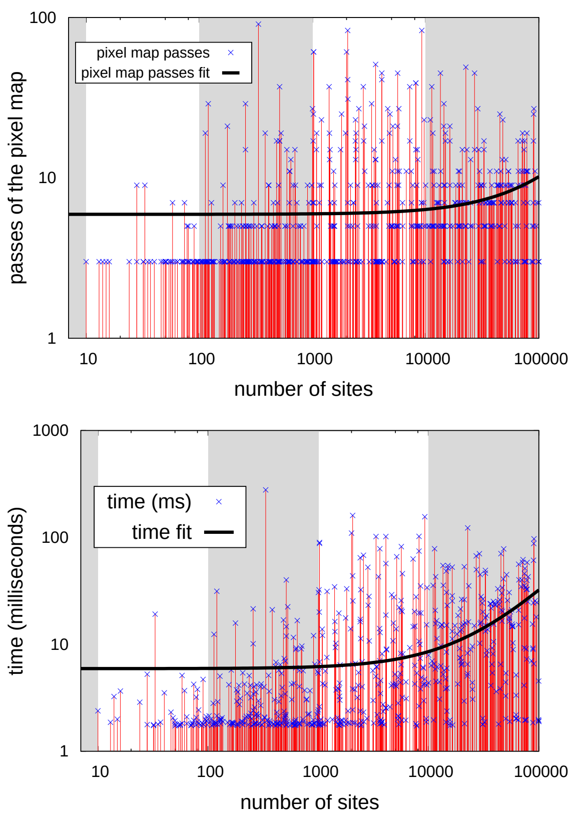


Figure 13: Plots of data relating sample runs of island labeling using the two-way-eight-neighbor testing scheme.

3.4GHz processor, and NVIDIA GeForce GTX 460 graphics card. The results presented here are based on tests with a pixel map resolution of 1000×1000. Handling datasets with more than a hundred-thousand point sites is not much meaningful with this resolution as this increases the chance of many very closely spaced sites being approximated by a single pixel. Therefore, the analytical data has been restricted to be bounded in the range $[1, 100000]$ for the number of input sites. Moreover, due to lack of space, the statistics relating weighted Voronoi diagram computation have been omitted. The presented results relate to the experiments approximating the straight skeletons. The Table 1, and Figures 12 & 13, summarize some information from the benchmarks.

It is evident from the figures presented in Table 1 that the four-way-four-neighbor testing scheme for island detection is a robust algorithm. This is also supported by the plots shown in Figure 12. Experiments show that, while detecting islands using this scheme, the number of passes of the pixel map do not grow with an increase in the number of sites being processed. Rather, this count of passes follows a constant value. Similarly, it can be observed in the second plot shown in Figure 12 that the increase in the time consumed by this island detection algorithm variant also does not exhibit any significant growth with respect to the number of input sites. A very slight increase in the time consumption occurs when the number of input sites crosses several tens of thousands and approaches the hundred-thousand limit. This is because a majority of pixels now pass the island test and require their island boolean flags to be updated.

Variant	Passes of pixel-map	Time (ms) consumed
1-way; 8-neighbor	455	3610.790
1-way; 8-neighbor; interlaced[1]	612	2771.825
1-way; 4-neighbor	612	2780.503
2-way; 8-neighbor	225	231.160
2-way; 8-neighbor; interlaced[1]	519	1848.751
4-way; 8-neighbor	14	104.622
4-way; 4-neighbor	14	99.104

Table 1: Peaks of data relating sample runs of various variants of the island detection algorithm applied to same datasets.

[1] Odd and even rows of pixels are processed in alternate iterations, similar to the interlaced raster scanning for scan-conversion of an image.

Figure 13 presents the plots of data relating sample runs of the island labeling algorithm using the two-way-eight-neighbor testing scheme. The least-squares line fit both for the number of passes of the pixel map, and for the time consumed by the labeling algorithm, with respect to the number of input sites, follows a constant value until the size of the input reaches an order of 10^4. A slight increase is observed in both of these trends beyond this level. The reason for this ascent is the increase in the number of island pixels requiring their label information to be updated. The increase in this case can observed slightly more significant than the one in case of island detection. This is because every island pixel now requires eight comparisons for getting its label information updated in contrast to the earlier case where every island pixel required four comparisons for having its island boolean flag modified.

4 CONCLUDING REMARKS

This paper highlights a very challenging problem (existence of islands) faced by geometric algorithms which make use of the graphics rendering and interpolation capabilities of a GPU. An efficient and general solution to this problem is proposed. The presented algorithms can be applied in combination with any such geometric technique thus ensuring its conformance to the task. The concept of black listing bad sites has been introduced and efficiently utilized for recovering correct diagram patches in lie of the islands. An incremental application of this patchwork on the pixel map transforms it into the required final diagram of the geometric structure being computed. The proposed approach ensures the accuracy of a patch by avoiding blacklisting of the sites which actually form the correct portion of the diagram in replacement of the corresponding island. This also certifies removal of all islands which may be stacked under this island at no extra processing cost. Moreover, the proposed algorithms also serve as a few steps towards the first ever GPU-based attempt to approximate the straight skeleton of a simple polygon.

ACKNOWLEDGEMENTS

This work was supported by the Higher Education Commission (HEC), Pakistan, and the Universität Salzburg, Austria.

REFERENCES

[AAAG95] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A novel type of skeleton for polygons. *Journal of Universal Comp. Sc.*, 1(12):752–761, 1995.

[CT05] J. Champagne and W. Tang. Real-time simulation of crowds using Voronoi diagrams. In *Theory and Practice of Comp. Graphics*, pages 195–201, Canterbury, United Kingdom, 2005. Eurographics Association.

[Den03a] M. O. Denny. *Algorithmic Geometry via Graphics Hardware*. PhD thesis, Universität des Saarlandes, Saarbrücken, 2003.

[Den03b] M. O. Denny. Solving geometric optimization problems using graphics hardware. *Comp. Graphics Forum*, 22(3):441–452, 2003.

[FG06] I. Fischer and C. Gotsman. Fast approximation of high-order Voronoi diagrams and distance transforms on the GPU. *Journal of Graphics, GPU, and Game Tools*, 11(4):39–60, 2006.

[Hae90] P. Haeberli. Paint by numbers: Abstract image representations. In *SIGGRAPH '90: Proc. of the 17th annual Conf. on Comp. Graphics and Interactive Techniques*, pages 207–214, New York, NY, USA, 1990. ACM.

[HCK+99] K. E. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized Voronoi diagrams using graphics hardware. In *SIGGRAPH '99: Proc. of the 26th Annual Conf. on Comp. Graphics and Interactive Techniques*, pages 277–286, New York, NY, USA, 1999. ACM.

[LZC09] C. L. Li, G. Zhou, and C. W. Chan. A graphical approach to approximate offset computation. In *Proc. of the 6th Intl. Conf. on Comp. Graphics, Imaging and Visual.*, CGIV '09, pages 217–221, Washington, DC, USA, Aug 2009. IEEE Comp. Society.

[OBS92] A. Okabe, B. Boots, and K. Sugihara. *Spatial tessellations: Concepts and applications of Voronoi diagrams*. John Wiley & Sons, Inc., New York, NY, USA, 1992.

[VR05] M. Vona and D. Rus. Voronoi toolpaths for PCB mechanical etch: Simple and intuitive algorithms with the 3D GPU. In *IEEE Intl. Conf. on Robotics and Automation*, pages 2759–2766, 2005.

[WND97] M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide (2nd ed.): The Official Guide to Learning OpenGL Version 1.1*. Addison-Wesley Longman Pub. Co., Inc., Boston, MA, USA, 1997.

[Yam05a] O. Yamamoto. An acceleration technique for the computation of Voronoi diagrams using graphics hardware. In *ICCSA (Part 1): Intl. Conf. on Computational Sc. and its Appl.s*, volume 3480 of *Lecture Notes in Comp. Sc.*, pages 786–795. Springer Berlin / Heidelberg, 2005.

[Yam05b] O. Yamamoto. Fast computation of three-dimensional convex hulls using graphics hardware. *Japan Journal of Industrial and Applied Mathematics*, 22:291–310, 2005.

Search for small monostatic polyhedra

Christian MINICH

Comp. Science department,
Metz University,
Ile du Saulcy
France-57045 Metz

minich@univ-lorraine.fr

ABSTRACT

We study a class of convex polyhedra that contains the famous 19-face monostatic polyhedron discovered by R. K. Guy in 1969. The properties of the polyhedra of this class allow to efficiently scan them so that it becomes possible to check in a reasonable time, with a computer program, whether the class contains a convex monostatic polyhedron with fewer than 19 faces.

Keywords
Monostatic, polyhedron.

1. INTRODUCTION

A convex monostatic polyhedron (CMP) is an homogeneous convex polyhedron that remains in stable equilibrium on only one of its faces. If laid on any other face, it rolls until it finally stops on the only face on which it is stable. In 1969, R. K. Guy and, two months later, Ken Knowlton independently discovered such a solid that had only nineteen faces (Figure 1) [Guy69] [Bry08].

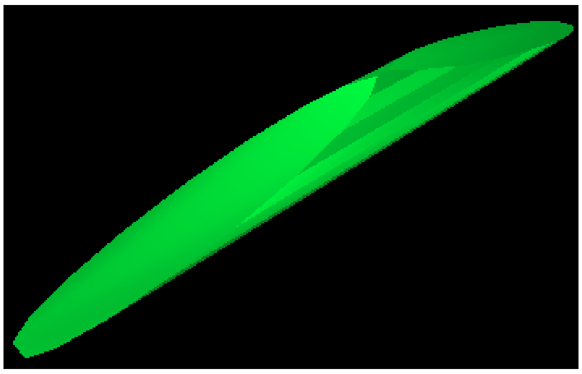


Figure 1: Guy's monostatic polyhedron

In dimensions other than 3, or for some special categories of polytopes, there has been some other results. For example, Arnold has proven that no 2-polytope (2D convex polygon) can be monostatic. In other words, any convex polygon is stable on at least two of its edges; another proof is available in [Pak].

Conway has also proven that there are no monostatic simplices in dimension 3 (no reference available) and Dawson did the same for dimensions up to 6 [Daw85]. This result has further been extended to dimensions 7 and 8 in [DFM98]. Surprisingly, Dawson has also proven that there exists monostatic simplices in dimension 10 and up [DaF01].

In dimension 3, there is no theoretical value for the least number of faces of a monostatic convex polyhedron. Up to last year, 19 was the minimum known value but, in last September, Andras Bezdek presented a construction procedure to build a 18-face CMP (Figure 2) [Bez11].

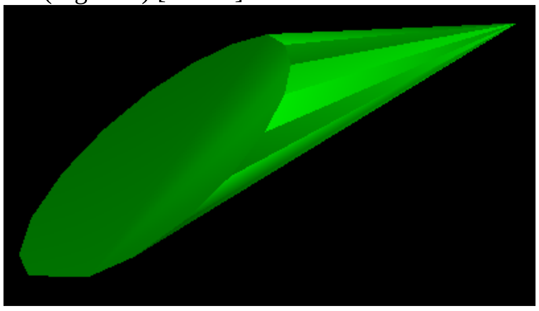


Figure 2: Bezdek's monostatic polyhedron

Besides this, Guy has proven that the least number of faces for the CMPs built with his procedure is 19 but, if some of the constraints he has combined to produce his CMP are relaxed, one still gets CMPs (see section 2). Because of this, and because it is now known that 19 is not a minimum, we think it is worth trying to find CMPs similar to Guy's object but with fewer than 19 faces.

This paper summarizes an experiment conducted in this goal: it has consisted in exploring a class of polyhedra similar to Guy's one with kind of an exhaustive traversal (with a given step). It has not been successful, in the sense that many CMP with 19 faces have been found but none with less than 19. On the other hand, a 17-face solid that is very close from being a CMP has been found in the class, so that some doubts remain on the fact that this class contains a CMP with fewer than 19 faces.

The paper is structured as follows: parts 2 and 3 present Guy's CMP and a class of polyhedra to which it belongs. Part 4 gives an algorithm to scan all the polyhedra in this class and part 5 details the first optimization of this algorithm. The next part gives and proves an important property concerning the center of mass of the studied polyhedra and part 7 explains how to use this property to implement another optimization. Then a method to evaluate the quality of a polyhedron of the class is described (is it monostatic or, if it is not, how far is it from being monostatic). The last part gives some results, concluding remarks and presents planned future work.

2. GUY'S OBJECT

For an convex polyhedron to be in stable equilibrium when laid on one of its faces, its center of mass has to project within this face. For example, the tetrahedron below (Figure 3) [Hep67] is in stable equilibrium on faces (A,B,D) and (A,C,D) only (and also has the unusual property that, when laid on one of the other two faces, it first rolls on the other unstable face before rolling again on a stable face). So, for a convex polyhedron to be monostatic, its center of mass must project in one and only one face.

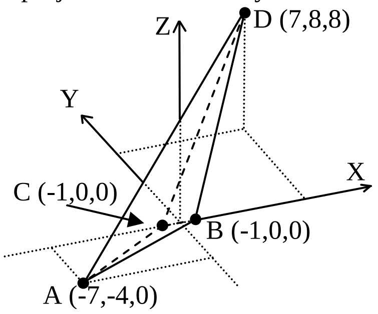


Figure 3: a tetrahedron with two stable faces

Guy's solid is a prism truncated by two symmetric planes. The section of the prism is a polygon of the xz-plane, symmetric with respect to the z- axis; the half-section is obtained by gluing a sequence of m right triangles with the same aperture π/m (Figure 4). As the bottom edge and its symmetric generate the same face, the total number of faces is 1 + 2.(m-1) + 2 = 2.m + 1.

Guy proved that, if m ≥ 9, by tilting and moving away the truncation planes sufficiently, it is possible to lower the center of mass so that:

- it stands lower than the point shared by all the triangles (O in Figure 4); so, it projects only in the lowest lateral face;
- it projects in none of the "caps" (the faces generated by the truncation).

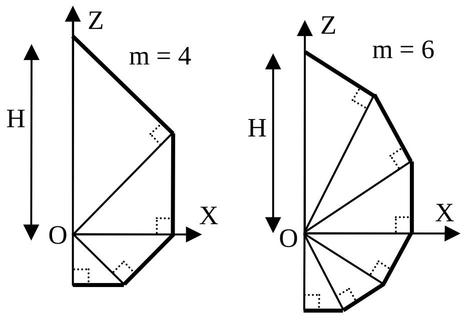


Figure 4: the half-sections of two Guy's objects

So the least number of faces for a CMP built with this procedure is 19.

But if one slightly moves some vertices of the profile of such a CMP (so the new profile no longer meets Guy's requirements), it is still possible to build a CMP. For example, if H = 100 (see Figure 4) and m = 9, the vertices of the half-profile of Guy's object are those in Figure 5-left; but with the vertices of Figure 5-right (this half-profile is also made of 9 right triangles but their apertures are not all the same) and with cutting planes y = -6.z + 624 and y =6.z - 624, we still get a CMP.

Guy's half-profile		Another valid half-profile	
X	Z	X	Z
0	100	0	100, 581019047
32.139380	88.302222	32. 326114341	88, 815276618
56.759574	67.643427	58, 637925590	65, 124016131
71.860144	41.488473	72, 895434578	37, 142079636
76.788242	13.539839	77, 297886249	16, 430172842
72.157344	-12.723287	72.157344678	-12.723286842
59.627353	-34.425868	59.627353253	-34.425868547
41.588017	-49.562669	41.588017084	-49.562668849
20.794008	-57.131069	20.794008542	-57.131069
0	-57.131069	0	-57.131069

Figure 5: two valid 19-face half-profiles.

Hence, Guy's construction procedure is not the only one that leads to half-profiles that allow to build CMPs; and may be the least number of faces of CMPs built with other procedures is smaller than 19. That's why we have undertaken to explore several classes of polyhedra; the class studied in this paper in described in the next section.

3. THE CLASS OF POLYHEDRA STUDIED IN THIS PAPER

The polyhedra studied in this paper have the following properties (Figure 6):

- they are prisms whose main axis is parallel to the y-axis, truncated by two planes that are symmetric with respect to the xz-plane;
- the profiles of the prisms are convex polygons of the xz-plane, symmetric with respect to the z-axis;

- a half-profile always starts with a line segment whose slope is negative and ends with a line segment perpendicular to the z-axis. Without loss of generality, the ends of the half-profile can be fixed to (0,0,100) and (0,0,0);
- the slopes of the cutting-planes are such that the faces they generate, the "caps", have an upwards outward normal vector (the z-coordinate of the outward normal vector is strictly positive);
- the cutting-planes are perpendicular to the yz-plane, so that the x-coordinate of the normal vector to the caps is 0.

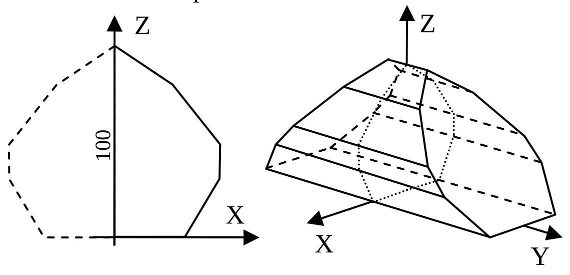


Figure 6

In one word, the polyhedra in this class have the same aspect as Guy's object, except that the half-profile just has to be convex. For any polyhedron in this class, due to the symmetries, the center of mass G lies on the z-axis , somewhere between (0,0,0) and (0,0,100) and the base face, generated by the lowest segment of the half-profile and its symmetric, is a stable face, as G projects in it. So, for a element of the class to be monostatic, G must project in no other face. This implies that G must project on no other edge of the profile but the base edge.

Note that Guy's object belongs to the class and that, for any polyhedron in the class, if its half-profile contains m edges, it has 2.(m-1)+3 = 2.m+1 faces. For a CMP to have fewer faces than Guy's object, m may vary from 2 to 8.

4. SCANNING THE CLASS

The base mechanism to traverse all the polyhedra of the class for a given m is to generate all possible m-sided half-profiles and, for a given half-profile, to enumerate all possible cutting planes. Each produced polyhedron is then checked for monostaticity. Letting m grow from 2 to 8 achieves a complete scan of the polyhedra we wish to study. In fact, we'll see later in the paper that it is useless to enumerate the cutting planes and to really build the polyhedra : an efficient validity test may be performed as soon as the half-profile is available. If the half-profile is declared to be valid, in a sense given below, it is sure that it is possible to exhibit cutting planes that lead to a monostatic polyhedron.

Of course, the generation of all possible half-profiles is performed with a given step (in fact, there are an angular step, $\Delta\alpha$, and a distance step, ΔL); the thinner the steps, the slower the algorithm. These steps are input parameters.

We first present the m-sided half-profile generation process and then its optimizations. This process is recursive. Let k be the current number of vertices in the profile (k=0 at the beginning as the profile is empty) and let P_i be the i^{th} vertex of the half-profile ($i \geq 1$):

Case 1: k = 0 (the profile is empty)

```
Add vertex $P_1$ (0,0,100), set k to 1 and recursively complete the half-profile
```

Case 2: 0 < k < m-1

For the profile to remain convex, the angle for the next edge $[P_k, P_{k+1}]$ may vary from α_{min} to α_{max} (both excluded), where :

- α_{min} is the anti-clockwise angle of $[P_k , O]$ with the horizontal (Figure 7)
- α_{max} is the anti-clockwise angle of $[P_{k-1} , P_k]$ with the horizontal

Exception: if k = 1, $\alpha_{min} = 3.\pi/2$ and $\alpha_{max} = 2. \pi$.

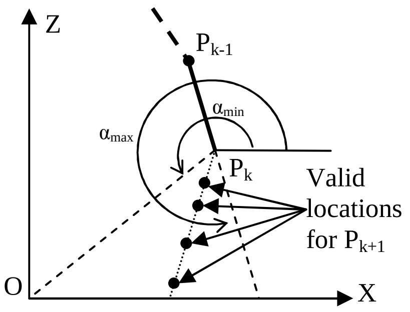


Figure 7: angular interval (dashed lines) and distance interval for the next vertex

For a given angle, the length of the next side may vary from 0 (excluded) to the distance D to the x-axis (excluded) (Figure 7). Beyond this length, the profile can't remain convex once O is added. So the algorithm is:

```
for $\alpha$ from $\alpha_{min}$ to $\alpha_{max}$ (both excluded) step $\Delta\alpha$
•   compute the intersection I between the x-axis and the line of angle $\alpha$ starting in $P_k$. Let D be the distance from $P_k$ to I.
•   for d from 0 to D (both excluded) step $\Delta L$
        add $P_k$ + d.(cos ($\alpha$), 0, sin ($\alpha$)) to the half-profile, add one to k and recursively complete the half-profile.
```

Case 3: $k = m-1$ (two more vertices must be added: P_m and P_{m+1})

The next vertex, P_m, has to be on the x-axis (Figure 8).

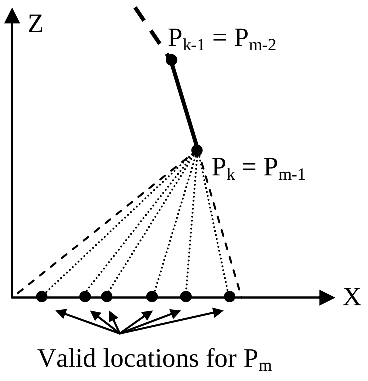


Figure 8: choosing the penultimate vertex

For the profile to remain convex, the angle for the next edge $[P_k, P_{k+1}]$ (that is $[P_{m-1}, P_m]$) may vary from α_{min} to α_{max} (both excluded and defined as above). So the algorithm is:

```
for α from $\alpha_{min}$ to $\alpha_{max}$ (both excluded) step Δ α
  • compute the intersection of the x-axis with the line of angle α starting in $P_k$.
  • add this point to the half-profile, set k to m and recursively complete the half-profile.
```

Case 4: $k = m$ (the half-profile just lacks the last vertex)

```
Add vertex $P_{m+1}$ (0,0,0), set k to m+1 and test the half-profile (see section 8)
```

Figure 9 shows some of the half-profiles produced with this method for m = 5.

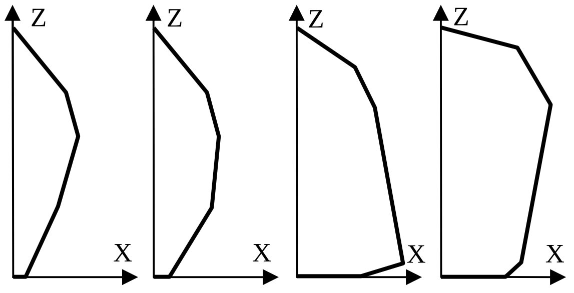


Figure 9: some half-profiles generated by the algorithm

We have no formula to express the time complexity of the half-profile generation but a simple reasoning gives a good idea of the unreasonable duration of the basic version. The iterative version of the algorithm is a sequence of 2.m - 3 nested loops:

```
...
for $\alpha_1$ from $\alpha_{min1}$ to $\alpha_{max1}$
  for $d_1$ from 0 to $D_1$
    add $P_2$ to the half-profile
    for $\alpha_2$ from $\alpha_{min2}$ to $\alpha_{max2}$
      for $d_2$ from 0 to $D_2$
        add $P_3$ to the half-profile
        ...
          for $\alpha_{m-1}$ from $\alpha_{min\_m-1}$ to $\alpha_{max\_m-1}$
            add $P_m$, $P_{m+1}$
            test the half-profile
```

Assume Δ α and Δ L both equal one. Then α_1 can take 89 values (from 270° to 360° both excluded); if α_1 is small (280 degrees, for example) then α_2 can only take a few values but if α_1 is large (350°), α_2 can vary up to 350°. Depending of d_1, this may represent more than 100 values. To make a rough approximation, we suppose that all loops (α and d loops) iterate only 20 times and that each instruction lasts 10^{-9} seconds. Then, for m = 8, only for the half-profiles generation, we get a duration of:

$$20^{2.m-3}.10^{-9} \text{ s} = 948 \text{ days}$$

If variations of the cutting planes, calculation and evaluation of the resulting polyhedra had to be added, it is clear that a computational exploration would be unaffordable.

5. FIRST OPTIMIZATION

Let $PP_{i\ (1 < i \leq m)}$ be the intersection point between the z-axis and the normal to $[P_{i-1}, P_i]$ in P_i (PP_i stands for Projected P_i) (Figure 10.a).

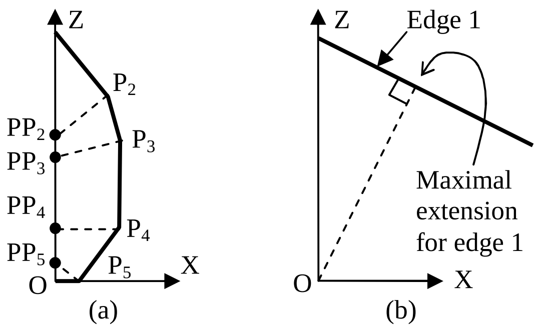


Figure 10

If PP_2 is lower than O, any point in $[P_1, O]$ projects on the first edge of the profile, including the center of mass as it is somewhere in $[P_1, O]$ by construction. So it is certain that the profile cannot lead to monostatic polyhedra. The consequence for the algorithm is that, once the slope of the first edge is chosen, it is useless to extend the first edge beyond the projection of O on this edge (Figure 10.b).

This optimization can be applied to any of the first m-2 edges but it is useful only for the edges whose angle is larger than $3.\pi/2$. Below this limit, the x-axis is intersected before the projection of O is reached. So case 2 of the algorithm becomes:

Case 2: $0 < k < m-1$

```
for $\alpha$ from $\alpha_{min}$ to $\alpha_{max}$ (both excluded) step $\Delta\ \alpha$
    if $\alpha > 3.\ \pi/2$
        D = distance from $P_k$ to the projection of O on the line of angle $\alpha$ starting at $P_k$
    else
        compute the intersection I between the x-axis and the line of angle $\alpha$ starting in $P_k$.
        D = distance from $P_k$ to I

    for d from 0 to D (both excluded) step $\Delta$ L
        add $P_k$ + d.(cos ($\alpha$), 0, sin ($\alpha$)) to the half-profile, add one to k and recursively complete the half-profile.
```

Case 3 is modified similarly. With this optimization, half-profiles 3 and 4 in Figure 9 are no longer generated as edge 3 (number 3) or edge 1 (number 4) are too long.

6. LOWER BOUND FOR THE Z-COORDINATE OF THE CENTER OF MASS

As depicted in Figure 11.a, the polyhedra of this study are composed of three parts: a prism (central part) and two truncated prisms (at both ends). For a given half-profile, the prism, the profile and the half-profile have centers of mass with the same z-coordinate, we call it ZG_1. The centers of mass of the two truncated prisms have the same z-coordinate, called ZG_2 (Figure 11.b). This section mainly aims at proving that whatever the slope of the cutting planes is, ZG_2 remains the same: the z-coordinate of a truncated prism is *not* a function of the slope of the cutting plane.

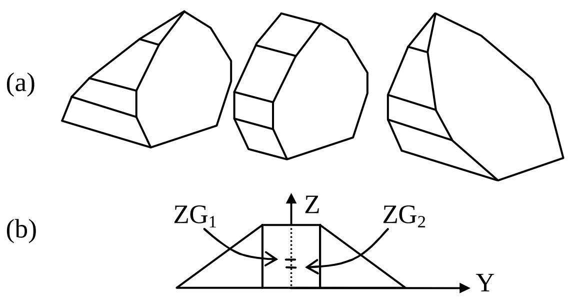


Figure 11

To do this, without loss of generality, we suppose that the truncated prism is positioned as depicted in Figure 12.a. In particular, the cutting plane contains P_1 so its equation is $y = a.z + b$ where $b > 0$ fixes its slope and where $a = -b / z_1$ (Figure 12.a). Besides this, we only consider the truncated prism produced by the half-profile as its center of mass has the same z-coordinate as the center of mass of the truncated prism. The half-profile is divided into (m-1) triangles $(P_1, P_i, P_{i+1})_{1 < i \leq m}$ and the truncated prism it generates is divided into (m-1) truncated triangular prisms built from these triangles (Figure 12.b).

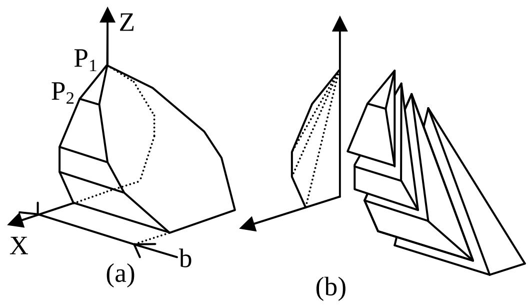


Figure 12: partition of the truncated prism

Let us first calculate the volume and the z-coordinate of the center of mass of the ith truncated triangular prism generated from the ith triangle. For the sake of clarity, the vertices of the triangle are not denoted (P_1, P_i, P_{i+1}) but (A,B,C), where $A = (x_A,0,z_A)$, $B=(x_B,0,z_B)$ and $C=(x_C,0,z_C)$. Without loss of generality, we can assume that $z_A \leq z_B \leq z_C$ and, at the moment, we assume that $z_A < z_B < z_C$ (Figure 13.a).

The equations of lines (A,B), (A,C) and (B,C) are:

$x = \alpha_{AB}.z + \beta_{AB}$ where $\alpha_{AB} = (x_B - x_A) / (z_B - z_A)$ and $\beta_{AB} = x_A - z_A.\alpha_{AB}$

$x = \alpha_{AC}.z + \beta_{AC}$ where $\alpha_{AC} = (x_C - x_A) / (z_C - z_A)$ and $\beta_{AC} = x_A - z_A.\alpha_{AC}$

$x = \alpha_{BC}.z + \beta_{BC}$ where $\alpha_{BC} = (x_C - x_B) / (z_C - z_B)$ and $\beta_{BC} = x_B - z_B.\alpha_{BC}$

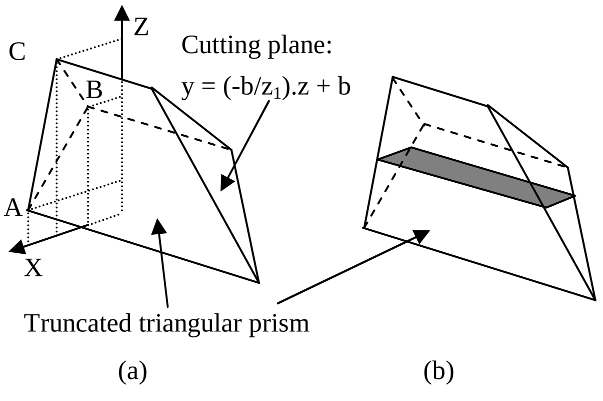


Figure 13

The intersection between the triangle and a line of the xz-plane, parallel to the x-axis, located between A and B and whose z-coordinate is z is a line segment of length $| \alpha_{AC}.z + \beta_{AC} - \alpha_{AB}.z - \beta_{AB} |$. This line segment generates in the prism a rectangle of area $(a.z + b) . | \alpha_{AC}.z + \beta_{AC} - \alpha_{AB}.z - \beta_{AB} |$ (Figure 13.b) and, of course, the z-coordinate of the center of mass of this rectangle is z. If the line is between B and C, the formulas are similar. So the volume V_i of the truncated triangular prism is the sum of the areas of all these rectangles, for z varying first from z_A to z_B and then for z varying from z_B to z_C:

$$V_i = \int_{z=z_A}^{z=z_B} (a.z+b).|\alpha_{AC}.z+\beta_{AC}-\alpha_{AB}.z - \beta_{AB}|.dz + \int_{z=z_B}^{z=z_C} (a.z+b).|\alpha_{AC}.z+\beta_{AC}-\alpha_{BC}.z - \beta_{BC}|.dz$$

As a = -b / z_1, V_i can be factorized under the following form:

$$V_i = b.\left(\int_{z=z_A}^{z=z_B}\left(\frac{-z}{z_1}+1\right).|\ldots|.dz + \int_{z=z_B}^{z=z_C}\left(\frac{-z}{z_1}+1\right).|\ldots|.dz\right)$$

that is: $V_i = b.M_i$ where M_i contains no occurrence of b.

The center of mass is the weighted average of the centers of mass of the rectangles, so its z-coordinate ZG_i is:

$$z_{G_i} = \frac{1}{V_i}\int_{z=z_A}^{z=z_B} (a.z+b).|\alpha_{AC}.z+\beta_{AC}-\alpha_{AB}.z - \beta_{AB}|.z.dz + \frac{1}{V_i}\int_{z=z_B}^{z=z_C} (a.z+b).|\alpha_{AC}.z+\beta_{AC}-\alpha_{BC}.z - \beta_{BC}|.z.dz$$

Again, b can be put into factor and ZG_i can be put in the following form:

$$z_{G_i} = \frac{1}{V_i}.b.R_i$$

where R_i contains no occurrence of b.
In other words,

$$z_{G_i} = \frac{R_i}{M_i}$$

where neither R_i nor M_i contain any occurrence of b.

If two points of the triangle have the same z-coordinate (the assumption $z_A < z_B < z_C$ becomes wrong), one of the integrals vanishes, in both R_i and M_i, but z_{G_i}keeps the same aspect (a ratio with no occurrence of b).

We can now calculate the z-coordinate of the center of mass of the truncated prism: it is the weighted average of the z-coordinates of the centers of mass of the (m-1) truncated triangular prisms:

$$ZG_2 = \frac{\sum_{i=2}^{m} V_i\,.z_{G_i}}{\sum_{i=2}^{m} V_i} = \frac{b.\sum_{i=2}^{m} R_i}{b.\sum_{i=2}^{m} M_i} = \frac{\sum_{i=2}^{m} R_i}{\sum_{i=2}^{m} M_i}$$

So b, which controls the slope of the cutting plane, does not appear in the expression of ZG_2. This ends the proof that ZG_2 does not depend on the slope of the cutting plane but only on the shape of the profile.

This has interesting consequences for the polyhedra of this study. The z-coordinate of their center of mass is:

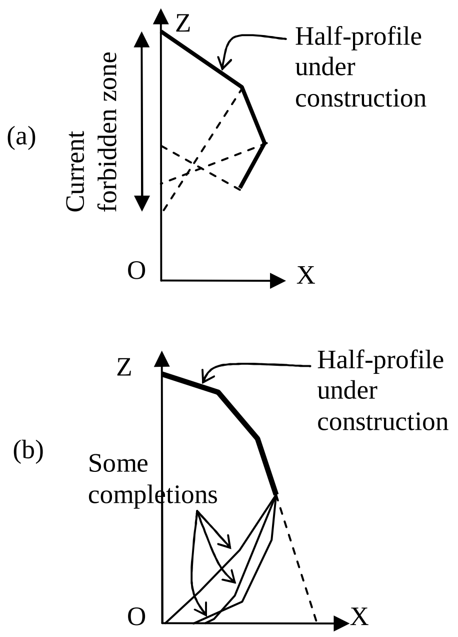
$$ZG = \frac{VP}{VP+2.VTP}.ZG_1 + \frac{2.VTP}{VP+2.VTP}.ZG_2$$

where VP is the volume of the prismatic part and VTP is the volume of the truncated prism. This means that the minimum z-coordinate of the center of mass of all the polyhedra that can be built from a given half-profile is ZMinG = Min (ZG_1, ZG_2). Besides this, if $ZG_1 > ZG_2$ (ZMinG = ZG_2), by increasing the slope of the cutting planes, that is by getting them closer to the horizontal, it is possible to increase VTP while VP remains the same. This means that the center of mass can be brought infinitely close to (0, 0, ZG_2), as the weight of ZG_1 in the previous formula tends to 0 and the one of ZG_2 tends to 1. Symmetrically, if $ZG_1 < ZG_2$, by bringing the cutting planes closer and closer to the vertical and by moving them away, the weight of ZG_1 increases and the center of mass can be brought infinitely close to (0, 0, ZG_1). In other words, for a given profile, by acting on the cutting planes, it is possible to build a polyhedron whose center of mass is infinitely close to the lower of the two partial centers of mass (the one of the prism and the one of the truncated prism).

7. SECOND OPTIMIZATION

Remind that PP_i is the projection on the z-axis of the ith vertex of the profile (P_i), perpendicularly to the (i-1)th edge. For a given half-profile (under construction or complete), we call "forbidden zone" the line segment from P_1 to the lowest PP_i (Figure 14.a). This zone can easily be incrementally maintained each time a point is added to the half-profile: one just has to project the new vertex on the z-axis and to compare the z-coordinate of the projected point with the current lowest point of the forbidden zone; the forbidden zone is thus available at any stage of the construction of a half-profile. We call ZMinFZ the z-coordinate of the lowest PP_i. The forbidden zone has two properties useful to the second optimization: first, any point in the forbidden zone projects on at least one of the edges of the half-profile under construction; second, the forbidden zone can never reduce: it consists in a single point when the half-profile is added its first point and increases downwards or remains the same each time a vertex is added to the half-profile.

Consider now a half-profile under construction; among all the manners to complete it (Figure 14.b), the one that allows to build the polyhedron with the lowest center of mass is the one where as much matter as possible is added in the lower part of the profile. Let's call PHP this particular half-profile. PHP is obtained by extending the current last edge of the half-profile up to its intersection with the x-axis (in dashed line in Figure 14.b) and then by adding O.



(a)
Z
Half-profile under construction
Current forbidden zone
O
X
(b)
Z
Half-profile under construction
Some completions
O
X

Figure 14

We can (see previous section) calculate the lowest possible z-coordinate for the centers of mass of all polyhedra that would be built from PHP, we call it ZMinG. By construction, we know that the centers of mass of all the polyhedra that can be built from the half-profile under construction are above ZMinG. If ZMinG > ZMinFZ, we can deduce that all these centers of mass are in the forbidden zone so they project in one of the side faces. As the forbidden zone cannot reduce while the half-profile is completed, we know that none of these polyhedra can be monostatic so it is useless to complete the current half-profile. So the second optimization just consists in testing, each time a vertex is added to a half-profile under construction, whether ZMinG > ZMinFZ. If this occurs, it is useless to complete the current half-profile and a huge amount of calculations is saved.

8. VALIDATING A PROFILE

The algorithm presented in part 4 and the optimizations of part 5 and 7 allow to enumerate potentially interesting half-profiles. Each time a half-profile is completed, it must be checked whether it can produce a CMP. For this, the test is strictly the same as the one of the second optimization: ZMinFZ and ZMinG (which are now those of a complete half-profile) are compared. If ZMinG ≥ ZMinFZ, it is sure that no CMP can be built from the half-profile. Otherwise, we know it is possible to build a polyhedron whose center of mass, G, is infinitely close to (0,0,ZMinG) (section 6). We know that G doesn't project on any edge of the profile (except the last one, of course). The remaining problem is that G should not project in one of the two oblique faces. This can be achieved easily: if ZMinG=ZG2, it suffices to increase the slope of the cutting planes until G doesn't project in the oblique faces anymore; this increases the relative volume of the truncated prisms and gets G even closer to ZMinG (which ensures that it doesn't project in any side face except the base face). If ZMinG=ZG1, it suffices to extend the prismatic part of the polyhedron without changing the slope of the planes, until G doesn't project in the oblique faces anymore; this increases the relative volume of the prismatic part and brings G even closer to ZMinG. So validating a profile just consists in checking that ZMinG < ZMinFZ; besides this, if the profile is validated, there is no need for scanning all possible cutting planes, which, again, enormously reduces the complexity of the traversal of the class of polyhedra targeted in this study. Finally, ZMinG and ZMinFZ also offer a way to assign a value to a profile: this value is ZMinFZ - ZMinG. If it is positive, the profile is valid. Otherwise, the more negative the value, the worse the profile.

9. RESULTS, CONCLUSION AND FUTURE WORK

The algorithms described in the previous sections have been implemented in C++ and tested on Personal Computers with an Intel Core I5-2500 processor.

Even though the optimizations presented above drastically reduce the time to traverse the whole class, it still takes months, sometimes years to do it with thin steps on a single computer when m = 7 or 8. To reduce these durations, some tests have been distributed on 60 PCs, so that an execution that would have last one year only lasts a week. Even though, thin steps (e.g. 1 degree for the angular step and 1 for the length step) remain inaccessible for m = 8. The table below gives the durations we have experienced (as if the tests had been conducted on a single computer).

m (nb of edges)	Δα and ΔL	Duration	Best profile value	Number of profiles tested
4	1° - 1	< 1 s	-18.37	102 294
5	1° - 1	3 mn	-13.95	287 331 878
6	1° - 1	30 hours	-10	307 492 054 424
7	3° - 3	4,5 days	-7.97	594 003 064 639
8	10° - 3	66 mn	-7.52	3 851 621 193
8	5° - 3	19 days	-4.66	492 902 974 042

At the moment, no CMP has been found for values of m smaller than 9. The reason may be that, because of the big steps we have to use to reduce the duration of the program when m > 6, some CMPs have escaped from the search; of course, another reason could be that there is no CMP in this class but we believe it is not the case because another experiment, that has just been launched and consists in traversing the class with an evolutionary algorithm, has found a 17 faces polyhedron that is very close from being monostatic; its value is -0.55249746 (see Figure 15).

Nearly valid half-profile	
X	Z
0	100
28.5812	83.5373
41.9472	66.1401
47.4035	49.0845
46.1614	30.7800
40.8933	17.5852
32.1911	7.12819
20.2575	0
0	0

Figure 15: the half-profile of a quasi monostatic 17-faces polyhedron

For m = 9, things are easier (and faster) as a solution is known (the half-profile of Guy's object). By scanning only the neighborhood of this profile, thousands of CMPs have been found but this is of little interest as, first, it doesn't improve Guy's record and, a fortiori, it doesn't improve Bezdek's new record.

Several extensions to this work are planned: the neighborhood of the 17 faces quasi monostatic polyhedron will be explored with the exhaustive approach and thin steps. The values of the steps will dynamically adapt during the exhaustive approach, so that the zones where the half-profiles are bad (very negative values) are scanned faster than those where the half-profiles are good (nearly valid). After that, other classes of polyhedra will be explored, beginning with a class containing Bezdek's CMP.

Finally, two questions arise from the observation of Guy's and Bezdek's objects: both are built from the same profile (a sequence of 9 stuck right triangles) so : 1) is it possible to build other classes of CMPs from this profile and 2) would this profile be the common denominator of all CMPs ?

10. ACKNOLEDGMENTS

The author wishes to thank the reviewers for their valuable comments and R. Dawson who let him know about the existence of Bezdek's CMP.

11. REFERENCES

[Hep67] Heppes, A. A double tipping tetrahedron. SIAM review, Vol 9 (3), pp.599-600, 1967.

[Guy69] Guy, R.K. Solution to problem 66-12, Stability of Polyhedra. SIAM review, Vol 11, No 1, pp.78-82, 1969.

[Daw85] Dawson, R. J. M. Monostatic simplexes. Amer. Math. Monthly 92 (1985), no. 8, 541–546

[DFM98] Dawson, R. J. M., Finbow, W., Mak, P. Monostatic simplexes. II. Geom. Dedicata 70 (1998), 209–219

[DaF01] Dawson, R. J. M., Finbow, W. Monostatic simplexes. III. Geom. Dedicata 84 (2001), 101–113

[Pak] Pak, I. Lectures on Discrete and Polyhedral Geometry, Internet book, http://www.math.ucla.edu/~pak/

[Bry08] Bryant, J., and Sangwin, C. How round is your circle - Where engineering and mathematics meet. Princeton University Press, page 275, 2008.

[Bez11] Bezdek, A. On stability of polyhedra. Workshop on Discrete Geometry, Sep 13-16, 2011, Fields Institute, Canada

Creating Animatable Non-Conforming Hexahedral Finite Element Facial Soft-Tissue Models for GPU Simulation

Mark Warburton Steve Maddock
Department of Computer Science
The University of Sheffield
211 Portobello Street
Sheffield, S1 4DP, United Kingdom
{ M.Warburton | S.Maddock }@dcs.shef.ac.uk

ABSTRACT

Physically-based animation techniques enable more realistic and accurate animation to be created. Such approaches require the creation of a complex volumetric model that can be realistically and efficiently simulated, particularly for interactive computer graphics applications. We present an approach to automatically construct animatable non-conforming hexahedral finite element (FE) facial soft-tissue simulation models, including automatic determination of element material types, boundary conditions and muscle properties, making them immediately ready for simulation.

Keywords
physically-based modelling, soft-tissue modelling, facial modelling, physically-based animation, finite element method.

1 INTRODUCTION

Facial modelling and animation is one of the most challenging areas of computer graphics. While various techniques have been proposed to create and animate facial models, by using a physically-based model, the effects of muscle contractions can be propagated through the facial soft tissue to deform the model in a more realistic and anatomical manner.

Physics-based soft-tissue simulation systems often focus on either efficiently producing realistic-looking animations for computer graphics applications [TW90, KHS01], or simulating models with high physical accuracy for studying soft-tissue behaviour [BJTM08, HMSH09] or surgical simulation [KRG$^+$02, ZHD06]. Popular simulation techniques include the efficient mass-spring (MS) method [TW90, KHS01], the accurate but computationally complex finite element (FE) method [SNF05, HMSH09], and the FE-based but precomputation-heavy mass-tensor (MT) method [MSNS05, XLZH11]. Physics engines, which focus on performance and stability, can also be used [MHHR06]. Indeed, increases in computational power, and the use

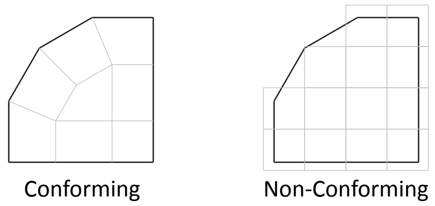


Figure 1: A conforming and non-conforming simulation model.

of GPU computing architectures mean that complex FE simulations are now possible in real time [TCO08].

Physics-based simulations require an appropriate simulation model to be created, for example, using surface meshes of the object to be simulated. As illustrated by Figure 1, such models can either conform to a surface mesh [MBTF03, BJTM08], or a non-conforming model with a bound surface mesh can be used [Coo98, DGW11]. High-quality conforming models that can be efficiently simulated are often difficult and time-consuming to create, although such models are usually required for high-accuracy applications. In contrast, non-conforming models can enable more efficient production of stable, realistic-looking animations for computer graphics applications.

The main aim of this work is to develop an automatic process to easily construct animatable non-conforming hexahedral FE simulation models, including automatic determination of element material types, boundary conditions and muscle properties. While the focus is on creating facial soft-tissue models (the soft tissue be-

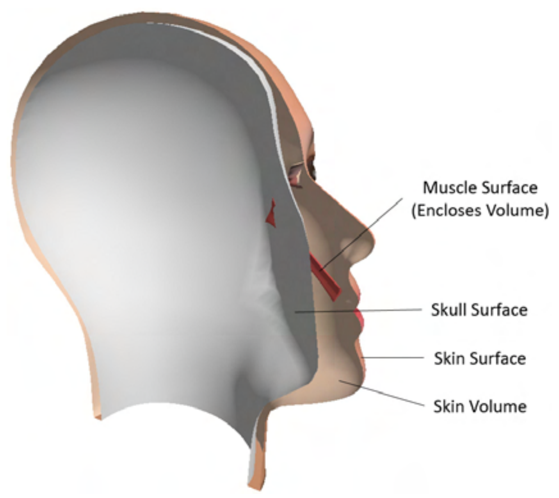


Figure 2: Surfaces and volumes of a facial soft-tissue model. The whole volume between the skull and skin surfaces (i.e. the skin and muscle volumes) is discretised to create an FE facial soft-tissue model.

tween the skull and outer skin surface, as shown by Figure 2), the process can be used to create any multi-layered model from any surface meshes. Such facial models can be used, for example, to efficiently produce realistic-looking facial animations for computer graphics applications. The following sections detail relevant related work, followed by a description and examples of the model creation process, including model simulation examples using our GPU FE system, finishing with a comparison between conforming and non-conforming simulation meshes, particularly for GPU FE simulations.

2 RELATED WORK

2.1 Physically-Based Facial and Soft-Tissue Models

Physically-based facial animation systems for computer graphics applications normally consist of muscle and skin models, sometimes along with a skull model and wrinkle models. For increased realism, a skull model can include a rotatable mandible, which can be geometrically controlled [KHS01] or physically-based [Cou05]. Muscles have been modelled using vectors [Wat87], and more anatomically accurate geometric [KHS01, Cou05] and physics-based volumes [BWL+10, RP07]. Many muscle contraction models are based on a Hill-type model [RP07, LST09], some of which are biologically inspired [HMSH09], and the direction of contraction can be approximated as parallel to the central action curve [TZT09], or, more anatomically, by using a fibre field [SNF05].

Various facial soft-tissue models have been proposed, ranging from simple but efficient physics-engine-based [CP06] and MS models [TW90, KHS01], to more anatomical and realistic FE models [SNF05, BWL+10]. Detailed models of blocks of skin and soft tissue have also been created [KSY08], along with complex soft-tissue constitutive models [Bis06]. Due to its efficiency, the TLED algorithm has been used for various non-linear FE soft-tissue simulations [TCO08], resulting in large speed-ups. The FE-based MT method has also been used to produce such simulations and also for facial surgical applications [MSNS05, XLZH11], showing similar accuracy to the FE method when simulating small displacements.

2.2 Model Creation Approaches

The model creation process is normally difficult and time-consuming, and it is also dependent on the required model structure. To create a model for FE simulation, a suitable mesh must be created, and FE simulation properties, such as boundary conditions, must be set. Regarding model element types for simulation, we only consider linear elements with a single integration point for optimal computational performance.

Simple automatic model creation approaches have been used that just create a layered MS model and skull from a surface mesh [TW90]. On the other hand, CT and MRI scans, or anthropometric data can be used to manually or automatically create an anatomical reference head model [MSNS05, KHS01]. Such data from the Visible Human Dataset[1] has previously been used for reference model creation [SNF05]. Various techniques have been proposed to deform reference skull, muscle or full physically-based head models using manually defined landmarks [KHS01, AZ10], although these often rely on good landmark placement. Kähler et al. also developed an interactive editor to enable easy muscle creation by processing user-specified grid points [KHS01].

Numerous algorithms exist for fast automatic generation of high-quality tetrahedral models that conform to surface meshes [MBTF03, SG05]; however, 4-node tetrahedra are susceptible to volume locking, particularly when simulating incompressible materials like soft tissue. In contrast, reduced-integration 8-node hexahedral elements (with hourglass control) have increased stability and accuracy [WJC+10], particularly when modelling non-linear anisotropic materials [fLhLfT11], and can be used to create meshes using fewer elements, normally outweighing the efficiency of tetrahedra. Hexahedra are therefore often preferred for FE simulations.

Although various algorithms for producing conforming hexahedral meshes have been proposed [SKO+10,

[1] `http://www.nlm.nih.gov/research/visible/visible_human.html`

ZHB10, NRP11], hexahedral mesh generation is often difficult and time consuming, and, without heavy manual work, many such algorithms suffer problems regarding element quality and robustness, particularly with complex geometries like soft tissue. Techniques have been proposed to improve the quality of hexahedral meshes [ISS09, SZM12], although these can produce models with an increased number of elements.

Simple hexahedral meshes can also be merged to produce a complex mesh [SSLS10, Lo12], although these approaches would require a manual decomposition of the complex model such that high quality elements are able to be produced during the merging process. Similarly, conforming and non-conforming domain decomposition FE methods can be used [Lam09], which involve performing an FE analysis on a model decomposed into several independent subdomains. Some other techniques involve deforming a reference hexahedral mesh [CPL00, fLhLfT11], although such approaches require a high-quality reference mesh, and often also require manual work or modifications to the final mesh.

Alternatively, non-conforming hexahedral meshes are easier to create, for example, using voxelisation techniques, and a surface mesh can also be bound to the volume mesh for visual purposes. Such meshes can be used to create models for more stable and computationally efficient FE simulations [DGW11]. Kumar et al. performed linear elastic FE simulations using structured non-conforming hexahedral grids, and compared these with conforming hexahedral simulation meshes [KPB08], which produced similar stresses, although only relatively simple models were examined. Non-conforming tetrahedral facial and soft-tissue FE models have also been used for stability and performance reasons [Coo98, SNF05], although linear tetrahedral elements can cause problems such as volume locking.

Once an FE simulation mesh has been created, model properties, such as element constitutive properties and boundary conditions, must be specified. Developing on current techniques for producing non-conforming hexahedral meshes with bound surface meshes, our model creation process automatically produces complete simulation-ready multi-layered FE models (i.e. with FE model properties computed) from any surface meshes. Focussing on facial soft-tissue models, this includes automatically assigning material properties to different regions of soft tissue, such as muscle and skin, and determining boundary conditions and muscle properties. With sensible model data organisation, the models can be efficiently simulated on the GPU.

3 NON-CONFORMING HEXAHEDRAL FINITE ELEMENT SOFT-TISSUE MODEL CREATION

A process has been developed to create simulation meshes with hexahedral elements that approximate but don't conform to surface meshes. Such meshes have several advantages over simulation meshes that conform to the shape of the object they are simulating, including advantages relating to mesh creation, stability and computational performance. A comparison between using conforming and non-conforming hexahedral simulation meshes with a CUDA FE system is presented in the discussion in Section 4.3.

A program has been implemented to voxelise a closed surface mesh, that can include closed internal boundaries, described by an OBJ file. Ray-polygon intersection tests for a ray fired from voxel centres are used to determine whether voxels are enclosed by a set of polygons. The enclosed voxels are used as hexahedral elements to produce a hexahedral FE mesh for use with our simulation system.

The surface mesh can contain various surfaces, and elements are separated depending on the user-defined collection of surfaces (volumes) by which they are enclosed, enabling different sets of elements to be assigned different material laws and properties. For example, with a facial mesh, there may be surfaces for the skull, skin and each muscle. The skull and skin surfaces might together define a single 'skin' volume (i.e. the volume of soft tissue between the skull and skin surfaces), whereas the volume-enclosing muscle surfaces could define their own volumes, as illustrated by Figure 2.

Element properties are determined by firstly assigning each volume a *level* and a *priority* within that level, where level 0 is the lowest level, and priority 0 is the highest priority within a level. Semantically, a volume in a higher level is contained within, and bound by, either a single or several volumes in the level immediately below (see Figure 3). Continuing with the facial mesh example, level 0 might consist solely of the skin volume, whereas level 1 might consist of the muscle volumes, each assigned a different priority. Lower-level volumes are voxelised first. Properties of elements are then overwritten if they are also contained within an immediately higher-level volume; hence, properties of elements within the skin volume would be overwritten by those that are also enclosed by a muscle volume. Within a level, higher-priority volumes are voxelised first, although properties of elements within lower-priority volumes are not overwritten.

This level-based voxelisation process is described by Algorithm 1, and illustrated by Figure 3. It can be seen that only lenient requirements are imposed on the

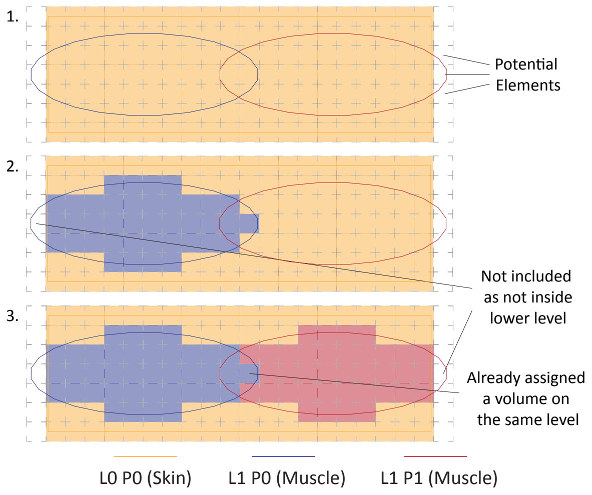

1.
2.
3.
Potential Elements
Not included as not inside lower level
Already assigned a volume on the same level
L0 P0 (Skin)
L1 P0 (Muscle)
L1 P1 (Muscle)

Figure 3: An example of the level-based voxelisation process for a skin block containing two muscles, showing the model state after each level priority has been considered in turn. Note this is a 2D illustration of a 3D process.

```
1:  $volLevs \leftarrow$ volume levels
        {array (levels) of arrays (priorities)}
2:  for all voxels, $v$ do
3:      {from lowest to highest level}
4:      while $levPris \leftarrow volLevs.next$ and
        $levChanged(v)$ do
5:          $lev \leftarrow$ level
6:          {from highest to lowest priority}
7:          while $pri \leftarrow levPris.next$ and not
            $priChanged(v)$ do
8:              if $insideVolume(v, lev, pri)$ then
9:                  $setVolume(v, lev, pri)$
```

Algorithm 1: The voxelisation process, which includes determining the volume by which each element is enclosed.

creation of the surface mesh; for example, as muscles should be contained within the skin volume, parts of muscle surfaces that cross the bounds of this volume are appropriately ignored. Muscle surfaces can therefore simply penetrate the skull (like the muscle in Figure 2), rather than having to 'attach' and conform to the skull surface, therefore simplifying the modelling of surface meshes.

User-specified sections of a surface mesh within a particular bounding box can also be voxelised, enabling easy simulation of only relevant sections of meshes with increased simulation speed, as opposed to simulating the whole mesh. Although distorting element shape, and therefore possibly impacting simulation stability, the hexahedral length, width and height can be independently set, which could be useful, for example, to help improve performance when modelling large, thin materials.

```
1:  for all rigid vertices, lines and polygons, $r$ do
2:      for all voxels, $v$ do
3:          $c \leftarrow centre(v)$
4:          $p \leftarrow getClosestPosition(r, c)$
              $\{p \leftarrow r$ for vertices$\}$
5:          if $insideBoundingBox(v, p)$ then
6:              $l \leftarrow p - c$
7:              $i \leftarrow getIntersection(v, l)$
8:              for all $v$ nodes, $n$ do
9:                  if $withinRange(n, i)$ then
10:                     $setRigid(n)$
```

Algorithm 2: The process to determine rigid nodes.

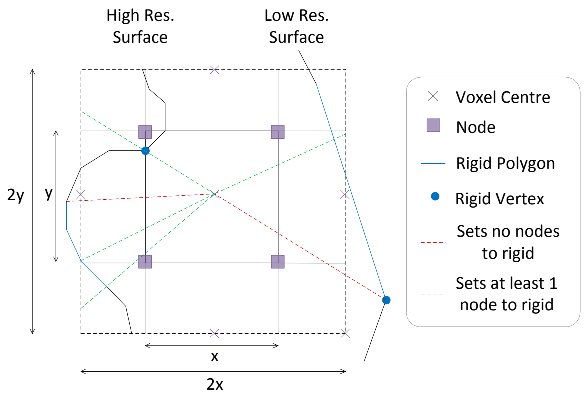


Figure 4: Examples of arbitrarily selected rigid surfaces and nodes inside and outside a voxel bounding box (the dashed box surrounding the voxel). Note this is a 2D illustration of a 3D process.

When voxelising a soft-tissue mesh, if surfaces are given names that correctly identify them (e.g. if the names of surfaces enclosing muscles start with the word 'muscle'), an animatable soft-tissue mesh can be automatically created with fixed skull nodes and animatable muscles. Using a collection of user-defined rigid vertices, lines and polygons on the surface mesh, an attachment process, described by Algorithm 2, has been developed to determine rigid simulation nodes, such as fixed skull nodes on a facial model.

Figure 4 illustrates the bounding box test in line 5 of the attachment algorithm, which is successful if position p (a rigid vertex position, or the closest point on a rigid line or polygon to the voxel centre) lies within a box, centred at voxel centre c, that has sides twice the length of voxel v in each direction. The test in line 9, to determine which nodes to set as rigid depending on the location of intersection point i with respect to the nodes of the voxel face on which it lies, is illustrated by Figure 5. For each node n, if i lies within a box starting at n that has sides 0.75x the length of v in each direction, n is set to rigid. The box side lengths could be changed

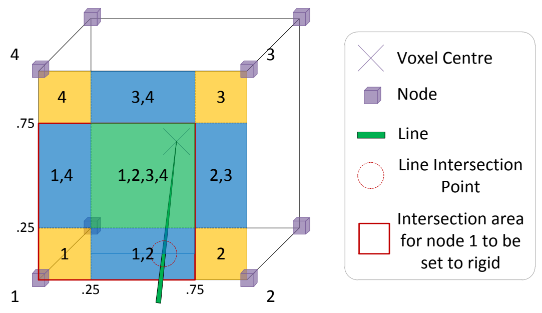


Figure 5: An indication of which nodes are set to rigid depending on the location of the line intersection point on the voxel face, indicated by the node indices inside the various sections. An example line is shown, which would set nodes 1 and 2 to rigid.

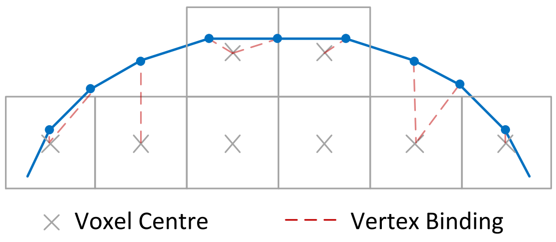


Figure 6: Vertices of a surface mesh bound to the closest elements. Note this is a 2D illustration of a 3D process.

to make the algorithm more or less 'strict'; for example, with side lengths of 0.5x, only one node would be set to rigid for each rigid polygon, line and node. However, 0.75x seems to work well and produce the desired results with the current examples.

The attachment algorithm can also handle cases when the simulation mesh is much higher or lower resolution than the surface mesh, as shown by Figure 4. After attaching skull nodes, each muscle can then be assigned a contraction centre, which is calculated as the average of the muscle nodal positions that lie on the simulation skull.

After simulation model creation, as with Dick et al.'s simulation approach [DGW11], the vertices of the surface mesh can be simply bound to and animated with elements of the FE mesh using trilinear interpolation and extrapolation (see Figure 6). A position, **p**, is bound to the closest element (determined by a distance test from the element centres) using three weights, w_i - one along each local axis, i, from the first node of the element, **x**. For a simple voxel element at its rest position and aligned with the global axes, these can be easily calculated:

$$w_i = 1 - \frac{p_i - x_i}{v_i} \tag{1}$$

where v_i is the voxel dimension in the i^{th} direction.

Using these weights, trilinear interpolation or extrapolation (if any weight is negative) can be easily applied before rendering the surface mesh.

4 EXAMPLES AND RESULTS

4.1 Model Creation Examples

Various simulation models of varying complexity have been automatically created using the described technique, which include a sphere, cylindrical muscle, skin block with an inclined relatively flat cylindrical muscle, and simple facial model with a simple left zygomatic major muscle. The facial surface, including an inner surface used as an approximation of the skull, was generated using FaceGen[2]. As the face model includes only a single muscle, only the relevant section of the face containing this muscle was voxelised. Figure 7 shows the simulation meshes, and also their rigid nodes by which they were constrained during simulations, and the muscle contraction centres (except for the sphere model which contains no muscles). Figure 8 better shows the surface models used to create the skin block and facial models, and Figure 9 shows the voxelisations of the skin and muscles within these models.

The model creation process generally produced desirable results, for example, with the desired facial model nodes set to rigid on a complex surface, and muscle elements correctly determined. The muscle part of the skin block simulation model, however, shows the importance of using a high enough resolution to capture the surface model shape well. With a too low resolution, surfaces that cross voxel boundaries at small angles may not be captured by the simulation model.

4.2 Model Simulations

We have implemented a non-linear total Lagrangian explicit dynamic (TLED) FE solver on the GPU using the Fermi CUDA architecture to efficiently perform accurate FE simulations. The details of the system, and the formulation and presentation of the TLED algorithm are beyond the scope of this paper, although the TLED algorithm has been presented by Miller et al. [MJLW07].

Various simulations have been performed using the TLED FE system with the models presented in Section 4.1 - a sphere of soft-tissue material, cylindrical muscle, skin block with a muscle, and simple facial model with a single muscle. Each model is composed of reduced-integration 8-node hexahedral elements, and uses a single Neo-Hookean material (for both passive tissue and muscle where applicable) with a Young's modulus of $3000Pa$ and a Poisson ratio of 0.49. A time-step of $0.1ms$ was used for each simulation. Such a small time-step was necessary for stability due

[2] http://facegen.com/

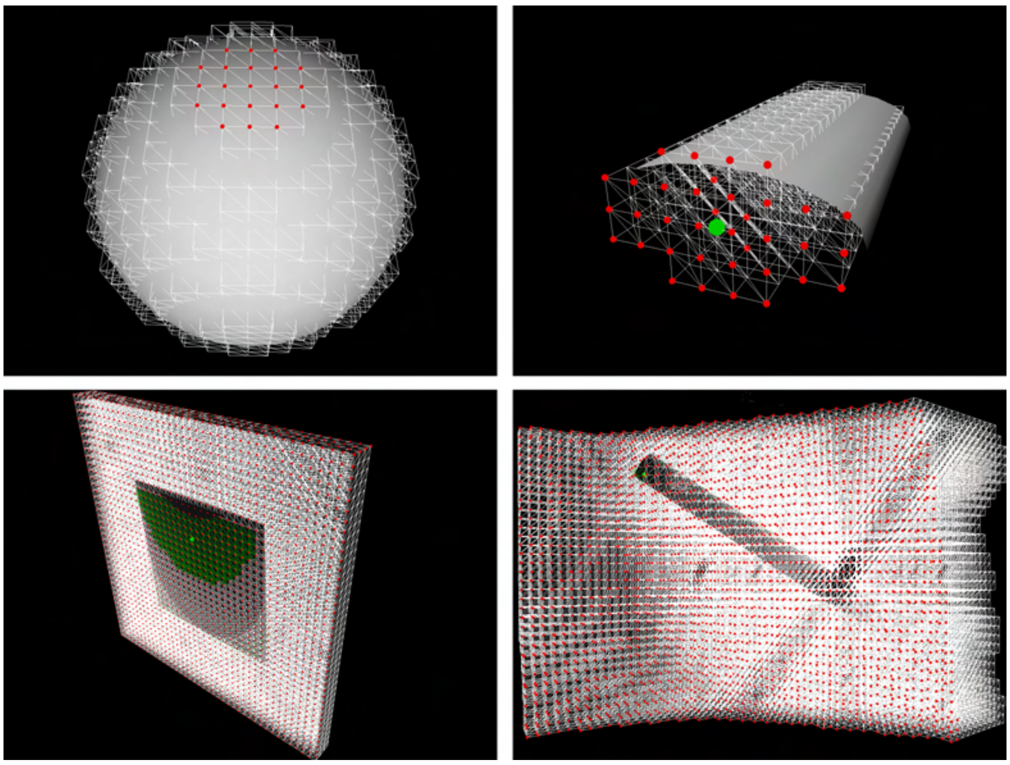


Figure 7: Model rigid node attachments for, from top left to bottom right, a sphere, cylindrical muscle, skin block with muscle, and face, represented by red spheres. The green spheres are muscle contraction centres (where applicable), and muscle elements are coloured green for the skin block and face models.

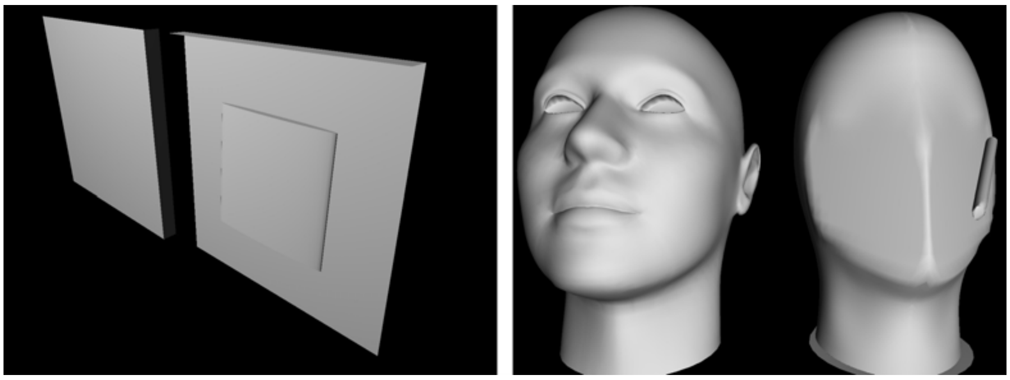


Figure 8: Surface models for the skin block model (left) and facial model (right). Within each image, the left-hand side shows the skin surface, while the right-hand side shows the rigid surfaces to which the muscles are attached.

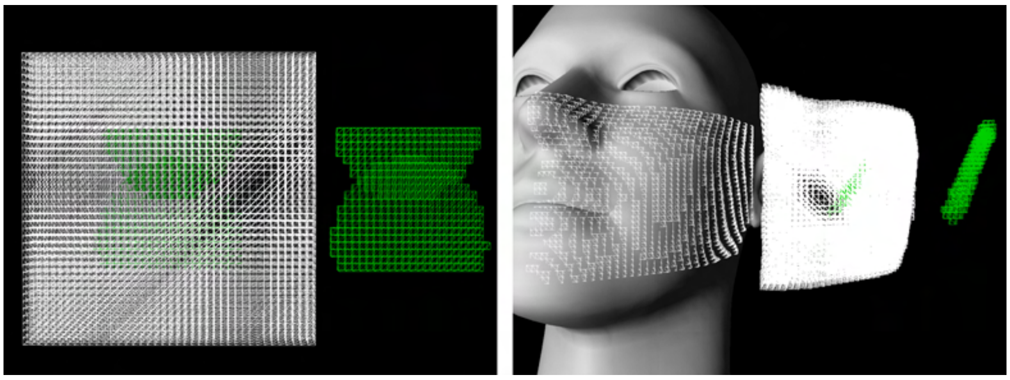


Figure 9: Model voxelisations for the skin block model (left) and facial model (right), where muscle elements are coloured green.

Detail	Sphere	Muscle	Block	Face
Nodes	2849	480	6724	15516
Elements	2176	280	4800	12267
Element Length (*mm*)	12.5	1.5	1.5	2.5
Muscle Contraction	N/A	0.25	0.6	0.4
Time-Step Solution Time (*ms*)	0.187	0.205	0.342	0.542
FPS	34	38	15	7

Table 1: Some model and simulation statistics of simulations performed on a NVIDIA GeForce GTX 460 1GB GPU.

to the small, highly incompressible elements used. The sphere model was influenced by external forces, such as gravity or from user interaction, whereas the other models were influenced purely by internal active muscle stresses. Table 1 shows data on the structure of each model, as well as some simulation statistics, and Figure 10 shows the equilibrium positions for the sphere and face simulations.

Simulations perform well under gravity and user interaction, although it is likely that the face model will be simulated a lot better with more accurate surface models and constitutive laws. For example, it has been shown that it is necessary to model the inhomogeneity and anisotropy of skin to simulate any decent wrinkling effects [HMSH09], rather than as a single isotropic material, as with this work. With the independent muscle simulation, it was necessary to use a small contraction value as there is no surrounding soft-tissue to offer resistance against the muscle active stresses.

4.3 Comparison of Conforming and Non-Conforming Simulation Meshes

For our work, non-conforming hexahedral FE simulation meshes have been used. However, simulation meshes that conform to surface meshes can also be used for FE simulations. Table 2 summarises the creation and use of conforming and non-conforming hexahedral meshes with a CUDA TLED FE system.

Experiments using a conforming (created using IA-FEMesh[3]) and a non-conforming hexahedral simulation mesh for a sphere have demonstrated the performance and stability advantages of non-conforming simulation meshes. As shown by Table 3, using a non-conforming simulation mesh with a bound higher resolution surface mesh has led to performance increases of almost 2x compared to using a conforming simulation mesh with roughly the same number of nodes and elements. Also, stable simulations were able to be performed using a considerably higher Young's modulus and Poisson ratio with a non-conforming mesh, which is necessary for simulating, for example, the stiff properties of the epidermal skin layer, and incompressible soft-tissue material. On the downside, depending on element size, accuracy is likely to be reduced using a non-conforming simulation mesh.

[3] `https://simtk.org/home/ia-femesh`

5 CONCLUSIONS

This work has presented a process for creating animatable non-conforming hexahedral FE simulation models to which the object surface meshes are bound. Using regular or irregular voxel sizes, the process involves discretising the volume enclosed by surface meshes, possibly with internal surfaces, and separating elements accordingly. Rigid nodes on the simulation mesh can also be automatically determined, as can muscle parameters. Some soft-tissue models, including a simple facial model with a muscle, have been created to demonstrate the process, which could also be used to produce models for more complex surface meshes. A GPU TLED FE system has been used to produce some example real-time and interactive simulations with the created models, showing that a surface mesh can be easily transformed into a complex animatable physically-based model. Finally, the various advantages provided by using a non-conforming simulation mesh have been outlined, including model creation, speed, memory and stability advantages.

Various improvements could currently be made to the model creation process. For example, the voxelisation process could be based on the the proportion of a voxel that is enclosed by a volume, which could be used to determine whether a voxel is inside the volume, and to calculate the element material properties, similar to the approach by Lee et al. [LST09]. Using such an approach would also require the rigid node attachment algorithm to be modified. Future work will also focus on creating and animating a more accurate full facial model with inhomogeneous anisotropic viscoelsatic materials.

6 REFERENCES

[AZ10] Olusola O. Aina and Jian Jun Zhang. Automatic Muscle Generation for Physically-Based Facial Animation. In *SIGGRAPH 2010 Posters*, pages 105:1–105:1, Los Angeles, California, USA, 2010. ACM.

[Bis06] J. E. Bischoff. Reduced Parameter Formulation for Incorporating Fiber Level Viscoelasticity into Tissue Level Biomechanical Models. *Ann. Biomed. Eng.*, 34(7):1164–1172, 2006.

[BJTM08] Giuseppe Barbarino, Mahmood Jabareen, Juergen Trzewik, and Edoardo Mazza. Physically Based Finite Element Model of the Face. In *Proc. ISBMS 2008*, pages 1–10, London, UK, 2008. Springer-Verlag.

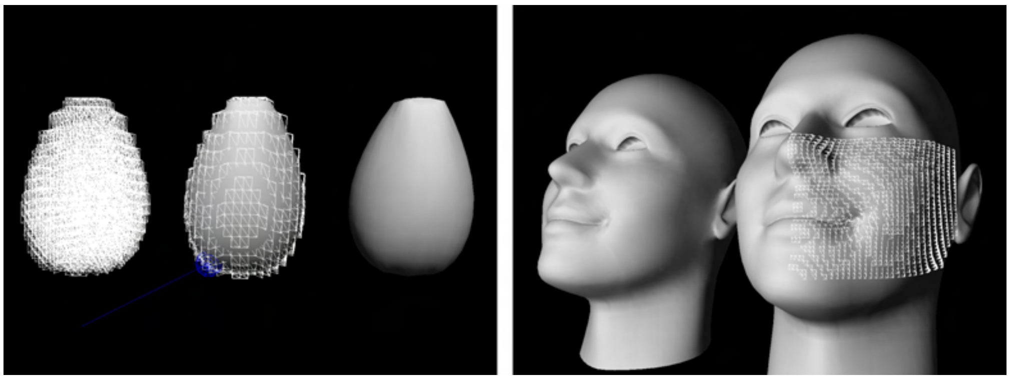


Figure 10: Simulation equilibrium states for sphere under gravity and interaction (where the blue line represents the interactive force), and a face with a left zygomatic major muscle.

Criterion	Conforming	Non-Conforming
Volumetric Model Generation	Complex, involving manual input	Simple and automatic
Simulation Speed	Difficult to achieve speed-ups	More efficient memory accesses and global memory coalescing can be achieved for increased speed
Memory Requirements	High, especially with large models	Low as some values are the same for each element
Stability	Badly-shaped or tiny elements can greatly reduce stability	All elements are regular for better stability
Accuracy	Mesh more closely matches the object being simulated	Less accurate approximation of the object being simulated may lead to less accurate results

Table 2: A summary of the differences between using a conforming and non-conforming regular hexahedral simulation mesh with a CUDA TLED FE system.

Criterion	Conforming	Non-Conforming
Nodes	2736	2849
Elements	2197	2176
Minimum Element Length (*mm*)	<12.5	12.5
Time-Step Solution Time (*ms*)	0.48	0.246
Stability with a 0.1*ms* timestep:		
Maximum E (*kPa*)	3	10
Maximum v	0.4	0.47

Table 3: Results of a comparison between using a conforming and a non-conforming regular hexahedral simulation mesh of a spherical object, constrained by its upper nodes and acting under gravity, with a Neo-Hookean material model (with a default Young's modulus, E, of 3*kPa* and Poisson ratio, v, of 0.4). Simulations were performed on a NVIDIA GeForce GTX 460 1GB GPU.

[BWL+10] Liliana Beldie, Brian Walker, Yongtao Lu, Stephen Richmond, and John Middleton. Finite element modelling of maxillofacial surgery and facial expressions – a preliminary study. *Int. J. Med. Robot. Comput. Assist. Surg.*, 6(4):422–430, 2010.

[Coo98] Lee Cooper. *Physically Based Modelling of Human Limbs*. PhD thesis, University of Sheffield, 1998.

[Cou05] Alasdair D. Coull. *A Physically-Based Muscle and Skin Model for Facial Animation*. PhD thesis, University of Glasgow, UK, 2005.

[CP06] C. Chen and E. C. Prakash. Physically based facial expression synthesizer with performance analysis and GPU-aided simulation. In *Proc. CyberGames 2006*, pages 171–176, Perth, Australia, 2006. Murdoch University.

[CPL00] Béatrice Couteau, Yohan Payan, and Stéphane Lavallée. The mesh-matching algorithm: an automatic 3D mesh generator for finite element structures. *J. Biomech.*, 33(8):1005–1009, 2000.

[DGW11] Christian Dick, Joachim Georgii, and Rudiger Westermann. Simulation Modelling Practice and Theory. *Simul. Model. Pract. Theory*, 19(2):801–816, 2011.

[fLhLfT11] Meng fei Li, Sheng hui Liao, and Ruo feng Tong. Facial hexahedral mesh transferring by volumetric mapping based on harmonic fields. *Comput. Graph.*, 35(1):92–98, 2011.

[HMSH09] A. Hung, K. Mithraratne, M. Sagar, and P. Hunter. Multilayer Soft Tissue Continuum Model: Towards Realistic Simulation of Facial Expressions. volume 54, pages 134–138, Paris, France, 2009.

[ISS09] Yasushi Ito, Alan M. Shih, and Bharat K. Soni. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. *Int. J. Numer. Methods Eng.*, 77(13):1809–1833, 2009.

[KHS01] Kolja Kähler, Jörg Haber, and Hans-Peter Seidel. Geometry-based Muscle Modeling for Facial Animation. In *Proc. GI 2001*, pages 37–46, Ottawa, Ontario, Canada, 2001. Canadian Information Processing Society.

[KPB08] Ashok V. Kumar, Sanjeev Padmanabhan, and Ravi Burla. Implicit boundary method for finite element analysis using non-conforming mesh or grid. *Int. J. Numer. Methods Eng.*, 74(9):1421–1447, 2008.

[KRG$^+$02] R. M. Koch, S. H. M. Roth, M. H. Gross, A. P. Zimmermann, and H. F. Sailer. A Framework for Facial Surgery Simulation. In *Proc. SCCG 2002*, pages 33–42, Budmerice, Slovakia, 2002. ACM.

[KSY08] O. Kuwazuru, J. Saothong, and N. Yoshikawa. Mechanical approach to aging and wrinkling of human facial skin based on the multistage buckling theory. *Med. Eng. & Phys.*, 30(4):516–522, 2008.

[Lam09] B. P. Lamichhane. Mortar Finite Elements for Coupling Compressible and Nearly Incompressible Materials in Elasticity. *Int. J. Num. Anal. Model.*, 6(2):177–192, 2009.

[Lo12] S. H. Lo. Automatic merging of hexahedral meshes. *Finite Elem. Anal. Des.*, 55:7–22, 2012.

[LST09] Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. Comprehensive Biomechanical Modeling and Simulation of the Upper Body. *ACM Trans. Graph.*, 28(4):99:1–99:17, 2009.

[MBTF03] Neil Molino, Robert Bridson, Joseph Teran, and Ronald Fedkiw. A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra. In *Proc. IMR12*, pages 103–114, Santa Fe, New Mexico, USA, 2003. Sandia National Laboratories.

[MHHR06] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position Based Dynamics. In *Proc. VRIPHYS 2006*, pages 71–80, Madrid, Spain, 2006.

[MJLW07] Karol Miller, Grand Joldes, Dane Lance, and Adam Wittek. Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. *Commun. Numer. Methods Eng.*, 23(2):121–134, 2007.

[MSNS05] Wouter Mollemans, Filip Schutyser, Nasser Nadjmi, and Paul Suetens. Very fast soft tissue predictions with mass tensor model for maxillofacial surgery planning systems. In *Proc. CARS 2005*, pages 491–496, Berlin, Germany, 2005. Elsevier.

[NRP11] M. Nieser, U. Reitebuch, and K. Polthier. CUBECOVER – Parameterization of 3D Volumes. *Comp. Graph. Forum*, 30(5):1397–1406, 2011.

[RP07] Oliver Röhrle and Andrew J. Pullan. Three-dimensional finite element modelling of muscle forces during mastication. *J. Biomech.*, 40(15):3363–3372, 2007.

[SG05] Hang Si and Klaus Gärtner. Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations. In *Proc. 14th Int. Meshing Roundtable*, pages 147–163, San Diego, California, USA, 2005. Sandia National Laboratories.

[SKO$^+$10] Matthew L. Staten, Robert A. Kerr, Steven J. Owen, Ted D. Blacker, Marco Stupazzini, and Kenji Shimada. Unconstrained plastering – Hexahedral mesh generation via advancing-front geometry decomposition. *Int. J. Numer. Methods Eng.*, 81(2):135–171, 2010.

[SNF05] Eftychios Sifakis, Igor Neverov, and Ronald Fedkiw. Automatic Determination of Facial Muscle Activations from Sparse Motion Capture Marker Data. *ACM Trans. Graph.*, 24(3):417–425, 2005.

[SSLS10] Matthew L. Staten, Jason F. Shepherd, Franck Ledoux, and Kenji Shimada. Hexahedral Mesh Matching: Converting non-conforming hexahedral-to-hexahedral interfaces into conforming interfaces. *Int. J. Numer. Methods Eng.*, 82(12):1475–1509, 2010.

[SZM12] Lu Sun, Guoqun Zhao, and Xinwu Ma. Quality improvement methods for hexahedral element meshes adaptively generated using grid-based algorithm. *Int. J. Numer. Methods Eng.*, 89(6):726–761, 2012.

[TCO08] Zeike A. Taylor, Mario Cheng, , and Sébastien Ourselin. High-Speed Nonlinear Finite Element Analysis for Surgical Simulation Using Graphics Processing Units. *IEEE Trans. Med. Imaging*, 27(5):650–663, 2008.

[TW90] Demetri Terzopoulos and Keith Waters. Physically-Based Facial Modeling, Analysis, and Animation. *J. Vis. Comput. Animat.*, 1(2):73–80, 1990.

[TZT09] C. Y. Tang, G. Zhang, and C. P. Tsui. A 3D skeletal muscle model coupled with active contraction of muscle fibres and hyperelastic behaviour. *J. Biomech.*, 42:865–872, 2009.

[Wat87] Keith Waters. A muscle model for animation three-dimensional facial expression. *SIGGRAPH Comput. Graph.*, 21(4):17–24, 1987.

[WJC$^+$10] Adam Wittek, Grand Joldes, Mathieu Couton, Simon K. Warfield, and Karol Miller. Patient-specific non-linear finite element modelling for predicting soft organ deformation in real-time; Application to non-rigid neuroimage registration. *Prog. Biophys. Mol. Biol.*, 103(2–3):292–303, 2010.

[XLZH11] Shaoping Xu, Xiaoping P. Liu, Hua Zhang, and Linyan Hu. A Nonlinear Viscoelastic Tensor-Mass Visual Model for Surgery Simulation. *IEEE Trans. Instrum. Meas.*, 60(1):14–20, 2011.

[ZHB10] Yongjie Zhang, Thomas J. R. Hughes, and Chandrajit L. Bajaj. An Automatic 3D Mesh Generation Method for Domains with Multiple Materials. *Comput. Methods Appl. Mech. Eng.*, 199(5–8):405–415, 2010.

[ZHD06] Stefan Zachow, Hans-Christian Hege, and Peter Deuflhard. Computer-Assisted Planning in Cranio-Maxillofacial Surgery. *J. Comp. Inf. Technol.*, 14(1):53–64, 2006.

3D modelling and analysis: ISO standard tools for air traffic

Axel François, Romain Raffin, Marc Daniel
Aix-Marseille University, LSIS
UMR CNRS 7296, Domaine
Universitaire de Saint-Jérôme,
13397 Marseille, FRANCE
axel.francois, romain.raffin,
marc.daniel@lsis.og

Jagannath Aryal
University of Tasmania,
School of Geography and
Environmental Studies,
Private Bag 78, Hobart,
Tasmania 7001,
AUSTRALIA
jagannath.aryal@utas.edu.au

Abstract

The Geographic Information Systems (GIS) applied to aviation use mostly modelling and analysis in 2D. Nevertheless, a tendency to represent and analyse in three-dimensions begins to emerge. It requires new descriptions and new operative tools for 3D objects in GIS. As an association of users and software producers, the Open Geospatial Consortium (OGC) has defined an ISO standard to describe two-dimensional and three-dimensional geometric objects. However, this standard does not permit a description of common Computer Aided Design (CAD) objects, a vital task remains for the modelling of 3D data and their uses, in the context of analysis (spatial query) or with the proposals of new primitives. In this research, our goal is to build a computing library fully compatible with ISO standard especially allowing characterising any gaps or parts requiring further development. In this paper, we present a solution validated by a use-case for modelling and analysis of aerial traffic in 3D on an ellipsoid of revolution that follows the ISO standard. In order to demonstrate whether new geometric objects that we propose are effective, in this simple approach, we have established a complete processing chain.

Keywords

ISO 19107, Air traffic, Geographical analysis, Geometric modelling, Analysis Dispatcher

1 INTRODUCTION

The Geometric modelling that is currently implemented in the GIS is mostly data and algorithms driven for 2 and 2.5 dimensions. However, the current studies on market analysis and representation of data in 3D space showed the increasing scenario of geometric modelling in 3D GIS [Ste05]. As shown in the work of Zlatanova [Zla08] many activities are still waiting concrete 3D GIS solutions. Current studies on GIS spatial objects show a categorisation following the application domain. The work of Danahy [Dan97] defines five groups for a 3D city model (e.g. vegetation, buildings, public utilities, traffic network and telecommunications). Nevertheless, it is also possible to separate from another group independently of a real representation (e.g. legal limits, institution, companies) for example in the case of cadastral modelling and analysis in 3D by Billen and Zlatanova [BZ03]. Currently, consumer software such as Google Earth [1] or World Wind [2] have defined up the first opportunities visualisation of 3D geo-referenced data. Despite these solutions provide no tool for 3D analysis.

The primary cause of developmental delay of 3D analysis part in GIS is the modelling multiplicity solutions and exchange, because each software is using their own method add is a common geometric modelling must be defined as in CAD domain with STEP [TC 94] standard to facilitate the efficiency and depiction of real world. The work of Zlatanova [Zla99] has a first approach to the VRML [ISO04a] format standard. Nonetheless, only the display is in a standard format. The recovery and data analysis use a specific format content in the application and there is no end-to-end standardisation. This standardisation for GIS domain is supported by the OGC [3] through the “Geometry/Topology” standard ISO 19107 [ISO03]. This standard is currently under review by the Working Group “Simple Feature” as a complement of the revision of ISO 19125 [ISO04b].

Our work is to implement a Globe3D platform with analysis tools fully compatible with the ISO 19107

[1] www.google.com/intl/en/earth
[2] http://worldwind.arc.nasa.gov
[3] OGC: Non-profit organization created to address the problem of interoperability between systems that process geospatial data.

standard. Compliance with this standard also provides interoperability between modules (such as SWE [4] [ISO10], WCS [5], GML [ISO07b] ...). This application aims to demonstrate the capabilities of our modelling approaches and 2D/3D analysis in the field of GIS. In this context, we organise this paper as follows: in section 2, we present ISO 19107 standard covering the geometry of the object including spatial analysis in ISO 19107. Section 3 focuses on the development of a "Decision Tool", allowing the management of modelling and analysis of standard that takes into account output requirements and architecture of the modules. In section 4, we present our Globe3D platform that uses the ellipsoidal representation for the Earth geometry. Globe3D platform is superior to the current visualisation softwares which only use a simple sphere as an Earth surface approximation. Section 5 deals with the use case of air traffic taking into account the analysis and trajectory modelling with examples. In section 6, we describe a parametric curve which is usefull for aircraft trajectory. Specific focus is there for traffic representation in 3D space in section 6.1. The analysis and representation of air traffic are typically defined in a 2D space. The trajectories of an aircraft are represented by a set of geo-localised positions and a linear interpolation between each point (polyline) is performed. However, it is possible to use a different interpolation method, as for example, Catmull-Rom [Twi03] and NURBS [PT97]. The computational development for air analysis and visualisation in 3D space is made in collaboration between Geomatys Company and LSIS CNRS 7296 public lab, in the open source frameworks GeoAPI 3.0 [6] and Geotoolkit [7]. As we based our work on these public frameworks, we will release our code to permit GIS community to use 3D geometries and analysis in their specific applications. In section 7, we conclude the paper with the contributions we made and our future direction in this exciting field of 3D modelling and analysis.

2 ISO 19107 STANDARD

2.1 General definition

The concept of an object in ISO 19107 is defined by three strongly linked parts. The first two parts allow a description of objects defined hierarchically according to their spatial dimensions (point, curve, surface, volume) and their topologies (node, edge, face, solid) which are already been used by CAD industries. Each geometric ISO object is preceded as "GM_" and topological link by "TP_". Geometric objects have however a third necessary characteristic: they are linked to geo-referenced coordinates (e.g. coordinate references on sphere, plane, geoid ...) with vector basis and mathematical projection. Each geometry is related to a Coordinate Reference System (CRS), as for an example the CRS WGS84. Besides, our work has made a proposal to add a new type of surface, constructed by revolution, which can possibly define the earth ellipsoid.

The standard also defines spatial analysis operations on geometric objects. There is no analytic (generic) way to process these methods, as their computation depend on the selected reference system and geometric description. Thereby, the spatial operations module supply analysis methods for aerial traffic, following ISO 19107 standard.

2.2 Geometry in ISO 19107

A geometric object respecting the ISO 19107 standard is represented by a GM_Object, defining a set of functions. The standard defines a hierarchical construction of geometric objects. A GM_Object may be described with different types. GM_Primitive object defines only a single object with its attributes, specific method "boundary" and geo-referenced position (GM_DirectPosition). There are 4 primary GM_Primitive (GM_Point, GM_Curve, GM_Surface and GM_-Solid) with 31 coordinates objects (GM_LineString, GM_BSpline, GM_Triangle, ...). Further, we also describe a new GM_Object with GM_RevolutionSurface forward in this paper.

Each object has associated functions (distance, centroid, envelope, ...) and geometric description. The mathematical methods implemented by these functions may vary depending on the referencing system used. The calling function itself is not altered, but the calculus method differs from a CRS to another. These mechanisms of evaluation are not defined in the ISO 19107 standard. Moreover, the construction methods and the interpolations used for geometric objects are not described in this standard. The gap in the standard is due to the many degrees of freedom left for its implementation.

2.2.1 *Spatial analysis in ISO 19107*

This standard also defines spatial analysis methods on geometrical objects like "intersection", "difference" and "contains". As shown in Figure 1, the methods are described in two groups: "predicates" and "operations", depending on their action. Tests done by "predicates" return logical (i.e. boolean) value as in "intersect", whereas the "intersection" method returns a resulting geometric object. These two groups are complementary as "predicates" indicate the operation feasibility. The following section details the mechanism of a "decision tool" in choosing the method of

[4] Sensor Web Enablement

[5] Web Coverage Service

[6] www.geoapi.org

[7] www.geotoolkit.org

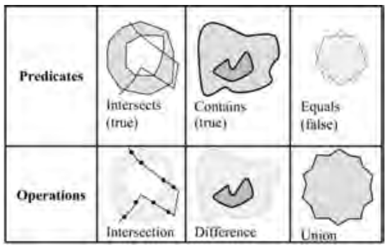


Figure 1: Predicates and operations analysis

modelling and analysis following its CRS or object type.

3 DISPATCHER MECHANISM

As we want to provide a library founded on ISO 19107 with an automatic processing phase analysis, we set up a module "Dispatcher" which has a goal to harmonise CRS and geometric objects definitions to prepare generic analysis phase. This phase exists in the availability of representing different types of geometric objects in a common mathematical model. Besides, the Dispatcher must take into account the constraints that are described in the standard output. That is to say, after all step of analysis and processing, the resulting object must be conformed to specification of the ISO 19107 standard prior to the translation in user specific format depending on the intended usage (e.g. visualisation, printing, storage ...).

3.1 Output requirements

The output model of the analysis step must be described in the CRS of the first object (ISO 19107 requirement). Thus, this strong restriction is taken into account by our "Dispatcher" module. It must determine whether the transformation from a CRS to another is possible. The translation possibilities depend on the libraries implementation based on ISO 19111 [ISO07a] standard. Another role of our "Dispatcher" module is to let it go in the output for GM_Object which is in compliance with the ISO 19107. For example, the intersection between a circle and a line (if it exists: predicate "intersect" returns true value) may be a set of GM_Point or a single one. In the case of an "intersection" between two ISO 19107 spheres (e.g. GM_Sphere) many resulting object types are possible, a GM_Point for just one intersection whereas polyline (GM_LineString), NURBS curve (generic object) for more than one intersections as described in François et al. [FRD10b].

3.2 Architecture

Our standard implementations rely on an architecture using input/output modules built around a core module containing the description of the ISO objects. Figure 2 describes in detail the three modules contained in the "Dispatcher" architecture, one that formalises the

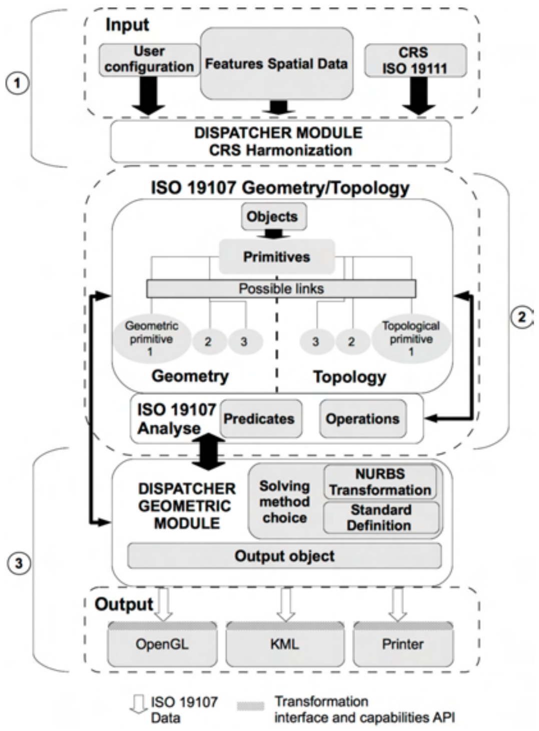


Figure 2: Dispatcher/ISO 19107 interaction

input data (module #1), a module that constructs and processes standardised data (module #2), and one that prepares the output data to user front end (module #3).

The first module is designed to prepare the data for the ISO 19107 part, taking into account user output specifications and reference of the main CRS. These last operations are done by the sub-module "CRS Harmonization" with the Geotoolkit library. However, if CRS source could not be transformed into a CRS destination or reverse transformation is not possible, the analysis process is cancelled. Figure 3 shows that the "Dispatcher", in a first step, verifies the compatibility of CRS objects A and B with the output CRS (CRS of object A) and also between them. The ISO 19111 library determines the list of compatible CRS. Unlike the JTS [DA03] library that uses only one cartesian space (e.g. WGS84 cartesian) for modelling and analysis, our "Dispatcher" allows the evolution of operations to another reference system (ellipsoid, spherical, ...). For example, an operation "distance" is already extended to ellipsoidal space.

The second module is the geometric kernel, it is composed of the ISO 19107 part with two submodules, "Geometry/Topology" and "Analysis" that strongly interact. The "primitives" must build the objects according to ISO standard specification (ex: GM_BSplineCurve object).

It must use the geometrical part of the standard, but also the topological part as well. An analysis between

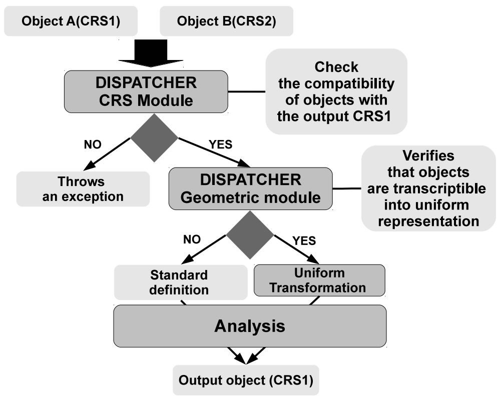

Object A(CRS1)
Object B(CRS2)
DISPATCHER CRS Module
Check the compatibility of objects with the output CRS1
NO
YES
Throws an exception
DISPATCHER Geometric module
Verifies that objects are transcriptible into uniform representation
NO
YES
Standard definition
Uniform Transformation
Analysis
Output object (CRS1)

Figure 3: Dispatcher transformation process

two objects implies a call to "predicate" module and depending on the result of the operation, a new object ISO is created. Special attention should be paid to CRS transformations: it causes a degeneration that spreads after each operation and can result in analysis errors. The architecture presented here does not cover actually this type of problem, but it is capable enough to calculate maximum error transformations via Geotoolkit library.

The sub-module "Dispatcher Geometric" manages the analysis module, it determines the most appropriate method to perform the processing. As shown in the detailed schema (see Figure 3), there are two possibilities: objects can be converted to a uniform representation, or they are of basic types (GM_LineSegment, GM_Point). In a worst case scenario, objects can not be converted and consequently the analysis is based only on their standard definitions. The output module takes into account the constraints of output interfaces with features such as printer, rendering engine using libraries like OpenGL, Java3D, file export formats (XML, KML [8], ...), or our software Globe3D. In the following section, we present the first step in developing our platform in modelling the Earth following an ellipsoid geometry.

4 MODELLING AN ELLIPSOID FOR THE EARTH GEOMETRY

4.1 Using a revolution surface

Current applications (World Wind, Google Earth or ArcGIS Explorer [9]) model the terrestrial globe by a simple sphere (GM_Sphere). The sphere representation is not sufficient in current methods. As CRS WGS84/World Mercator is widely used and rely on an ellipsoidal description of Earth surface, retro-projection is needed to obtain spherical coordinates from ellipsoidal one, eroding data quality.

[8] Keyhole Markup Language

[9] www.esri.com/software/arcgis/explorer/

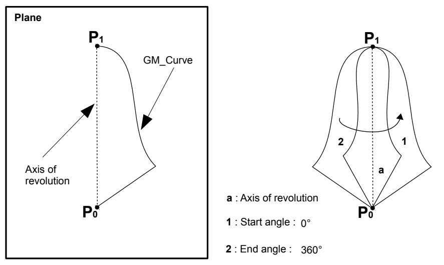


Figure 4: Definition of revolution surface

Moreover, the advantage of an ellipsoid based representation is the multiple possible formulations, like parametrical surface or implicit formulation with:

$$x^2/a^2 + y^2/b^2 + z^2/c^2 + 1 = 0$$

These formulations are used in the analysis and in the calculations.

François et al. [FRD10a] described a conversion process of a rational GM_Sphere surface (NURBS) to describe the ellipsoidal. Our approach here is to introduce a new surface family with revolution surface that was described in Shukla [Shu10] (family associated with GM_RevolutionSurface). It permits us to define the ellipsoidal Earth with a simple GM_Arc object. The new surface allows the ISO 19107 standard to create open or close surfaces. Its shape is defined by the rotation around an axis of a GM_Primitive type: GM_Curve. This type of surface creates a wide range of surfaces depending on the selected curve and its orientation.

The curve used for the revolution is called the "generating curve", in the case of a curve defined in 3D space, it should be reduced to planary projection to achieve revolution. However, the generator curve, GM_Curve, can be built by a set of sub-curves as shown in Figure 4.

It is possible to obtain an open GM_surface by specifying starting and ending angles of rotation. Nevertheless, it should lie between 0 and 360 degrees (see Figure 4). The positive value of the surface is defined by its rotation, and it defines outer normals, default value is counter clockwise. In this example, the result is a closed surface.

4.1.1 Integration to standard

This new GM_Object belongs to the surface family type and is naturally integrated as GM_Surface object. The class GM_Revolution Surface defines a surface of revolution. Two constructors are allowed, the first takes a parameter curve generator and an array containing two revolution angles (first and last). The second defines an axis of revolution, unlike the first constructor that uses OY by default. There is also a set of 4 subclasses that inherit directly from GM_RevolutionSurface. Their goals are to specify simple shapes (cylinder, cone, sphere). The current standard does not define an axis for the surface of revolution, consequently it was added to a new class GM_Axis. To define an ellipsoid with a surface of revolution, one can use the ISO GM_-Arc object as generator curve. This object can be constructed in two different ways with only three positions, or using the offset of the midpoint (bulge) between the start and end points.

4.1.2 Advantages

The first contribution is in the quantity of data used to define an object through revolution method versus classical way. Another advantage is the possibility to create an open surface, like an hemisphere with the same formalism.

4.1.3 Harmonization of generator curve GM_-Arc

In order to generalize mathematical operations between curve objects, we describe an arc with a parametric curve (see in François et al. [FRD10b]). The object ISO 19107 is transcribed into a rational representation with Non-Uniform Rational Basis Spline (NURBS) as in Faux and Pratt [FP79] to facilitate the processing analysis module ("Dispatcher"). This representation uses a control polygon to define the curve. Let $(x_0,...,x_{n-1})$ be coordinates of a point in the working space. These points can be represented in homogeneous coordinates $(x'_0,...,x'_n)$ with $x_i = x'_i/x'_n$ and $x'_n \neq 0$. The NURBS curve is defined as a perspective projection centred to the origin and the hyperplane $x_n = 1$. The general expression of a NURBS curve is given by equation 1 with the basis functions $N_{i,k}(t)$ and weight w_i (a more detailed definition is available in Piegl and Tiller [PT97]).

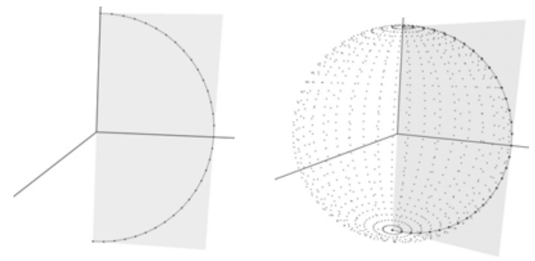
$$C(t) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,k}(t)}{\sum_{i=0}^{n} w_i N_{i,k}(t)} \text{ for } C(t), P_i \in \mathbb{R}^3 \quad t \in [0,1] \tag{1}$$

The first advantage of using a NURBS representation for revolution surface is to control the generating curve parametrisation. It is then possible to obtain any position on the curve with t parameter.

The control polygon can be defined with 3 or 4 vertices, it uses the tangent to the circle of R radius (see Sederberg [Sed09] and Lu [Lu09]).

4.2 Terrestrial globe

As we get a precise parametric definition of an ellipsoid, we use this model to define a globe in 3D space respecting the WGS84 coordinate reference system. An important advantage of parametric description is that discretisation can be made according to process constraints (precision, speed, density, ...).



(a) Generator curve GM_-Arc (b) Revolution application

Figure 5: Example of a spherical object with GM_RevolutionSurface

The two images in Figure 5 show the different steps used for globe modelling. The rendering application uses a geocentric coordinate referencing system, allowing three spatial coordinates (X, Y and Z). Z coordinate is used as an elevation value. As shown in Figure 5(a), the generator curve used is GM_Arc in the XY plane defined by three positions (GM_DirectPosition), the middle position of the arc uses the equatorial radius (6 378.137 km) given by semi-major axis a of WGS84 CRS. The first and last positions are centred on the Y axis and distance from the semi-minor axis b (6 356.752 km) from the origin is calculated with a flattening factor $f = (298.257)$ also given by the CRS:

$$f = \frac{a-b}{a} \text{ where } b = a(1-f)$$

Figure 5(b) shows the result of a rotation $\theta = 2\pi$ applied to a GM_Arc object contained in a plane, and that creates a GM_SurfaceRevolution. This GM_Surface can be rendered with a set of GM_Triangle or GM_-Polygon due to new implemented functions "asToTriangles" and "asToPolygons". One advantage of this discretisation by ISO 19107 library (e.g. GM_Triangle) is to pre-calculate the normals of each face before sending data to the rendering module. It can also permits to manage the discretisation method (homogeneous, based on curvatures, distances, ...).

Once the revolution surface defined and discretised respecting interoperable methods, the results thus obtained are then sent to a 3D engine, based on the widely used OpenGL API and Ardor3D, which is useful to handle 3D context events and objects visualisation. Figure 6 shows the result of our Earth ellipsoid in the rendering module. We add a "Blue Marble" [10] NASA [11] texture as adding on a GM_Surface. Nevertheless, it is possible to observe as predicted by the WGS factor a small crushing of the Earth at poles. We also represent the Earth's atmosphere with another GM_SurfaceRevolution encompassing the geoid of the Earth.

[10]www.earthobservatory.nasa.gov/Features/BlueMarble
[11]www.nasa.gov

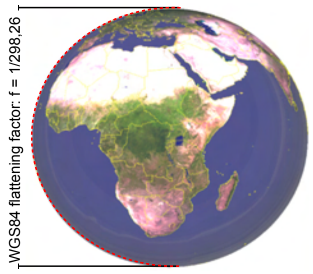
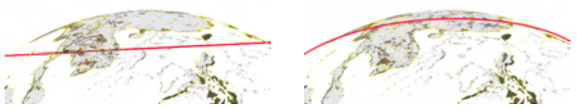


Figure 6: ISO Earth ellipsoid with country boundaries and "Blue Marble" texture



(a) Line discretisation crosses the Earth

(b) Line discretisation following earth curvature

Figure 7: Example of curve discretisation if parametric curves are used

5 TRAFFIC REPRESENTATION

5.1 Analysis and trajectory modelling

We import aircraft trajectories in our 3D globe with a set of GM_Curve primitives. These trajectories are loaded from a KML (see Reed [Ree07]) file, each of them is defined with a set of line string objects. Thanks to the interoperability between KML and ISO 19107 standard, these elements are then translated into GM_Curve parametric, that can be discretised in a set of GM_LineString.

As shown in Figure 7(a), there exist cases where the trajectory crosses the Earth. This is due to excessive distance between the positions that composes a GM_Segment (see Figure 7(b)). In this case it is necessary to calculate a parametrisation in accordance with the curvature of the ellipsoid. Equation 2 defines the equation of the Earth ellipsoid.

$$\begin{cases} x = 6378,137\cos(\alpha)\cos(\beta) & \\ y = 6356,752\cos(\alpha)\sin(\beta), & \text{for } -\dfrac{\pi}{2} \leq \alpha \leq \dfrac{\pi}{2} \\ z = 6378,137\sin(\alpha), & \text{and } -\pi \leq \beta \leq \pi \end{cases} \tag{2}$$

Let $P = \{P_i\}_{i=0}^{n}$ associated to each P_i the pair of angles (α_i, β_i) and $\forall i \in \{0, \dots, n\}$, we find that:

$$\alpha_i = \arcsin\left(\frac{z_i}{6378,137}\right)$$

and

$$\beta_i = \begin{cases} \arccos\left(\dfrac{x_i}{6378,137\cos(\alpha_i)}\right), & \text{or} \\ \arcsin\left(\dfrac{y_i}{6356,752\cos(\alpha_i)}\right) \end{cases}$$

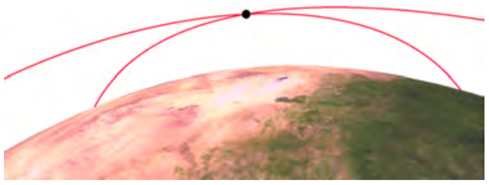


Figure 8: Examples of aerial trajectory analysis with trajectories intersection

N be the number of segments between two points P_j and P_{j+1}. The curvilinear abscissa L for each segment of $C(t)$ curve is defined by the following equation:

$$L_i = \int_{P_j}^{P_{j+1}} c(t)dt$$

and the overall curve is given by:

$$L_c = \sum_{i=0}^{N-1} L_i, \quad \text{with } N+1$$

Thus, it is determined by a discretisation on curvilinear abscissa distance L_c with p (for example, one point for a range of 500 m):

$$segmentsNb = {}^{L_c}/_{p}$$

As presented in Section 4.1.3, we use the "Dispatcher" which prepares the processing of the spatial analysis between two GM_Object (GM_Point, GM_Curve, GM_Surface, ...). Every curve representing the trajectories of the aircraft is transformed by the "Dispatcher" in a generic NURBS object. This processing step depends on the nature of the GM_Object used for analysis. For example, in case where the trajectories are represented by a GM_Arc or set of arc with GM_ArcString.

5.2 First examples of analysis and modelling

As shown in Figure 8, the application displays an example of analysis between two trajectories in 3D space, the example illustrates "intersect" and "intersection" operations contained in ISO 19107. The result is a GM_point object centre of a small Ardor3D [12] sphere. We can also perform this analysis across the complete aerial network.

The Globe3D application also supports analysis with another GM_Object. The decision tool ("Dispatcher") will determine whether it exists intersections with bounding boxes, and then run the analysis between

[12] www.ardor3d.com

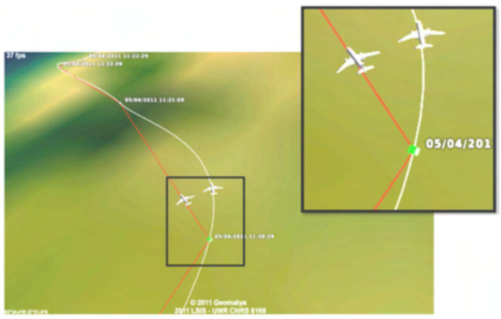


Figure 9: Multi-trajectories definition (linear polyline and spline curve)

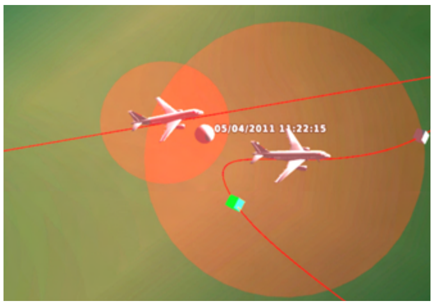


Figure 10: Analysis examples for influences zones along two trajectories (collision detected)

GM_Object. Resulting intersection points are given in 3D space along with recomputing altitude given by transformation matrix of WGS84 3D CRS (e.g. EPSG:4329 [13]).

6 PARAMETRIC CURVE FOR AIRCRAFT

In most current software, the trajectories of aircraft are modelled by linear interpolation (polylines) between positions. Our application allows the use of ISO 19107 parametric interpolation curves, of Catmull-Rom type [DB88][Twi03]. In commercial aircraft navigation, arcs are used to define trajectories, these primitives are based on underlying parameter objects.

The curve passes through all the points that define the trajectory. A new derivative is calculated at each new point of the curve and these derivatives can determine the direction of the aircraft. It forced also to keep C^1 continuity (tangency). In addition, this level of continuity allows us to obtain a smooth curve. This object is defined in ISO 19107 by GM_CubicSpline.

Figure 9 shows the two types of curve for the same positions series. These curves share the same timestamps, this time value is defined in the KML file for each position. The red line represents the linear interpolation GM_LineString, the white curve is the parametric interpolation specified with GM_SplineCurve.

Linear interpolation involves lack of accuracy. Conversely, a parametric interpolation type Catmull-Rom spline can better fit the data, GM_SplineCurve representation also provides continuity at least C^1, which gives a smooth curve. We can conclude that a parametric curve is more relevant than the linear representation because it provides more degrees of freedom. However, the curves Catmull-Rom does not define a speed value between two positions. Using a B-spline or NURBS curve allow this type of integration with the addition of other parameters such as pitch, roll, yaw of an aircraft.

[13]http://spatialreference.org/ref/epsg/4329

The following section presents a new analysis module that adds value for a specific case of air traffic.

6.1 Collision detection

To demonstrate the advantage of using 3D in the domain of air traffic management, we have developed a module to support collision detection in 3D space. This module identifies possible risks for air traffic. So, a sphere covers each aircraft and represent the position uncertainty.

Several choices are available for analysis with our application. These operations are done only for analysis between two objects. For the calculation, we use the timestamps in each trajectory. The time value t between two time t_0 and t_1 on the curve f is determined by the curvilinear abscissa of class C^1. The equation 3 defines the arc length L representing the distance traveled between two time values, and where $\|{}^{df}/_{dt}\|$ is the norm of the displacement speed vector.

$$L = \int_{t_0}^{t_1} \|\frac{df}{dt}\| dt, \quad \text{for } t \in [t_0, t_1] \tag{3}$$

We will assume that aircrafts have a constant speed K value between two GPS positions. Therefore, we have:

$$L = K \int_{t_0}^{t_1} dt = K(t_1 - t_0), \quad \text{for } t \in [t_0, t_1]$$

Figure 10 an example of analysis between two trajectories taking into is an account the time data. In a first step, the influence areas are initialised with a sphere. Figure 10 shows the result when a collision is detected. The gray sphere represents the time value of the collision. This method uses the analysis operation "contains" of ISO 19107 between two GM_Sphere.

7 CONCLUSIONS

This article has highlighted the capabilities of IS0 19107 standard to provide interoperable modelling and analysis. Our work was to model Earth's geoid with revolution surface (see section 4.2), based on the CRS ellipsoid WGS84, has yielded a better precision for modelling and analysis. Moreover, Globe3D entire platform is based on standardised modules as ISO 19111 for the referencing part in contrast to solutions proposed by Autodesk [14] or RhinoTerrain [15].

Section 6 has mounted the interest of using parametric curves to define trajectories. These curves offer a modelling closer to reality with the addition of degrees of freedom. However, it is possible to administer these degrees of freedom by using other curve like B-spline or NURBS. From this perspective, our future works are oriented with the ability to integrate semantic information on the aircraft as pitch, roll, yaw. This would produce a curve even closer to the real movement.

Further, we would like to study and add a new GM_-Primitive family in ISO 19107. For example, the extrusion surfaces are not currently referenced in the standard. These new surfaces would represent a tunnel or envelope of uncertainty around an aircraft parametric trajectory. We hope that this will be useful to realise intersections between volumes and trajectory (buildings, mountains, extrusion of country boarders). Our focus will also be on the formulation of curves in spherical or elliptical spaces rather than going through Euclidean projections.

From the application perspective, the next steps will also be integrating other standard interfaces enabling to extend its application to other domains such as management of marine data in real time with sensors addition (SWE) or permit the planning of air transport with geographic data connection like WFS [16], WCS or WMS [17].

8 ACKNOWLEDGEMENTS

This work is conducted while doing a Ph.D. thesis sponsored by the regional council of Provence-Alpes-Côte d'Azur and in collaboration with Geomatys [18] company and LSIS laboratory [19].

9 REFERENCES

[BZ03] R. Billen and S. Zlatanova. 3D spatial relationships model: a useful concept for 3D cadastre? *Computers, Environment and Urban Systems*, 27(4):411–425, July 2003.

[DA03] M. Davis and J. Aquino. JTS topology suite technical specifications. Technical Report 1.4, ViVid Solutions, 2003.

[Dan97] J. Danahy. *Automatic Extraction of Man-Made Objects from Aerial and Space Images (II)*, volume 2, chapter A set of visualisation data needs in urban environmental planning & design for photo- grammetric data, pages 357–365. Birkhäuser Verlag Basel, 1997.

[DB88] T. D. DeRose and B. A. Barsky. Geometric continuity, shape parameters, and geometric constructions for Catmull-Rom splines. *ACM Transactions on Graphics (TOG)*, 7(1):1–41, Jan 1988.

[FP79] I. D. Faux and M. J. Pratt. *Computational geometry for design and manufacture / I. D. Faux and M. J. Pratt*. Horwood ; Halsted Press, Chichester, Eng. : New York, 1979.

[FRD10a] A. François, R. Raffin, and M. Daniel. 3D ISO analysis and modeling for GIS - First steps in the norm implementation. *International Conference on Design and Decision Support Systems in Architecture and Urban Planning (DDSS)*, 33:1–16, july 2010.

[FRD10b] A. François, R. Raffin, and M. Daniel. Geometric data structures and analysis in GIS: ISO 19107 case study. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-4/W15(1682-1777):115–120, November 2010.

[ISO03] ISO TC 211. *ISO 19107 Geographic information – Spatial schema*. ISO, 2003.

[ISO04a] ISO JTC 1/SC 24. *ISO/IEC 14772-2 Information technology – Computer graphics and image processing – The Virtual Reality Modeling Language (VRML) – Part 2: External authoring interface (EAI)*. ISO, 2004.

[ISO04b] ISO TC 211. *ISO 19125 Geographic information – Simple feature access*. ISO, 2004.

[ISO07a] ISO TC 211. *ISO 19111 Geographic information – Spatial referencing by coordinates*. ISO, 2007.

[ISO07b] ISO TC 211. *ISO 19136 Geographic information – Geography Markup Language (GML)*. ISO, 2007.

[ISO10] ISO TC 211. *ISO 19130 Geographic information - Imagery sensor models for geopositioning*. ISO, 2010.

[Lu09] J. Lu. Circular element: Isogeometric elements of smooth boundary. *Computer methods in applied mechanics and engineering*, 198(30-32):2391–2402, 2009.

[PT97] L. Piegl and W. Tiller. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., 1997.

[Ree07] C. Reed. Kml 2.1 reference – an ogc best practice. Technical Report 0.0.9, Open Geospatial Consortium Inc., February 2007.

[Sed09] T. W. Sederberg. *Computer-Aided Geometric Design Course Notes*. Course Notes, Department of Computer Science Brigham Young University, September 2009.

[Shu10] A. Shukla. *Engineering Mathematics*. McGraw-Hill Education (India) Pvt Ltd, 2010.

[Ste05] C. Stein. *3D-Gis Marktanalyse*. PhD thesis, Fraunhofer IGD, 2005.

[TC 94] TC 184/SC 4. *ISO 10303-1 Industrial automation systems and integration – Product data representation and exchange*. ISO, 1994.

[Twi03] C. Twigg. Catmull-rom splines. *Computer*, 41(6):4–6, March 2003.

[Zla99] S. Zlatanova. VRML For 3D GIS. *Proceedings of the 15th SCCG*, 15:74–82, April 1999.

[Zla08] S. Zlatanova. Advances in 3D GIS. *DDD - Disegno Digitale e Design*, 1(718):24–29, 2008.

[14] usa.autodesk.com
[15] www.rhinoterrain.com
[16] Web Feature Service
[17] Web Mapping Service
[18] www.geomatys.com
[19] www.lsis.org

License Plate Detection using NMF with Sparseness constraints through still Images

Muhammad Jawad[1], M. Yasin[1], M. Saquib Sarfraz[1, 2]

[1]Computer Vision Research Group (COMVis), COMSATS Institute of IT, Lahore Pakistan

[2]Institute for Anthropomatics, Karlsruhe Institute of Technology, Karlsruhe, Germany

{mjawad, yasin.muhammad,ssarfraz}@ciitlahore.edu.pk

ABSTRACT

Real time license plate detection and recognition from CCTV videos is an active research problem. Most of the existing solutions are reasonably successful and efficiently fast but most of them are effective only in controlled environments where light intensity, illumination, orientation of plate do not vary much and image resolution is not too low. Two crucial image processing steps in an LPDR system are: (a) localization of license plates within an image and (b) recognition of license plate using an OCR system. The aim of this paper is to address the localization problem for low quality images. We use a novel, and robust framework to build a license plate detection system. Implemented system is intelligent enough to tackle varying environment conditions automatically with low hardware requirements and less complex algorithm. Experimental results on database collected under varying conditions demonstrate the robustness of the proposed approach.

Keywords

Automatic License plate Detection and Recognition (ALPDR), non-negative matrix factorization (NMF), Histogram of oriented gradient (HOG), Local Energy based Shaped Histogram (LESH)

1. INTRODUCTION

Automatic License plate detection and recognition (ALPDR) is one of the vital areas of research in computer vision and image processing over the past two decades. The demand of ALPDR System is increasing exponentially with the alarming increase in crime rates throughout the world. The objective of an ALPDR system is detection of license-plate-like regions in either still images or CCTV videos obtained through a road-side CCTV camera. The task becomes quite challenging when there is a wide variation in license plate shape, size and color. ALPDR can be put to use in various application areas including automatic highway toll collection, automatic parking control, traffic laws enforcement, border security, and crime prevention.

A number of techniques have been developed in the recent past for efficient detection of the license plates-like regions based on both still images and/or videos. A rank filter based approach is used by Martinsky [13] for the localization of license plate but the technique fails for skewed license plates. Most of the proposed works for ALPR systems [1, 6, 7,8,9] apply edge based feature extraction techniques for the localization of license plates. However, most of techniques are applicable only under highly controlled environment and fail of offer reasonable detection accuracy for non-uniformly illuminated license plates. A license plate localization and recognition technique is developed using attributes of the plates and neural network [14], this technique gives good results for localization but poor results on recognition. Mean shift algorithm [16] is another technique used for localization of license plate; rectangularity attribute, edge density, and aspect ratio are used as feature in Mahalanobis distance based classifier for detecting the correct license plate among several candidates. This method is robust as long as the color of the license plate differs from the color of the body of the vehicle. In another approach [15] Hough transform for line detection is proposed on the assumption that the shape of license plate is determined by lines, but the approach also fails to give good results on skewed license plates. Mathematical morphology based method is another approach proposed by [18]. Parallelogram and Histogram based Vehicle License Plate Detection model is proposed by [10]**.** Apart from all these techniques many researchers prefer a hybrid detection algorithm, where license plate location method based on characteristics of license plate shape, character connection and projection [19]. Not much work has been done on the localization of license plates for the regions (countries) where there is no standard for the size and aspect ratio of License plate like in south Asia.

In this paper a robust framework named non-negative matrix factorization (NMF) [4] is proposed for the localization of license plates. This discriminative framework is based on part-based approach which is useful in reconstruction of novel areas (license plate like regions). NMF has been successfully used in the past in applications like; pose primitive based human action recognition [3], Eye detection [2], and Aging Estimation [17]. NMF is used for the first time in license plate detection in our work. It is equally applicable for extracting candidate (license-plate-like) regions in both images and videos with relatively good degree of accuracy. The framework can also be used in conjunction with a number of feature extraction algorithms. There is no standard database available online for testing and comparison of a newly developed algorithm with the other existing methods. We created our own database of car images for the current experiment. The license plates vary significantly in size, shape and relative location on vehicle. License plate detection results are compared with some known classifiers like nearest neighbor and Ada-boosting and results are comparably better.

In subsequent sections key modules of the present work are described.

2. FEATURE EXTRACTION

Three different types of features extraction techniques are tested for robust visual object (License Plate) recognition on connected components found in an image. These feature extraction algorithms are very functional to locate interest points like corners, edges, and valleys in an image. Then feature vector of each detected candidate region is classified with the help of training data.

First feature descriptor used for this purpose is the Local Energy based Shaped Histogram (LESH) [11], which is based on local energy model of feature perception (Koveri 2000) described as:

$$E = \frac{\sum W(x)\lfloor A_n(x)(\cos(\varphi_n(x)-\bar{\varphi}(x)) - |\sin(\varphi_n(x)-\bar{\varphi}(x))|) - T\rfloor}{\sum_n A_n(x)+\varepsilon} \qquad (1)$$

Gabor filtering is used to obtain the local energy of an image which gives stable response in terms of high energy on edge corners. This algorithm generates 8 bins local histogram 'h' as:

$$h_{a,b} = \sum w_a \times E \times \delta(L-b) \qquad (2)$$

Where 'b' represents current bin, 'L' is the orientation label map, 'E' is the local energy extracted from equation 1, and 'w' represents Gaussian weighting function at required region 'a' (see equation 3).

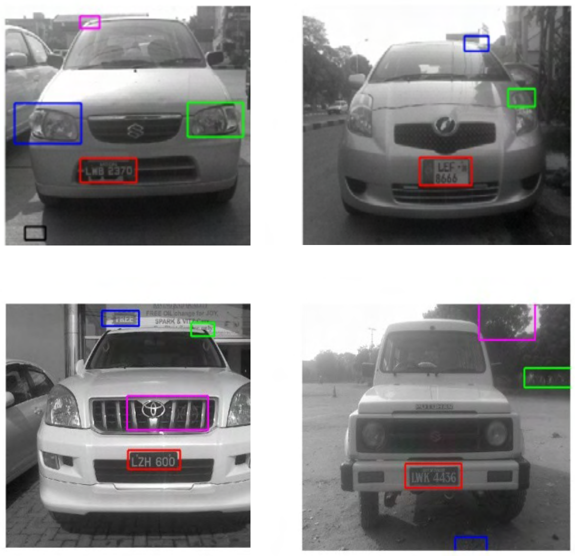


Figure 1. Examples demonstrating the connected components found in the images, where Red rectangular bounding box represents true localization of license plates.

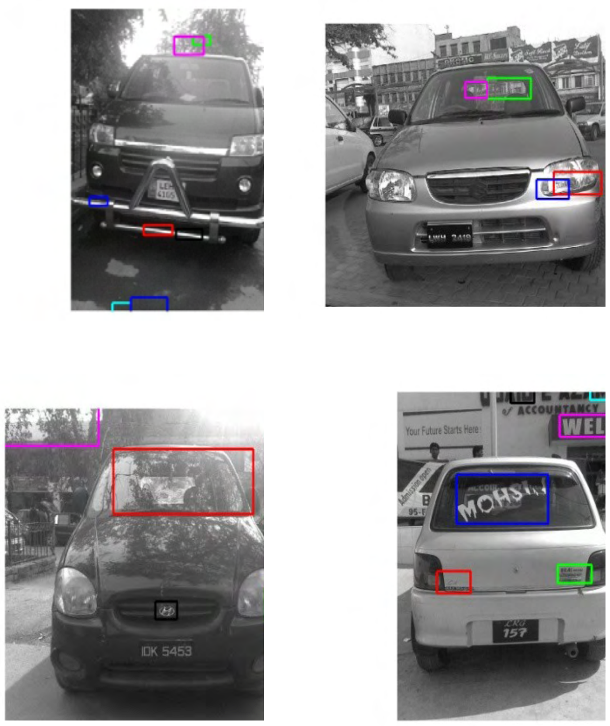


Figure 2. Examples demonstrating the false localization of candidates regions in the test images.

$$w_r = \frac{1}{\sqrt{2\lambda\sigma}} e^{[(X-a_{x0})^2+(y-a_{y0})^2]/\sigma^2} \quad (3)$$

In order to keep the spatial information of the image, we extracted 8 bin histograms by taking into consideration local energy along 8 filter orientations on 16 image partition then these local histograms are concatenate together, which makes 128-dimentional feature vector containing all spatial information of a connected component found during candidate selection.

Second descriptor used as an alternate of LESH is Histogram of oriented gradient (HOG) presented by [12], it is a well-known descriptor which use distribution vector of edge direction (gradient). It is useful for applications like License plate detection where edge orientation information helps for detecting an object shape and form in a well defined manner. In order to extract this feature descriptor, we divide the image (connected component) window into 4×4 spatial regions, then computing 9-bin histogram of edge orientations (directions) for each region. In the end these 16 histograms are concatenate together to form a 144-Dimensional feature vector which contains edge direction information for entire image window.

Third feature descriptor used is Scale Invariant feature Transform (SIFT) by [5]. These features are invariant to image scale and rotation and partially invariant to illumination change and 3D camera view point. This feature extraction approach is well localized in both frequency and spatial domain and highly distinctive. In this algorithm we have used 4×4 array of histograms with 8 orientation bins in each. Therefore 128-Dimension feature vector is extracted for each image window.

3. PROPOSED FRAMEWORK

In the present work we have developed a new technique for the localization of license plates in captured images from surveillance cameras. This Proposed framework can be observed clearly from Figure 3. In the subsequent sections the key modules involve in the present framework has been discussed.

3.1. Candidate Regions selection

Standard connected component analysis algorithm is applied on the pre-processed images for the selection of suitable candidate (license plate like) region (s). This algorithm scans whole image by moving along the rows from left to right and group all the pixels into component having same property (binary values) see Figure 1. This framework is equally applicable on both gray scale and binary images. Geometrical constraints are applied in order to minimize the number of selected components found in an image. These constraints are based on prior knowledge of license plates sizes used locally and globally like width-to-height ratio of license plates, so applied constraint will only select candidates having ratio b/w 1 to 4. Width and Height lengths of license plates are 150×300 pixels and 40×100 pixels depending upon the resolution of the images taken. Prior knowledge tells that eccentricity of license plates like regions should be less than 0.8. These connected components are then used for training and testing in NMF based proposed detection framework.

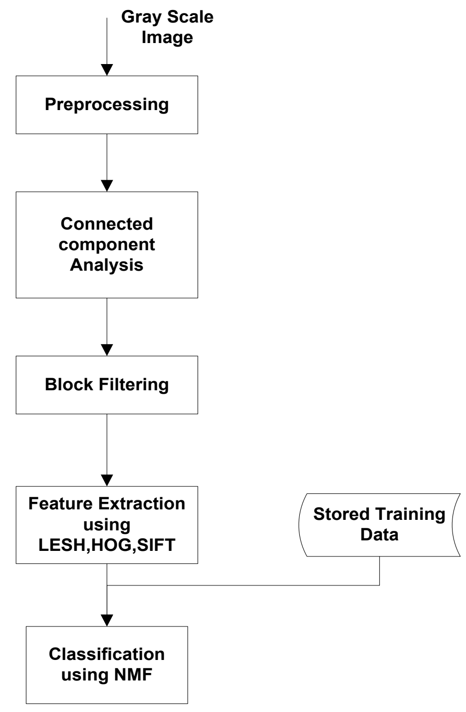


Figure 3. Block Diagram of the Proposed Framework

3.2. Classification using NMF

Non-Negative Matrix Factorization (NMF) [4] is used for classification of License plate like regions in test images. Main problem in most data-analysis problems is to find an appropriate representation of data. A functional representation usually makes latent structure in the data explicitly and often reduces the dimensionality of data. Non-negative matrix factorization is a suitable method for finding such a representation. Non-negativity constraints make NMF additive (allowing no subtraction) in contrast to other linear representations like PCA and ICA and this algorithm can only be applied on a non-negative data set [feature vector '*F*']. In our experiment three different types of feature descriptors are tested and all of them are non-negative. NMF is the part based approach which makes it quantitatively better. Another useful property of NMF is the sparse representation of data; benefit of such representation is that it encodes much of data using few 'Active' components which makes encoding easy to interpret. e.g. when we are trying to learn useful features from a dataset of images, it might make sense that matrices should be sparse. In NMF a linear data representation is simply a factorization of descriptor vector '*F*' written as $F = BW$, where '*B*' is known as the basis vector and '*W*' be a weight matrix and both are constrained to be non-negative, this factorization process is so powerful that we can approximately reconstruct descriptor vector '*F*' again by multiplying '*B*' and '*W*' this reconstruction power of NMF can be observed from Figure 5. NMF use standard multiplicative rule [4] for approximate factorization $F \approx BW$.

In the training phase positive (B_pos, W_pos) and negative (B_neg, W_neg) parameters vectors were extracted independent to each other, where parameter B_pos is the basis vector contains information of meaning-full parts of positive Samples (license plate). Initial data analysis stated that 80-100 positive and negative training samples each are sufficient for getting desired detection results. During testing W_test is computed from feature vector (F_test) of each connected component found in a test image by holding *'B'* fixed (extracted during training phase) using inverse NMF technique based on standard iterative algorithm as described in equation 4.

$$B = [B\_pos \quad B\_neg] \tag{4}$$

Key aspect of this algorithm is the use of additive combinations of weights that's why new coefficient vector W_test composes of separate coefficients of positive and negative weights like $W\_test = [W_{pos}^{test} \quad W_{neg}^{test}]$, so positive and negative appearances can be decoupled. License plate detection from 'n' number of connected components found in each image is based upon likelihood ratio, calculated by [3] as given in equation 5.

$$L = \frac{P(\mathrm{Im}/neg)}{P(\mathrm{Im}/pos)} \approx \frac{1 - \left|(F_{new} - F_{pos}^{new} / \left|F_{new}\right|)\right|}{1 - \left|(F_{new} - F_{neg}^{new} / \left|F_{new}\right|)\right|} \tag{5}$$

Here, $F_{pos}^{new} = B\_pos * W_{pos}^{test}$, $F_{neg}^{new} = B\_neg * W_{neg}^{test}$, and $F_{new} = F_{pos}^{new} + F_{neg}^{new}$ respectively.

From the repetition of experiment it is concluded that connected component which has likelihood ratio close to '1' should have more probability to be a license plate, so adjusted threshold of likelihood ratio for this experiment is [0.9-1.1].

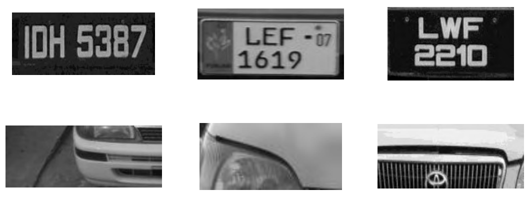


Figure4. Examples demonstrating training Samples,

First Row: Positive Sample images (License plates),

Second Row: Negative Sample Images (Non-license plates)

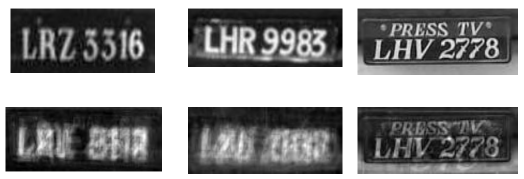


Figure 5. Examples demonstrating the power of reconstruction of NMF

First Row: Original License plates

Second Row: Reconstructed License plates

4. EXPERIMENT AND RESULTS

In order to show the efficiency of the algorithm dataset for the current experiment is collected from a major town in local area. Different surveillance cameras were installed at different locations like car parking, road crossing at different height places from the road surface. The complete image dataset comprises of more than 2000 frontal and a rear view of different types of vehicles under varies conditions of lighting, color, shadows, pollution levels and camera angles. Images were captured at a frame rate of 25fps and resolutions of the images are mostly low. We have used half of the captured images for training and half are used for testing in the current experiment which includes complete license plate like regions. Data set used for this experiment is also available online at (http://comvis.ciitlahore.edu.pk/).

4.1. Training dataset generation for NMF Framework

From a large number of images, we have in the dataset, rectangular license plate regions are extracted manually and stored separately. This way, a dataset of positive samples is created and all the images are marked as positive (+1). In the same way, random sub-images of the size of license plate images are cropped from non-plate regions from the dataset and are labeled as negative (-1). The feature vector (LESH, HOG or SIFT) is computed for every image in both the classes and concatenated to form feature matrix 'F'. This Matrix is factorized into basis matrix 'B' and weight matrix 'W' using NMF. These training matrices are used for real time localization of license plates in testing phase.

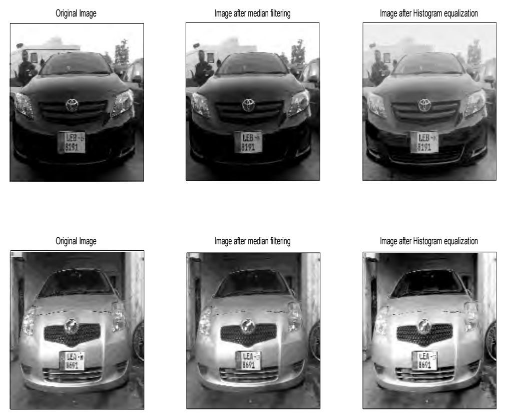

Original Image
Image after median filtering
Image after Histogram equalization
Original Image
Image after median filtering
Image after Histogram equalization

Figure 6. Representing the first phase of the algorithm, where standard pre-processing techniques are applied to increase image quality. First Column contains Original input images, Second Column contains images after removing salt & pepper noise, Third Column is representing image after contrast enhancement.

4.2. License Plates Localization

Test images are taken under varying environment conditions through multiple cameras over different day timings, so they include embedded salt and pepper noise and huge contrast variations.

Firstly, standard pre-processing techniques are applied sequentially to alleviate or attenuate some of the artifacts mentioned earlier. Median filtering is applied on gray images to reduce "Salt and pepper" noise, it replaces gray value of a pixel by the median of the gray values of its neighbors for this purpose we have used default 3×3 neighborhood of a pixel. Contrast of images is enhanced by using histogram equalization technique. Results of pre-processing are shown in Figure 6. Although the exact location of license plate is unknown in this phase, this process locally balances the illumination in different parts of an image.

Training set for the experiment which comprises of positive (license plates) and negative (non-license plates) image samples (100 each) see Figure 4. We have compared the performance of the proposed NMF based classifier against K-nearest neighbor (KNN) where K = 3, and Ada-boost (decision Stump as weak classifer) on the basis of Precision/Recall/F-Score and Accuracies on both the training and test datasets (See Table 1). Although the processing time of NMF is more than the other two classifiers but there is significant improvement in terms of Precision, Accuracy and F1-Score.

NO	**Descriptor**		**Precision rates/ classifier**		
			NMF	**KNN**	**ADA-boosting**
1	**LESH**	Precision	72%	57.1%	48.95%
2		Recall	87%	92.1%	49.47%
3		Accuracy	70.2%	61.5%	48.94%
4		F1-Score	0.809	0.705	0.4920
5		Processing Time/ image	0.8 sec	0.7 sec	0.55 sec
1	**HOG**	Precision	68.3%	52.5%	54.8%
2		Recall	76.3%	80.8%	56.3%
3		Accuracy	70.5%	57.3%	57.6%
4		F1-Score	0.721	0.681	0.555
5		Processing Time/ image	0.7 sec	0.6 sec	0.42 sec
1	**SIFT**	Precision	65.2%	50.2%	52.6%
2		Recall	80.4%	82.4%	62.5%
3		Accuracy	60.5%	56.8%	58.3%
4		F1-Score	0.718	0.618	0.702
5		Processing Time/ image	0.6 sec	0.5 sec	0.3 sec

Table 1. Comparison of Precision rates of three different Classifiers for testing data

5. CONCLUSION AND DISCUSSION

NMF based car license plate detection algorithm is proposed and discussed in this paper. NMF is robust framework which is used for the first time as a classifier in license plate detection application precisely with high degree of accuracy. Our experimental results show that NMF can be a useful classifier used in ALPDR systems. Robustness of this method is verified by comparison with other known classifiers on a big mixed dataset taken by us under various conditions because there is no standard dataset available online for testing of such algorithms. Another advantage of this algorithm is the high precisions of detection boxes, so character recognition algorithm can be applied easily which is the next step of our work. There are some vital appealing aspects for future research like candidate selection procedure can be improved to increase detection rates also the processing time of present research work can be reduced by implementation in C++ and open CV for real applications.

6. REFERENCES

[1] C. Anagnostopoulos. I. Anagnostopoulos, V. Loumos and E Kayafas, “A license plate recognition algorithm for intelligent transportation system application,” IEEE Transactions on Intelligent tansportation systems, Vol. 7, No. 3, pp. 377-392, September 2006.

[2] C.W. Park, K.T. park and Y.S. Moon, “Eye detection using eye filter and minimization of NMF-based reconstruction error in facial image”, IEEE Electronics letter, Vol. 46, No. 2, Jan 2010.

[3] Christian Thurau, Vaclav Hlavac, “Pose primitive based action recognition in videos or still images”, Conference on computer vision and pattern recognition, June 22-24, 2008.

[4] D. D. Lee and H. S. Seung., “Algorithm for non-negative matrix factorization.”, In NIPS’01, 2001.

[5] David G. Lowe, “Distinctive image features from scale-invariant key points”, International journal of computer vision, Jan 2004.

[6] Erik Bergenudd, “Low-Cost Real-Time License Plate Recognision for a Vehicle PC”*, Master’s Degree Project*, KTH Electrical Engineering, Sweden, December 2006.

[7] H. Kawasnicka and B. Wawrzyniak, “License Plate Localization and Recognition in Camera Pictures”, AIMETH 2002, Poland, November 2002.

[8] H. Mahini, S. Kasaei, F. Dorri and F. Dorri, “An Efficient Features-Based License Plate Localization Method”, Proceedings of 18th International Conference on Pattern Recognition, 2006.

[9] J. R. Parker and P. Federl, “An Approach to License Plate Recognition”, Computer Science Technical Report (1996-591-1. I), 1996.

[10] Kaushik Deb, Hyun-Uk Chae, Kang-Hyun Jo,” Parallelogram and Histogram based Vehicle License Plate Detection”, International Conference on Smart Manufacturing Application, 2008

[11] M. Saquib Sarfraz, Olaf Hellwich, “Head pose estimation in face recognition across pose scenarios”, International conference on computer vision theory and applications, 2008

[12] Navneet Dalal and Bill Triggs, “Histograms of Oriented Gradients for Human Detection”, Conference on Computer Vision and Pattern Recognition, 2005

[13] O. Martinsky, “Algorithmic and Mathematical Principles of Automatic Number Plate Recognition System”, B. Sc. Thesis, BRNO University of Technology, 2007.

[14] Satadal. Saha, Basu. Subhadip, Nasipuri. Mita, Kr. Basu. Dipak,” Localization of License Plates from Surveillance Camera Images: A Color Feature Based ANN Approach”, International Journal of Computer Applications, volume 1, 2010, no.23.

[15] V. Kamat and S. Ganesan, “An efficient implementation of the Hough transform for detecting vehicle license plates using DSP’S,” in proc. first IEEE Real-time Technol. and Applications Symp.(RTAS’95), pp. 58-59, May 15-17, 1995.

[16] W. Jia, H. Zhang, X. He and M. Piccardi, “Mean Shift for Accurate License Plate Localization”, Proceedings of 8th International IEEE Conference on Intelligent Transportation Systems, Vienna, Austria, Sept. 2005.

[17] Yong-Qing. Ye, Ji-Xiang. Du, and Chuan-Min. Zhai, “Aging Simulation of Human Faces Based on NMF with Sparseness Constraints”, Springer-Verlag Berlin Heidelberg, 2010.

[18] Z. Qi and S. Zhong-ke, “A real-time algorithm for license plates extraction based on mathematical morphology,” Journal of Image and Graphics, Vol. 8, No. 3, pp. 281-285, 2003.

[19] Z. Xu and H. Zhu, “An Efficient Method of Locating Vehicle License Plate,” Third Int. Conf. on Natural Computation (ICNC’07), 2007.

Video-based Human Activity Analysis: An Operator-based Approach

Xiao Bian
North Carolina State University
Department of Electrical and Computer Engineering
27606, Raleigh, NC, USA
xbian@ncsu.edu

Hamid Krim
North Carolina State University
Department of Electrical and Computer Engineering
27606, Raleigh, NC, USA
ahk@ncsu.edu

ABSTRACT

Human activity in video sequences may be viewed as a sampled trajectory on a low dimensional manifold embedded in a high dimensional ambient space. Due to the unknown underlying manifold structure of the image frames, we propose a novel framework to define a neighborhood for high dimensional data which, when acted upon by a mapping operator, results in a subset in an a priorily well defined range space. We exploit the so-called correlation filtering with a specifically selected output response to effectively approximate the data manifold by way of encoding local neighborhoods on it. This helps us propose an unsupervised learning algorithm of human activity, and demonstrate its performance in classifying and clustering of different activities taking place in observed video sequences.

Keywords

Manifold learning, Human activity, Correlation filter

1 INTRODUCTION

Video-based human activity analysis plays an important role in video content analysis such as video surveillance and video indexing. Its inherent complexity is due to the high dimensionality and nonlinearity of the associated feature space. Using the low dimensionality which is intrinsic to the human activity dynamics, numerous manifold learning techniques have been proposed [AWAR11] [BR07] [SKN12]. By reducing the dimension of the feature space, one can reduce the impact of 'the curse of dimensionality', and thereby potentially improve the human activity classification performance [BNS06]. The raw data on its own, is, however, insufficient to reflect the necessary information for characterizing the underlying manifold structure (or associated tangent space) of the frame sequences in various human activities. This problem was overcome by shape-based methods [SKN12] as well as in [GBS$^+$07] by exploiting the well defined shape manifold of the silhouette contours in images of video sequences. These unfortunately, exhibit limitations when the targets (human images in the frames) undergo a topological change, or when there is more than one target in a scenario to be analyzed.

The key idea underlying manifold learning is the assumed ability to faithfully encode its essential local information . This local information is, in turn, determined by the k-nearest neighbor technique. It is thus critical to find a robust way to identify the neighborhood for each data sample (i.e. frame). In this paper, we choose to define a neighborhood in a high dimensional space by way of a sequence of correlation operators defined on the manifold. The manifold structure is thus now encoded in the operator sequences which act on the corresponding neighborhoods. We also exploit the correlation filters [BB10] to develop an unsupervised learning algorithm for capturing human activity characteristics under the proposed framework, and to thereby demonstrate its learning capacity by interpolation and activity transition detection. The clustering and classification potentials of the algorithm are in addition illustrated, and their performance demonstrated.

The remainder of this paper is organized as follows, in Section 2, we describe the mathematical formulation of the local information encoding by way of an associated operator sequence on the underlying data manifold of interest. In Section 3, we introduce the detailed algorithms used in analyzing the human activity video sequences of interest. In

Section 4, we substantiate our development by running clustering and classification experiments on real data, and demonstrate the performance of our proposed method. In Section 5, we conclude with some remarks and with a brief discussion of future research directions.

2 CLUSTERING OPERATORS ON MANIFOLD

It is a clear fact in human activity analysis or in general video sequence analysis, that we neither have the explicit form of the underlying manifold of the images/frames, nor do we have the ability to approximate the tangent space of this manifold. Many learning techniques have explored underlying structures of manifolds by tangent space estimation, which in turn is approximated by neighborhood points determined by k-nearest neighbors [Row00] [Don03] [BN01]. However, distance measures become very sensitive with the increase of dimensionality, and the proportional difference between a farthest point and a nearest point vanishes, making the k-nearest neighbor approach unreliable [HK00] [HK10]. In addition, sample points (in our case our video frames) are not necessarily uniformly distributed in ambient space making the tangent space estimation even more challenging, and learning the manifold more elusive.

When considering human activity, we may, however, assume a low dimensional structure for a given video sequence, or as a time sequence of sample points in high dimension, which may in turn be viewed as a trajectory curve on a lower dimensional manifold embedded in high dimensional space.

As discussed in the next sections, we take advantage of mapping operators to more precisely define sequential neighbors in a high dimensional space.

2.1 Bounded operators on a manifold

Building on [Yao98] [RV07], we proceed to determine "neighboring frames" by way of mapping operators applied to data to achieve a predefined desired output.

To be precise, the following definition of neighbors is in order,

Definition 1 *Consider a given continuous mapping $\mathscr{T} : U \mapsto V \subset R^n$, U is a subset of a data manifold M which is embedded in R^n. Given $\sigma > 0$.* ***A σ-neighborhood*** *of x_0 is a subset $N \subset U$, $\forall x \in N$ satisfying $\|\mathscr{T}x_0 - \mathscr{T}x\| \leq \sigma$.*

When searching for neighbors of $x_1, \|\mathscr{T}x_1 - \mathscr{T}x_2\|$ may also be interpreted as $h : U \mapsto \mathbb{R}^+, h(x) = \|\mathscr{T}x_1 - \mathscr{T}x\|, x$ is a neighbor of x_1 if and only if $h(x) \leq \sigma$. From the theory of bounded operators [Kre78], we may alternatively provide a more general scope by the following,

Definition 2 *Consider $h : U \mapsto V$ a continuous mapping on $\mathscr{U}$, x is a* ***W-neighbor*** *of y under h, if $h(x) \in W, h(y) \in W, W \subset V$, $S = h^{-1}(W) \subset U$ is the* ***W-neighborhood*** *under h.*

The complexity and the nonlinearity of a human activity require that an operator sequence $\tilde{h} = \{h_1, \ldots, h_q\}$ be applied to cover all the degrees of freedom intrinsic to high dimensional video sequences. Each operator h_i in the sequence $\tilde{h}$ defines a neighborhood S_i, which can be seen as an approximation of a tangent space of the data manifold. The set of $S = \{S_i\}$ covers the whole high dimensional space of video sequences of a given human activity. Practically, this is also a way to overcome the difficulties of k-nearest neighbor estimation, and to encode local information of a manifold simultaneously, as noted earlier in Section 2. This is a more general and flexible depiction/representation of a local open set on a manifold than that of a local tangent space often used in manifold learning algorithms. To sum up the idea above, we may succinctly state the following,

Definition 3 *Consider a curve $\mathscr{X} : [0,1] \mapsto U \subset R^n$, given a set of operators $\tilde{h} = \{h_1, \ldots, h_q\}, h_i : U \mapsto V$, $\mathscr{X}$ is said to be covered by a W-neighborhood under $\tilde{h}$, if $\forall t \in [0,1], \exists i \in \{1, \ldots, q\}, h_i(\mathscr{X}(t)) \in W, W \subset V$.*

2.2 Representative Operators of a High Dimensional Image Space

To exploit the framework in Section 2.1 in addressing a video sequence modeling, we need to construct an operator set associated to video sequences in the class of human activities of interest. An important property of any resulting operator, is that it must be least sensitive to noise commonly encountered in measured images. Instead of vectorizing each image in R^n, and encoding the local information by selecting k-nearest neighbors in the metric space R^n, as is commonly practiced in manifold learning techniques [Row00] [Law05], we choose to radically depart from this idea. This is primarily due to the high sensitivity of Euclidean distance in R^n to common noise terms in image processing (i.e. White noise), and to the limitation brought on by "the curse of dimensionality" [HK00] [HK10]. Inspired by the successful application of filters in image processing [BB10], we proceed to choose our fore-described mapping operators from the class of 2-dimensional convolution kernels: $h_i(x) = h_i * x$. In contrast to the distance-based low dimensional subspace representation of vectorized frames, the inherent relations among

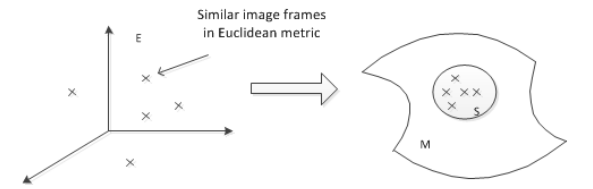


Figure 1: Similar frames may not close to each other in Euclidean metrics. The Trained Operators can map similar frames close to each other in the range space

video frames will be preserved in the neighborhood under the mappings, and the noise largely filtered out.

To more clearly define the data neighborhood, we proceed to define the covering of $X(t)$ (frame/video sequence) by an operator sequence,

Definition 4 *$h : U \mapsto V$ is a convolution operator on $\mathscr{U} \subset R^n$, then x is a neighbor of y if $h * f \in W, h * y \in W, W \subset U$.*[1]

Definition 5 *$\mathscr{X} : [0,1] \mapsto U \subset R^n$, is covered by $h = \{h_1, \ldots, h_q\}, h_i : U \mapsto V$ if $\forall t \in [0,1], \exists i \in \{1, \ldots, q\}, h_i * \mathscr{X}(t) \in W, W \subset V$.*

3 CONVOLUTIONAL OPERATOR SEQUENCE CONSTRUCTION

Convolutional kernels have been successfully used in video analysis for correlation filter-based tracking [BB10]. Trained correlation filters can robustly track targets in a video sequence. It is a special case of the filters in Definition 5 since one filter is sufficient to learn from the data sequence, and to track the target with small perturbation. In human activity analysis, a target may dramatically change because of articulation deformation. This is hence tantamount to saying that a set of operators will be required to capture all the potential information embedded in a video sequence. Building on MOSSE filters [BB10], the training of operators is carried out on the basis of a sample point set of a high dimensional curve $\mathscr{X}$ per Definition 5. For clarity as well as computational efficiency, we discuss all implementational issues in the Fourier domain. Let $F = \mathscr{F}(f)$ denote the 2D Fourier transform of a given image frame, and $H = \mathscr{F}(h)$ that of the h. In that light, and given a data set F_i, the optimal filter H^* is obtained by

$$\min_{H^*} \sum_i |F_i \bigodot H^* - G_i|, \tag{1}$$

where $G_i = \mathscr{F}(g_i)$ denotes the Fourier transform of the ideally desired output g_i, a spatial ($2D$) gaussian signal, and $\bigodot$ denotes a Schur product (i.e. an element-wise multiplication).

As shown in [BB10], the optimal solution of Eq.(1) is

$$H^* = \frac{\sum_i G_i \bigodot F_i}{\sum_i F_i \bigodot F_i^*}. \tag{2}$$

3.1 Neighborhoods Information Encoding

As noted earlier, the operator sequence with their associated neighborhood encode the information of the data manifold, and hence implicity describe a corresponding manifold of operators which captures and reflects the information of the initial data (video sequence). It is hence important that the defined neighborhood be preserved following the mapping (i.e. close points in the initial manifold space should remain close in the range space upon mapping) and be robust to noise. To this end, we also adopt a Peak-Sidelobe Ratio (PSR) as the optimization criterion of choice [BB10],

$$PSR = \frac{g_{max} - \mu_s}{\sigma_s}, \tag{3}$$

where g_{max} is the maximum value of the response; μ_s and σ_s respectively are the peak value and variance of a sidelobe.

Given $g_i = f_i * h$, f_i is the input data and h is the operator, then PSR is a function defined on the output image domain, $PSR(g_i) \in R^+$.

More specifically, neighbors of a given frame under a correlation filter operator is defined as,

Definition 6 *For $x, y \in \mathscr{U}$, h is a operator on $\mathscr{U} \subset R^n$, y is called a neighbor of x if $h * x \in W$ and $h * y \in W, W \subset U$. Furthermore, $h * x \in W \Leftrightarrow PSR(h * x) > \eta > 0$*

3.2 Learning an Operator Kernel Sequence from Data

As discussed above, we propose a specific algorithm to automatically explore the structure of the data manifold by way of a sequence of kernel convolutional operators whose manifold is more easily characterized. The cardinality of the set of operators being unknown a priori, we first randomly pick a frame f_i from the available data set. We subsequently select all points of the neighborhood of an f_i-trained operator to further train the operator H_j. Excluding all data points from the neighborhood under H_j, we pick the farthest point in the data set from S, to iteratively determine the next operator and until all data in the training set is covered by the operator sequence $H = \{H_i\}$.

The algorithm to construct an operator sequence is described below.

[1] x and y are samples/frames of a sequence.

- Given image sequences $\{f_i\}$ as a training set T
- Randomly pick one frame f from the training set T
- While T is not an empty set
  1. Train operator H from frame f. Find all frames f_{H_i} which belong to S_H(The neighborhood under H)
  2. Train operator $\hat{H}$ from S_H. Find all frames $f_{\hat{H}_i}$ which belong to $S_{\hat{H}}$(The neighborhood under $\hat{H}$)
  3. Define T as the complement of T with respect to $S_{\hat{H}}$, $T = T - S_{\hat{H}}$
  4. Define f_i to be the 'farthest' point to $S_{\hat{H}}$ in T: $f_i = \arg\min_{f_j \in T} PSR(h(f_j))$

4 VALIDATION AND EXPERIMENTS

We next carry out a series of experiments to demonstrate the performance of our algorithms for human activity analysis. We use the database in [GBS+07], which includes 9 different people performing 9 different activities individually, such as jumping, running and walking, etc. The relatively low resolution video clips with a naive foreground extraction, provide a good approximation to the noisy and imperfect environment of video surveillance. And since the textural information for each person does not impact the activity itself, we exclude the textures, and only use distance maps for each frame.

4.1 Interpolation and Segmentation

With an operator set $h = \{h_i\}$ and video frames $f = \{f_j\}$, each f_j has a index i_j from its corresponding operator h_{i_j}. The sequence of indices demonstrates the stages of a video sequence. Fig. 2 gives an example of the segmentation of a video sequence.

With the information of the operator sequence, we can interpolate between two frames, as shown in Fig. 3 and Fig. 4. Note that if data on the manifold as defined by our operator sequence is used, the interpolation should then successfully recover the true activity frame.

4.2 Activity change detection

Activity change will influence the response of operator sequences. Since the operator set is specifically designed to cover all variations in each activity, the reduced response always means the variation of activity, as shown in Fig. 5

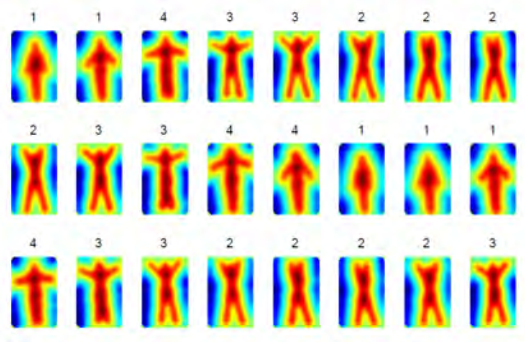


Figure 2: Segmentation of human activity based on neighborhood under each operator. The number on each frame represents the index of the associated operator

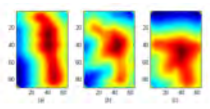


Figure 3: Human activity sequences (bending) interpolation. (b) The interpolated gesture between left and right frames

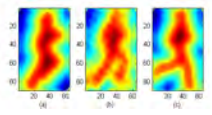


Figure 4: Human activity sequences (running) interpolation. (b) The interpolated gesture between left and right frames

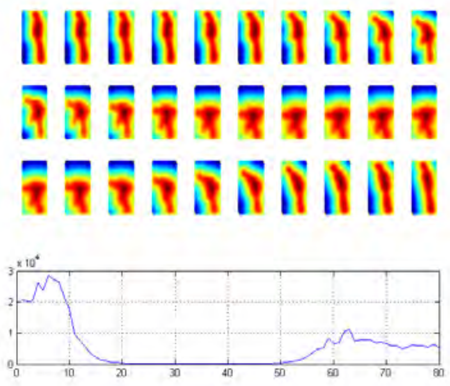


Figure 5: The response of the operator set is severely decreased with activity transition. x axis represents frame number and y axis represents PSR value

4.3 Activity Clustering

Similarity between two sequences $f^i = \{f_k^i\}$ and $f^j = \{f_k^j\}$ is defined as

$$A_{ij} = (\frac{N_{ij}}{N_{ii}} + \frac{N_{ij}}{N_{jj}})/2 \quad (4)$$

N_{ij} = Cardinality of $S_j \bigcap f^i$. S_j is the neighborhood of f^j under their own operator set. Notice that here no further registration or alignment is needed for this metric.

In this experiment we applied a regular spectral clustering algorithm [NJW01] on the similarity matrix(Fig. 6)of all 81 video sequences. With 7 out of

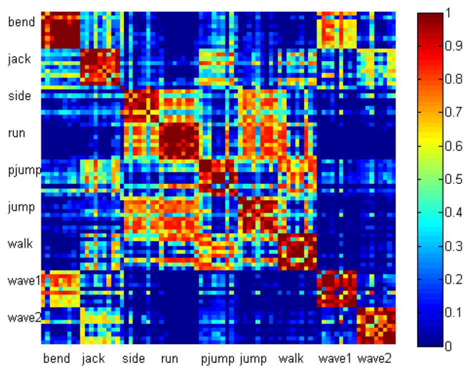


Figure 6: Similarity matrix of 9 classes of human activities, 9 realizations for each activity. 1 means the given two sequences are similar to each other and 0 vice versa

81 misclassified, 91.36% sequences are correctly clustered. The success of clustering here demonstrates that the similarity measure reflects the intrinsic closeness of different human activity sequences.

4.4 Activity Classification

We separate the database into a training set and a test set by randomly selecting 4 sequences from each activity and let the remaining 5 sequences of each activity as a test set. For an input sequence, each operator set H_i will convolve the signal and we can have a set of sequences of PSR value. We calculate the maximum PSR at each frame for each operator set, and use them as similarity features between input and the corresponding class. The input sequence is then assigned to the class of operator with maximum median, such as Id of Input = Id of $\max median(PSR(g))$

In spite of selecting relatively a small training set, the rank-1 recognition rate is 90.06%, with 31 misclassified in 315 tests.

5 CONCLUSION

In this paper, we present a novel framework of manifold learning by using operators on a manifold and propose an unsupervised learning algorithm to represent human activity sequences with a small number of operators. By using the set of operators for each human activity, we demonstrate the high performance for clustering and classification. Combined with successfully interpolating frames among sequences, the experiments show that the feature extracting by an operator set construction is fast, accurate and robust. Furthermore, from the experimental results, a high dimensional neighborhood in our framework is more robust compare to Euclidean distance, and shows potential to overcome the curse of dimensionality.

6 REFERENCES

[AWAR11] M. Abdelkader, A. Wael, S. Anuj, and C. Rama. Silhouette-based gesture and action recognition via modeling trajectories on riemannian shape manifolds. *Computer Vision and Image Understanding*, 115(3):439–455, March 2011.

[BB10] D. Bolme and Jr Beveridge. Visual object tracking using adaptive correlation filters. *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[BN01] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001.

[BNS06] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold Regularization : A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[BR07] J. Blackburn and E. Ribeiro. Human motion recognition using Isomap and dynamic time warping. In *Proceedings of the 2nd conference on Human motion: understanding, modeling, capture and animation*, pages 285–298. Springer-Verlag, 2007.

[Don03] D. Donoho. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Academy of Sciences of the United*, (650):1–15, 2003.

[GBS+07] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE transactions on pattern analysis and machine intelligence*, 29(12):2247–53, December 2007.

[HK00] A. Hinneburg and D. Keim. What is the nearest neighbor in high dimensional spaces. In *Proc. VLDB*, 2000.

[HK10] M. Houle and H. Kriegel. Can shared-neighbor distances defeat the curse of dimensionality. In *22nd International conference on scientific and statistical database manangement*, 2010.

[Kre78] E. Kreyszig. *Introductory Functional Analysis with Application*. Wiley, 1978.

[Law05] C. Lawrence. Algorithms for manifold learning. *University of California, San Diego, Tech. Rep. CS2008*, 2005.

[NJW01] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information*

Processing Systems, pages 849–856. MIT Press, 2001.

[Row00] S. T. Roweis. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, December 2000.

[RV07] J. Rowe and M. Vose. Neighborhood graphs and symmetric genetic operators. *Proceeding FOGA'07 Proceedings of the 9th international conference on Foundations of genetic algorithms*, pages 110–122, 2007.

[SKN12] Y. Sheng, H. Krim, and L. Norris. Human Activity Modeling as Brownian Motion on Shape Manifold. *Scale Space and Variational Methods in Computer Vision*, pages 628–639, 2012.

[Yao98] Y. Yao. Relational interpretations of neighborhood operators and rough set approximation operators. *Information sciences*, 111(1):239–259, 1998.

Digital Restoration of Old Paintings

Nidhi Arora, Ankush Kumar and Prem Kalra
Indian Institute of Technology Delhi
New Delhi, India
Email: $\{nidhi, pkalra\}$@cse.iitd.ac.in, ankush.coe@gmail.com

ABSTRACT

Paintings are made up of materials that can suffer damage with the passage of time. To protect them over a long period, they are coated with a protecting covering of varnish layer. The varnish layer over paintings is affected by atmospheric conditions, fluctuations in temperatures, humidity and sunlight. Over a period of time, the transparency of the varnish becomes clouded and discolored, often resulting in a picture being viewed as if through an amber or even brown or black filter. We analyze the effect of varnish layer on the visual appearance of old paintings and provide the correlation of degradation with the quantitative measures such as entropy and standard deviation of the points cluster of the image in the color space. We further develop a method of color restoration by appropriately transforming the color space. We provide both an interactive method and an automated example-based method. In addition to the color degradation, cracks which appear apparently on the surface, are also caused by the aging process. As a result of cracks and color degradation, paintings may lose their aesthetic and historical values. In this paper, we also integrate crack detection with color restoration.

Keywords
color restoration, crack detection, crack filling, inpainting, crack restoration.

1 INTRODUCTION

Digital image processing techniques are widely used in all scientific fields. Image processing techniques are recently being applied to analyze, preserve and restore artwork. Art work restoration is a very demanding field which requires considerable expertise. As the years roll by, a number of defects appear in paintings like the development of cracks, scratches, discoloration of the varnish layer, accumulation of dust, dirt, smoke on the surface of the painting, loss of paint, etc. Hence, the restoration of such old degraded paintings include stabilization, surface cleaning, the removal of discolored varnish, the repair of tears and punctures, filling areas of paint loss, and expert retouching. Digital processing on the old paintings is analogous to this manual, chemical cleaning of old paintings. Hence, art conservators and restorators can be helped using such digital image processing techniques.

In this paper, we present an approach for restoration of digital paintings inspired by the manual process for cleaning and color restoration of old paintings which may have undergone a degradation in their visual appearance due to the accumulation of dirt, smoke and oxidation of the varnish layer (in oil paintings) etc.

The paper is organized as follows. In section 2, some of the vital previous research carried out in this field is discussed along with our contribution. Section 3 analyses the effect of the varnish layer on the color space of old paintings. Section 4 describes the similarity metric chosen to assess the quality of results and the procedure adopted to quantitatively compare the results. Section 5 describes the role and purpose of user intervention in the process of digital color restoration. In section 6, we present another approach for digital color restoration of old paintings based on example clean paintings. Section 7 presents some of the results followed by section 8 which concludes our work.

2 RELATED WORK

Pappas & Pitas [PP00] report a pioneering work on digital color restoration of old paintings. They mention five approaches for cleaning an image. In each of these, they try to derive a color transformation f such that $s = f(x)$ using cleaned samples (denoted by s) and old samples (denoted by x) of the painting. Linear Approximation and White Point transformation approaches yield promising results as compared to other three approaches. Palomero & Soriano [PS11] propose a method which uses a neural network to learn the transformation from dirty to clean segments of a painting image. Gasparini and Schettini [GS03] suggest an algorithm which is structured in two main parts: a cast detector and a cast remover. An acquired image may have a cast which is an undesirable shift in the entire color range. Reinhard et al. [RA01] use a simple statistical analysis to impose one image's color characteristics on another. They achieve color correction by choosing an appropriate source image and apply its characteristic to another image.

Giakoumis & Pitas [GP98] propose a method for digital restoration of cracks in old paintings. The technique consists of the following steps:

- Detection of cracks using the top hat transformation.
- Separation of brush strokes which have been misidentified as cracks.
- Crack filling procedure using seed growing approach.

Separation of brush strokes is achieved by classification using Minimal Radial Basis Function (MRBF) neural network. Therefore, a huge number of samples of cracks and brush strokes are required for the learning stage. The Minimal Radial Basis Function (MRBF) neural network [ZV09], after its training phase, becomes capable of deciding whether the pixel area in the original image belongs to a brush stroke or a crack. Crack filling is performed using the Bertalmio et al.'s inpainting [BS00] and Exemplar based inpainting technique [CP04].

Contribution: In this paper, we analyze the role of oxidized varnish layers on the color space of old paintings and on their appearance. We also propose an approach for digital color restoration without performing any chemical treatment on the surface of the painting. We provide an interactive method to modify the color space, represented as point clusters of RGB. Further, we introduce an example-based technique to automatically transform the RGB space of the given degraded image in accordance to an example image. We provide a quantitative measure of KL-divergence (relative entropy) to support the results.

3 ANALYSIS OF VARNISH LAYER

The varnish layer protects the painting from abrasion and pollution in the atmosphere. It also brings out the colors to the brilliance they had when applied while it is still transparent, but over a period of time, due to oxidation and deposition of dirt and smoke, the varnish layer becomes opaque, resulting in a picture being viewed as if through an amber or even brown or black filter.

In Figure 1 we can observe that, overlapping area has dull and amber look as compared to other region of the painting. On experimenting with a number of paintings, it has been observed that due to the dirty varnish layer, the standard deviation and entropy of the image decreases. Tables 1, 2 & 3 exhibit the comparison of the results of standard deviation and entropy of a number of chemically cleaned and old paintings.

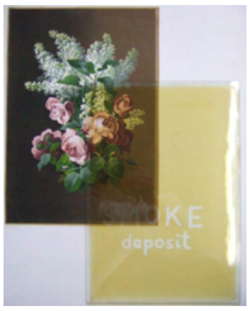


Figure 1: Effect of dirty varnish layer

They represent the point cluster of old and chemically cleaned paintings in RGB color space. It is observed that due to the effect of oxidized varnish layer the mean color ($\bar{R}$, $\bar{G}$ and $\bar{B}$) changes and subsequently results in shifting the origin of the point cluster of the paintings.

It is also observed that the point cluster volume of the old painting also decreased (due to decrease in the standard deviation). Further, due to decreased point cluster volume, only limited range of colors is available to represent the image giving it an amber coloration and hiding the vibrant colors beneath.

Poor contrast images have a dull look due to the low range of gray levels available for the image and we can improve their appearance by increasing the range of gray levels (histogram stretching). We observe that transformations such as scaling, translation and rotation must have been applied on the point cluster of old painting to obtain the point cluster of the chemically cleaned painting. The point clusters could be obtained in either RGB or $L\alpha\beta$ color space. Thus, we utilize this concept of applying the various transformations on the given point cluster of the old painting and observe its corresponding effect on the painting. Now, in the next section, we show user intervention in specifying the various parameters for scaling and translation and obtain a satisfactorily clean image based on the visual perception.

4 SIMILARITY METRIC

Many image quality assessment(QA) algorithms exist in the literature whose goal is to automatically assess the quality of images in a perceptually consistent manner. These image QA algorithms [SB05, SB06, ZJ06, WS05, WL11, MS01] interpret image quality as fidelity or similarity with a 'reference' or a 'perfect' image. Wang and Simoncelli [WS05] predict the visual quality of distorted images using only the partial information about the reference images. They propose QA

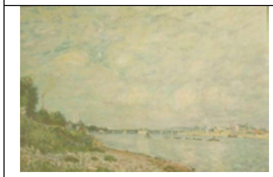
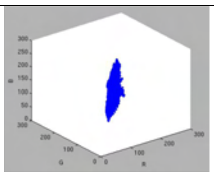
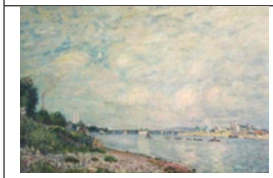
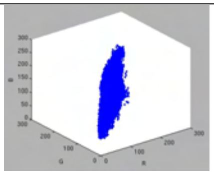
Image	Point cluster	Red Mean	Red Std. Dev.	Red En-tropy	Green Mean	Green Std. Dev.	Green En-tropy	Blue Mean	Blue Std. Dev.	Blue En-tropy
		184.6	20.8	6.032	182.2	19.5	5.825	144	22.7	5.983
		155.9	54.1	7.386	158.2	54.3	7.287	158.0	66.3	7.278

Table 1: Statistics of a pair of degraded and clean paintings

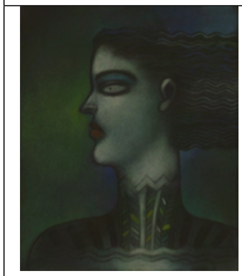
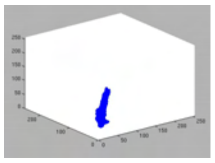
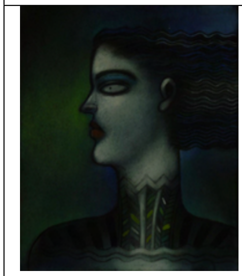
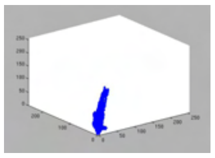
Image	Point cluster	Red Mean	Red Std. Dev.	Red En-tropy	Green Mean	Green Std. Dev.	Green En-tropy	Blue Mean	Blue Std. Dev.	Blue En-tropy
		44.2	13.2	4.818	48.4	15.4	5.424	40.1	13.5	5.095
		21.1	16.8	5.136	27.2	19.5	5.754	23.6	17.1	5.423

Table 2: Statistics of a pair of degraded and clean paintings

method in the wavelet transform domain. They use the Kullback-Leibler(KL) distance [S07, CT91] between the marginal probability distributions of wavelet coefficients of the reference and distorted images as a measure of image distortion. We use KL-divergence values to find out the relative entropy between two color distributions. In order to assess if the new sample painting has similar color distribution to the old panting, we first perform eigen-space transformation followed by KL-divergence calculation. The Kullback-Leibler divergence (also information divergence, relative entropy) is a non-symmetric measure of the difference between two probability distributions P and Q. KL measures the expected number of extra bits required to code samples from P when using a code based on Q, rather than using a code based on P. Typically P represents the "true" distribution of data, observations. The measure Q typically represents an approximation of P.

$$D_{KL}(P||Q) = \sum_i P(i)\ ln\frac{P(i)}{Q(i)}$$

In our approach, we calculate the mean of pixel data along the three axes and compute the covariance matrices between the three components in the color space for both the given old and sample painting. Then, we decompose the covariance matrix using SVD algorithm and obtain the rotation and scaling matrices. Corresponding eigenvalues and eigenvectors are obtained from the covariance matrix. The point clusters of the image is translated to the origin by subtracting the mean obtained along the three reference axes. Eigenvectors are aligned with the reference axis by applying the rotation matrix on all the pixels of the image. Hence, the eigen-vectors of both the old and sample painting are now, aligned with the standard R, G and B axes. Now, we find the KL-divergence values between these transformed distributions. Low values of KL-divergence jus-

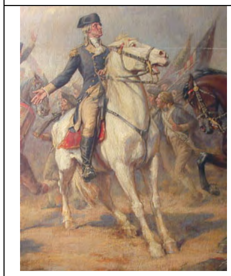
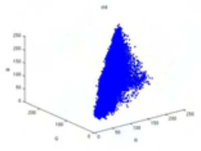
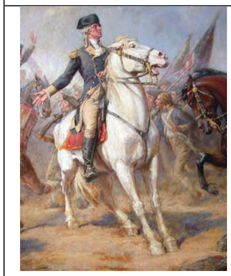
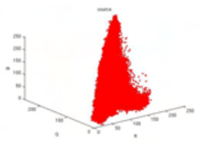
Image	Point cluster	Red Mean	Red Std. Dev.	Red En-tropy	Green Mean	Green Std. Dev.	Green En-tropy	Blue Mean	Blue Std. Dev.	Blue En-tropy
		156.8	40.5	7.332	132.9	38.5	7.281	112.7	34.2	7.105
		156.6	45.8	7.505	138.2	48.5	7.604	126.9	51.5	7.633

Table 3: Statistics of a pair of degraded and clean paintings

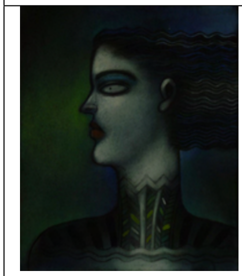
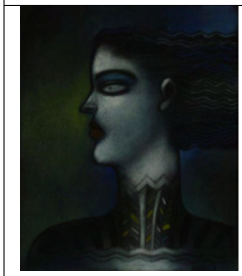
Image	Red En-tropy	Green En-tropy	Blue En-tropy	KL-divergence
	5.136	5.754	5.423	0.43
	5.274	5.812	5.438	0.48

Table 4: Statistics of a pair of manually cleaned and restored paintings

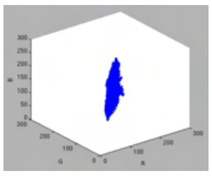
tify the visual similarity of the distributions. Table 4 exhibits the comparison of the values of entropy and KL-divergence of manually cleaned painting and restored painting.

5 USER INTERVENTION IN DIGITAL COLOR RESTORATION

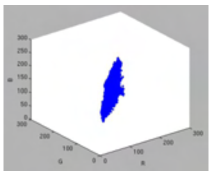
In this section, we incorporate the user interaction in our system. In this way, the user can interactively alter the distribution of the image (point cluster formed by all the pixels of the image in the Red, Green and Blue dimensions) by performing the various operations of rotation, scaling or translation. We use the following approach: we compute the mean for the old image in RGB space. We subtract the mean from all the data points as a result of which, the entire point cluster shifts to the origin. Then, user can scale the data points in all the three dimensions based on his visual perception of the corresponding impact on the painting. Next, we add the averages that we previously subtracted. Hence, the point cluster has now again been shifted to the original mean.

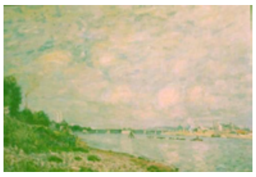
Figure 2 illustrates the above mentioned technique. Figure 2(a) and Figure 2(b) represent the old painting along with its corresponding point cluster. In Figure 2(d), scaling factors are scaled by 1.11 in the Blue dimension, 1.04 in the Green dimension and 1.07 in the Red dimension, Figure 2(c) shows the corresponding effect on the image. Similarly, in Figure 2(e), scaling in the Red dimension is done by a factor of 1.12. In Figure 2(g), further scaling factor of 1.10 in the Green dimension is applied. In Figure 2(i), the Blue dimension is scaled by 1.14. Further, the values in the R, G and B dimensions are normalized in the interval [0:1]. Figure 2(k) is the painting with satisfactory cleaning and Figure 2(j) is its corresponding point cluster. In supplement material, we have included more such examples. Basically, we increase the range of gray levels by performing scaling and translation. In the same way, we try to stretch the range of colors by matching the color spaces of the two paintings to give it a cleaned appearance.

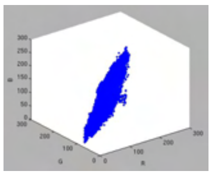
(a) Original painting


(b) Point cluster of (a)

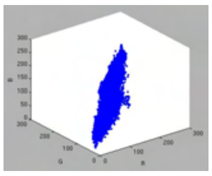
(c)

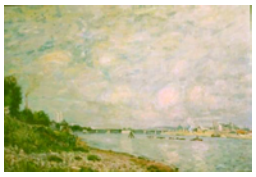

(d)

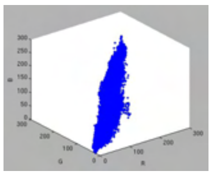

(e)

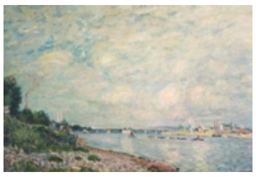

(f)

(g)


(h)


(i)


(j)


(k)

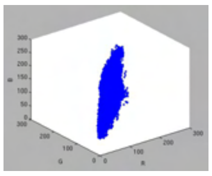

(l) Point cluster of (k)

Figure 2: Paintings and their corresponding point cluster obtained after interactively tweaking the scaling factors in the three R,G,B dimensions.

6 EXAMPLE-BASED DIGITAL COLOR RESTORATION APPROACH

In this section, we present how to match the color space of two paintings to give clean appearance to the old painting using a statistics based method [XM06].

First, calculate the mean of pixel data along all three axes R, G and B for both the old and sample cleaned painting, denoted as $(\bar{R}_{\text{old}}, \bar{G}_{\text{old}}, \bar{B}_{\text{old}})$ and $(\bar{R}_{\text{clean}}, \bar{B}_{\text{clean}}, \bar{G}_{\text{clean}})$ respectively.

Then calculate the covariance matrices for both the paintings Cov_{old} and $\text{Cov}_{\text{clean}}$

$$Cov = \begin{pmatrix} cov(R,R) & cov(R,G) & cov(R,B) \\ cov(R,G) & cov(G,G) & cov(G,B) \\ cov(R,B) & cov(G,B) & cov(B,B) \end{pmatrix}$$

Now, decompose the covariance matrix using singular value decomposition to get U and S which are required to derive rotation & scaling matrices.

$$Cov = U * S * V^{\mathrm{T}}$$

where U and V are unitary matrices and are composed of eigenvectors of covariance matrix, S is a diagonal matrix of eigenvalues of Cov. $S = diag(\lambda^{\mathrm{R}}, \lambda^{\mathrm{G}}, \lambda^{\mathrm{B}})$

Translation matrices

$$T_{\text{old}} = \begin{pmatrix} 1 & 0 & 0 & \bar{R}_{\text{old}} \\ 0 & 1 & 0 & \bar{G}_{\text{old}} \\ 0 & 0 & 1 & \bar{B}_{\text{old}} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$T_{\text{clean}} = \begin{pmatrix} 1 & 0 & 0 & \bar{R}_{\text{clean}} \\ 0 & 1 & 0 & \bar{G}_{\text{clean}} \\ 0 & 0 & 1 & \bar{B}_{\text{clean}} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Rotation matrices

$$R_{\text{old}} = U_{\text{old}}, R_{\text{clean}} = U_{clean}^{-1}$$

Scaling matrices

$$S_{\text{old}} = \begin{pmatrix} \lambda_{old}^{R} & 0 & 0 & 0 \\ 0 & \lambda_{old}^{G} & 0 & 0 \\ 0 & 0 & \lambda_{old}^{B} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$S_{\text{clean}} = \begin{pmatrix} s_{clean}^{R} & 0 & 0 & 0 \\ 0 & s_{clean}^{G} & 0 & 0 \\ 0 & 0 & s_{clean}^{B} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where $s_{clean}^{R} = 1/\sqrt{\lambda_{clean}^{R}}$, $s_{clean}^{G} = 1/\sqrt{\lambda_{clean}^{G}}$, $s_{clean}^{B} = 1/\sqrt{\lambda_{clean}^{B}}$

The final transformation will have the following form:

$$I = T_{clean} . R_{clean} . S_{clean} . S_{old} . R_{old} . T_{old} . I_{old}$$

where $I = (R, G, B, 1)^{T}$ and $I_{tgt} = (R_{tgt}, G_{tgt}, B_{tgt}, 1)^{T}$ denote the homogeneous coordinates of pixel points for the result and old paintings respectively. This transformation is applied to each pixel of the old painting.

We generate the results in $l\alpha\beta$ color space as well. $l\alpha\beta$ color space [RC98] minimizes correlation between channels for many natural scenes. This space is based on data-driven human perception research that assumes the human visual system is ideally suited for processing natural scenes. There is little correlation between the axes in $l\alpha\beta$ space, which lets us apply different operations in different color channels with some confidence that undesirable cross-channel artifacts won't occur. Additionally, this color space is logarithmic, which means to a first approximation that uniform changes in channel intensity tend to be equally detectable. We follow the following approach: the old painting is converted from RGB to $l\alpha\beta$ space. Then, we compute the mean and standard deviation for both the old and sample clean painting in l$\alpha\beta$ space. We subtract the mean from the data points:

$$l^* = l - \langle l \rangle$$
$$\alpha^* = \alpha - \langle \alpha \rangle$$
$$\beta^* = \beta - \langle \beta \rangle$$

Then, we scale the data points comprising the old image by factors determined by the respective standard deviations:

$$l' = \frac{\sigma_t^l}{\sigma_s^l} l^*$$
$$\alpha' = \frac{\sigma_t^\alpha}{\sigma_s^\alpha} \alpha^*$$
$$\beta' = \frac{\sigma_t^\beta}{\sigma_s^\beta} \beta^*$$

where $\sigma_t^l, \sigma_t^\alpha$ and σ_t^β are the standard deviations for the sample clean painting in l, α, β dimensions respectively, and $\sigma_s^l, \sigma_s^\alpha$ and σ_s^β are the standard deviations for the old painting in l, α, β dimensions respectively. After this transformation, the resulting data points have standard deviations that conform to the sample clean painting. Next, instead of adding the averages that we previously subtracted, we add the averages computed for the sample clean painting. Finally, we convert the result back to RGB color space.

As discussed earlier, for changing the appearance of an old painting to a cleaned one, we have to match its color point's cluster to another sample cleaned painting, which has the similar color distribution as the old painting. All in all, these transformations simulate morphing an ellipsoid to fit another one. The center coordinates of an ellipsoid is the mean, and the eigenvectors and eigenvalues of the covariance matrix indicate the directions and length of the three axes of the ellipsoid. In order to assess if the new sample painting has the similar color distribution as the old panting, we perform eigenspace transformation followed by KL-divergence. Table 5 lists the KL-divergence values between the histograms plotted in the 3 dimensions (red, green, blue) for the transformed old painting and the sample painting. Values are obtained for the two test cases. The relative entropy between the old and sample paintings is quite low, this observation justifies the visual similarity of the color distribution of the sample and the old painting.

Channels	**Fig3(c)(KL-divergence**	**Fig3(g)(KL-divergence)**	**Fig3(k)(KL-divergence**
Red channel	0.57	0.41	0.31
Green channel	1.43	0.52	0.74
Blue channel	1.51	1.87	0.36

Table 5: KL-divergence values for the old painting and the chosen sample paintings for Fig 3

Channels	**Fig4(b)(KL-divergence)**	**Fig4(d)(KL-divergence)**
Red channel	0.43	0.48
Green channel	1.90	1.91
Blue channel	0.37	0.94

Table 6: KL-divergence values for the manually cleaned painting and the restored painting with reference to old painting for Fig 4

Channels	**Fig5(b)(KL-divergence)**	**Fig5(d)(KL-divergence)**
Red channel	0.18	0.38
Green channel	0.15	0.62
Blue channel	0.50	0.81

Table 7: KL-divergence values for the manually cleaned painting and the restored painting with reference to old painting for Fig 5

Channels	**Fig7(b)(KL-divergence)**	**Fig7(d)(KL-divergence)**
Red channel	0.17	0.08
Green channel	0.24	0.21
Blue channel	0.28	0.13

Table 8: KL-divergence values for the manually cleaned and the restored painting with respect to the old painting for Fig 7

Figure 3 demonstrates the use of different sample paintings to restore an old painting. A different kind of look and feel is imposed on the old painting depending upon the sample painting chosen. In Figures 4, 5 & 7, an old painting is cleaned using an example clean painting having similar color distribution as that of the old painting. Figure 6 shows a painting partially cleaned using chemicals. The remaining old part is cleaned using the sample painting taken from the cleaned part of the old painting itself for color restoration. The result is shown in (c) which is comparable to the chemically cleaned part. Again, for the purpose of numerical analysis, we adopt the same technique of eigen-space transformation followed by KL-divergence as described above. Tables 6, 7 & 8 lists the KL-divergence values between the histograms plotted in the 3 dimensions (red, green, blue) for the manually cleaned painting and restored painting with reference to the old painting. Since, the values for the example1 painting is the minimum, it supports that its color distribution would correspond most closely to the candidate painting. Thus, it is chosen for the cleaning of the old painting.

7 RESULTS

In order to assess the quality of the color restored images, we apply the above mentioned technique for color restoration on 25 old and oxidized paintings. Evaluation is done by comparing the restoration results with the chemically cleaned paintings. The technique generates comparable results to chemically cleaned paintings depending upon the similarity in color distribution of old and sample cleaned painting. We compare the results generated by our algorithm with the ground truth (chemically cleaned images). Based on the similarity between the color distributions of the ground truth and the chemically cleaned paintings, we infer that our results are visually comparable with the ground truth. We have also incorporated crack restoration. Several efforts have been made in this field of crack restoration [SS10, PS11, GW08, GP98, BS00, OB01, T04, CP04, BB00]. The crack restoration technique consists of the following stages: crack detection and crack filling. Cracks can be detected using various techniques such as top-hat transformation (morphological filter) [SS10], [PS11], thresholding operation or techniques involving user-intervention [GW08]. Cracks can be filled using anisotropic diffusion [GP98], Bertalmio et al's inpainting technique [BS00], Oliveira et al's technique [OB01], Fast Marching Method (FMM) [T04], Exemplar based approach [CP04], Barni et al's crack filling technique[BB00]. We integrate crack detection with color restoration. Figure 8 depicts the original painting which requires both color restoration and crack filling. Initially, the colors of the painting are restored using the above mentioned example-based color restoration approach followed by crack filling using exemplar-based inpainting. Crack restoration results are depicted in Figures 9 & 10. In our approach, cracks are detected using top-hat transformation (disk as a structuring element) and a mask is created. Then, the cracks are filled using exemplar-based inpainting method [CP04]. Results on various other images are included in the supplementary material.

8 CONCLUSION

An analysis on varnish layer is described in the paper to understand how it affects the color space of the old paintings and a technique for digital color restoration is proposed. The simulation performed on the number of paintings indicates that satisfactory results can be obtained if we have a clean painting with the similar color distribution as the old painting.

9 REFERENCES

[XM06] X. Xiao and L. Ma, "Color transfer in correlated color space," *Proceedings of the ACM international conference on Virtual reality continuum and its applications*, pp. 305 -309, August 2006.

[PP00] M. Pappas and I. Pitas, "Digital Color Restoration of Old Paintings," *IEEE Trans. On Image Processing*, vol. 9, no. 2, pp. 291-294, Feb. 2000.

[GP98] I. Giakoumis and I. Pitas, "Digital Restoration of Painting Cracks," *IEEE Int. Symposium on Circuits and Systems (ISCAS)*, vol. 4, pp. 269 - 272, June 1998

[BS00] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image Inpainting," *Proceedings of ACM SIGGRAPH*, pp. 417-424, July 2000

[OB01] M. Oliveira, B. Bowen, R. McKenna, and Y. S. Chang, "Fast Digital Image Inpainting," *Proceedings of the International Conference on Visualization, Imaging and Image Processing*, pp. 261-266, Sept. 2001

[HJ01] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless and D. H. Salesin, "Image analogies," *Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01*, pp. 327 - 340, Aug. 2001

[T04] A. Telea, "An Image Inpainting Technique Based on the Fast Marching Method," *Journal of graphics, gpu, and game tools*, vol. 9, no. 1, pp. 23-34, 2004.

[CP04] A. Criminisi, P. Pérez and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on Image Processing*, vol. 13, issue. 9, pp. 1200-1212, Sept. 2004.

[GW08] R.C. Gonzalez and R.E. Woods, *"Digital Image Processing"*, 3rd edition, Prentice Hall, 2008.

[PS11] C. T. Palomero and M. N. Soriano, "Digital cleaning and "dirt" layer visualization of an oil painting," *Optics Express 21011*, vol. 19, no. 21, pp. 21011-21017, 2011.

[SS10] G. S. Spagnolo, F. Somma, "Virtual restoration of cracks in digitized image of paintings," *International Conference on Defects in Insulating Material, Journal of Physics: Conference Series 249 012059*, no. 1, 2010.

[PS11] L. Pezzati, R. Salimbeni, "Virtual restoration: detection and removal of craquelure in digitized image of old paintings," *Proceedings of the SPIE 8084*, pp. 80840B-80840B-8, 2011.

[BB00] M. Barni, F. Bartolini, and V. Cappellini, "Image Processing for Virtual Restoration of Artworks," *IEEE Multimedia*, no. 2, pp. 34-37, 2000.

[ZV09] J. Zapletal, P. Vanecek, V. Skala, "RBF-based

Image Restoration Utilising Auxiliary Points," *CGI 2009 proceedings, ACM ISBN 978-60558-687-8*, pp. 39-44, 2009.

[GS03] F. Gasparini and R. Schettini, "Color Correction for Digital Photographs," *International Conference on Image Analysis and Processing (ICIAP'03) ISBN 0-7695-1948-2*, pp. 646-651, Sept. 2003.

[RA01] E. Reinhard, M. Ashikhmin, B. Gooch and P. Shirley, "Color Transfer between Images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34-41, Oct. 2001.

[S07] Jonathon Shlens, *"Tutorial on Kullback-Leibler Divergence and Likelihood Theory"*, Aug. 2007.

[CT91] T. M. Cover and J. A. Thomas, *"Elements of information theory"*, Wiley, New York, 1991.

[RC98] D.L. Ruderman, T.W. Cronin, and C.C. Chiao, "Statistics of Cone Responses to Natural Images: Implications for Visual Coding," *J. Optical Soc. of America*, vol. 15, no. 8, pp. 2036-2045, 1998.

[SB05] H. R. Sheikh, A. C. Bovik and G. De Veciana, "An Information Fidelity Criterion for Image Quality Assessment using Natural Scene Statistics," *IEEE Trans. on Image Processing*, vol. 14, no. 12, Dec. 2005.

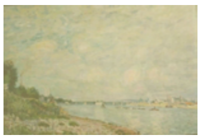
[SB06] H. R. Sheikh and A. C. Bovik, "Image Information and Visual Quality," *IEEE Trans. on Image Processing*, vol. 15, no. 2, Feb. 2006.

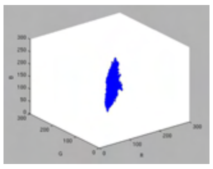
[ZJ06] D. Zhang and E. Jernigan, "An Information Theoretic Criterion for Image Quality Assessment based on Natural Scene Statistics," *Proceedings of IEEE ICIP 2006*, Oct. 2006.

[WS05] Z. Wang and E. P. Simoncelli, "Reduced-Reference Image Quality Assessment using a Wavelet-Domain Natural Image Statistic Model," *Proceedings of SPIE Human Vision and Electronic Imaging X*, vol. 5666, Jan. 2005.

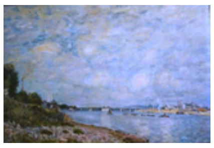
[WL11] Z. Wang and Q. Li, "Information Content Weighting for Perceptual Image Quality Assessment," *IEEE Trans. on Image Processing*, vol. 20, no. 5, pp. 1185-1198, May 2011.

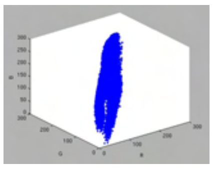
[MS01] H. Maitre, F. Schmitt and C. Lahanier, "15 years of image processing and the fine arts," *Proceedings of International Conference on Image Processing, 2001*, no. 1, pp. 557-561, 2001.

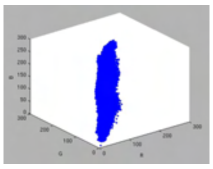

(a) Original painting


(b) Point cluster of (a)

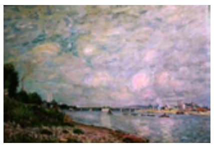
(c) Example painting1

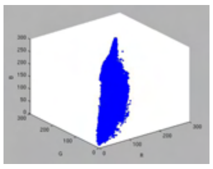

(d) Result using (c)

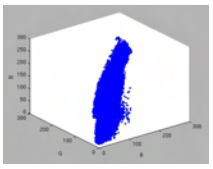

(e) Point cluster of (c)


(f) Point cluster of (d)

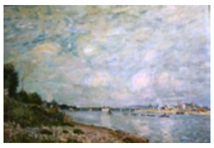
(g) Example painting2

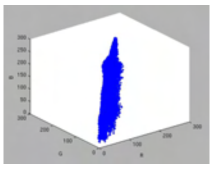

(h) Result using (g)


(i) Point cluster of (g)


(j) Point cluster of (h)

(k) Example painting3


(l) Result using (k)


(m) Point cluster of (k)

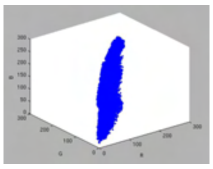
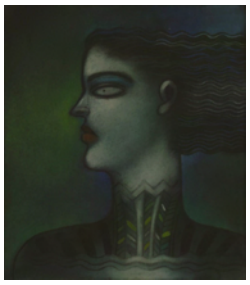
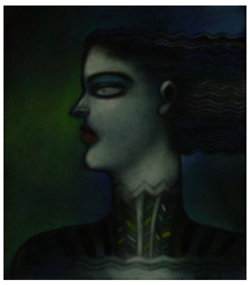
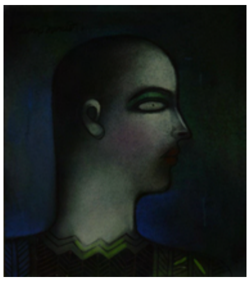

(n) Point cluster of (l)

Figure 3: Result to demonstrate the use of different clean paintings for color restoration of an old painting.


(a) Old Painting


(b) Chemically cleaned painting


(c) Example painting

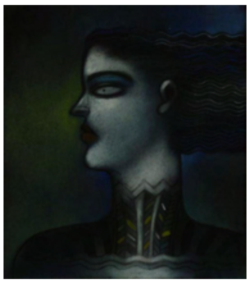
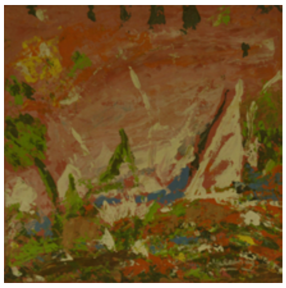
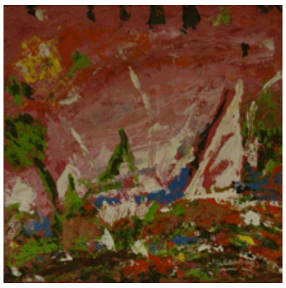
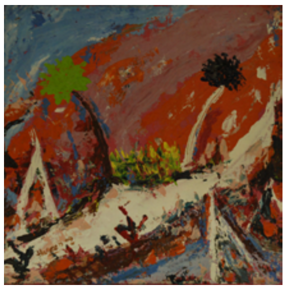

(d) Restored painting

Figure 4: Result of color restoration


(a) Old painting


(b) Chemically cleaned painting


(c) Example painting

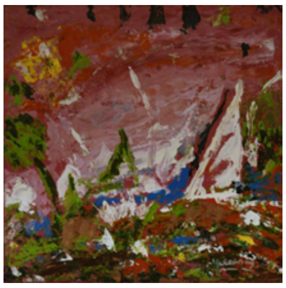
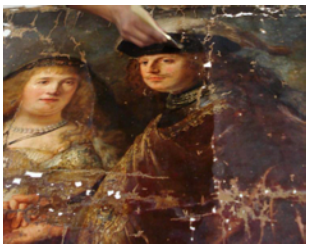
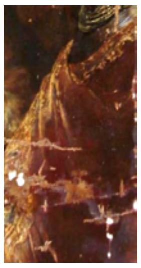

(d) Result

Figure 5: Result of color restoration


(a) Old painting under process of chemical cleaning


(b) cleaned part of the painting

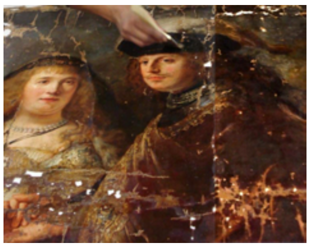
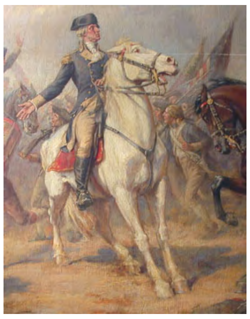
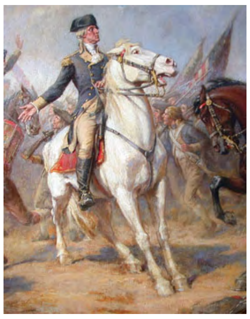
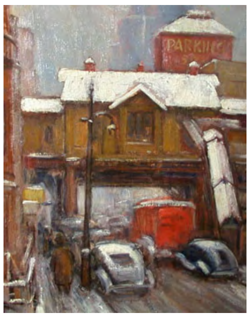

(c) Remaining part is cleaned using digital process

Figure 6: Restoration using cleaned part as sample cleaned painting to complete the cleaning process


(a) Original painting


(b) Chemically cleaned painting


(c) Example painting

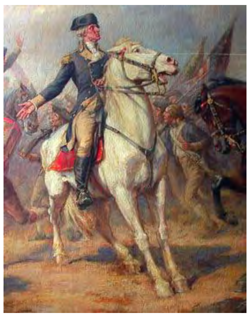
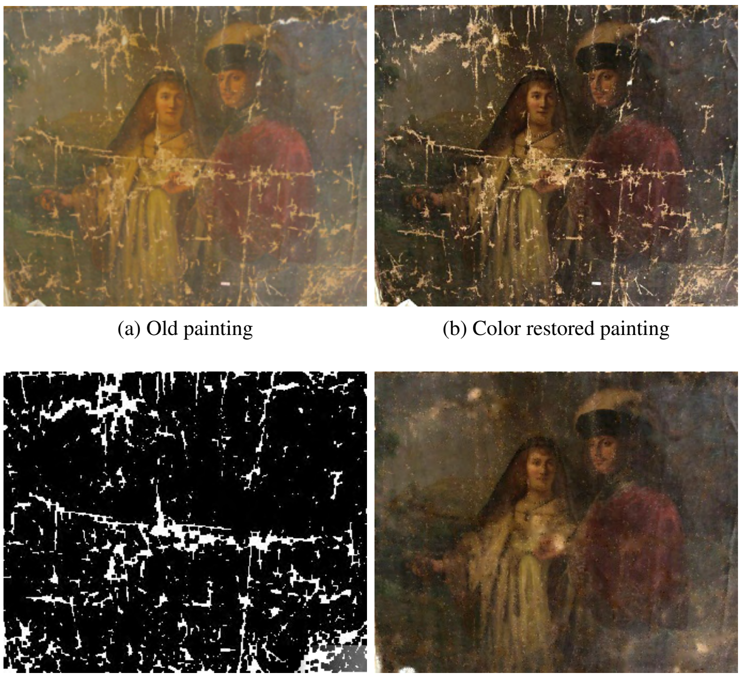

(d) Restored result

Figure 7: Result of color restoration


(a) Old painting (b) Color restored painting

(c) Mask (d) After crack filling

Figure 8: Integration of color restoration and crack filling

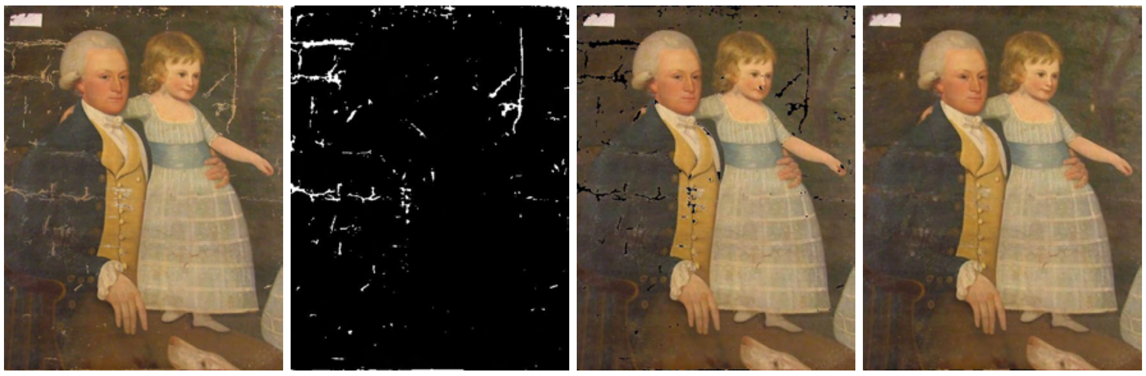

(a) Original painting (b) Result of crack detection (c) Detected cracks on original painting (d) Result of crack filling

Figure 9: Crack detection and filling

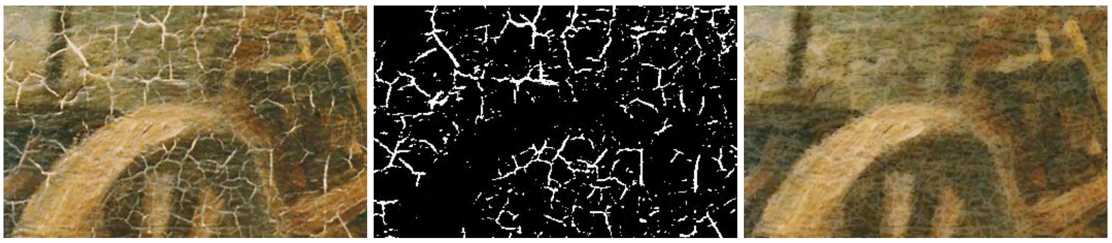

(a) Original image (b) Cracks detected in the image (c) Result after crack filling

Figure 10: Restoration of cracks

Multi Scale Color Coding of Fluid Flow Mixing Indicators along Integration Lines

Sandeep Khurana
Department of Computer Science[1]
Center for Computation & Technology[1]
skhura1@tigers.lsu.edu

Nathan Brener
Bijaya Karki
Department of Computer Science[1]
brener@csc.lsu.edu
karki@csc.lsu.edu

Werner Benger
Center for Computation & Technology[1]
Institute for Astro- and Particle Physics[2]
werner@cct.lsu.edu
Werner.Benger@uibk.ac.at

Somnath Roy[4]
Sumanta Acharya[1]
Dept. of Mechanical Engineering
somnath@iitp.ac.in
acharya@me.lsu.edu

Marcel Ritter
Inst. of Basic Sciences in Civil Engineering[2]
marcel.ritter@uibk.ac.at

S. Sitharama Iyengar
School of Computing & Info. Sciences[3]
iyengar@cis.fiu.edu

ABSTRACT

This paper presents a multi scale color coding technique which enhances the visualization of scalar fields along integration lines in vector fields. In particular, this multi scale technique enables one to see detailed variations in selected small ranges of the scalar field while at the same time allowing one to observe the entire range of values of the scalar field. This type of visualization, observing small variations as well as the entire range of values, is usually not possible with uniform color coding. This multi scale approach, which is linear within each division of the scale (piecewise linear), is a general visualization technique that can be applied to many different scalar fields of interest. As an example, in this paper we apply it to the visualization of fluid flow mixing indicators along pathlines in computational fluid dynamics (CFD) simulations. Pathlines are the trajectories of fluid particles over time in the CFD simulations, and applying multi scale color coding on the pathlines brings out quantities of interest in the flow such as curvature, torsion and specific measure of length generation, which are indicators of the degree of mixing in the fluid system. In contrast to uniform color coding, this multi scale scheme can display small variations in the mixing indicators and still show the entire range of values of these indicators. In this paper, the mixing indicators are computed and displayed only along line structures (pathlines) in the flow field rather than at all points in the flow field.

Keywords: transfer function generation, pathlines, nonlinear color map, computational fluid dynamics, vector field visualization, mixing indicators

1 INTRODUCTION

1.1 Motivation

This paper proposes a multi scale color coding scheme which can display nonuniform distributions of quantities that have small variations in some ranges and larger variations in other ranges. The paper shows the advantages of this multi scale technique compared to uniform color coding which usually cannot display both small and large variations in the same image. In this multi scale approach, the overall scale is divided up into intervals specified by the user, with a different linear scale inside each interval (i.e., it is piecewise linear). This technique enables one to see small variations in selected ranges of the quantity of interest and also observe the entire range of values of this quantity. This multi scale scheme is a general technique that can be used to visualize any scalar quantity of interest. As an example of this method, we use it to visualize fluid mixing indicators along integration lines (pathlines) in computational fluid dynamics simulations of fluid flow. In this paper the mixing indicators are computed and displayed only along the pathlines rather than at all points in the flow field.

[1] Louisiana State University, Baton Rouge, LA-70803, USA
[2] University of Innsbruck, Technikerstraße 25/8, A-6020 Innsbruck, Austria
[3] Florida International University, Miami, FL-33199, USA
[4] Indian Institute of Technology, Patna, India

Computational fluid dynamics (CFD) uses numerical methods and analysis techniques to study fluid flow. The computational results from numerical models can yield data which is on the order of hundreds of gigabytes and consists of multiple blocks containing millions of cells. Computation on such a grid can take a substantial amount of time to trace the paths of fluid particles over time (pathlines). But once these pathlines have been calculated, fluid mixing indicators, such as curvature and torsion, can then be quickly computed and visualized at every point on the pathlines. Curvature and torsion computations along pathlines in a stirred tank CFD simulation were done earlier [3] and uniformly color coded, but the distribution of these values is not uniform. In this paper the multi scale color coding technique is used to display the nonuniform distributions of these quantities. The paper also adds another mixing indicator, the specific measure of length generation, to complement the curvature and torsion so as to better understand the mixing. In addition it uses higher order difference formulas to improve the accuracy of the curvature and torsion calculations. As mentioned above, the multi scale color coding scheme is a general method that can be used to visualize any scalar field of interest, such as pressure, which is also shown in the paper.

1.2 Fluid Mixing Indicators

In order to develop an understanding of the mechanical macro mixing process, we investigate the motion of the fluid particles in the flow domain. The flow domain considered is a mechanically agitated turbulent stirred tank with down-pumping pitched blade impellers. Water is considered as the working fluid in which a blob of a tracer is injected at a particular location. We look into the pathlines of a number of points associated with the injected tracer. These pathlines show exactly how tracer particles move inside the stirred tank volume. From the pathline visualizations, we try to qualitatively understand the stretching and distortions on the fluid elements by calculating the curvature and torsion at different points along the pathlines. A higher value of the curvature indicates more straining of the fluid element associated with it and also higher residence time of the fluid particle in that particular region resulting in prolonged inter-diffusion of the fluids, which promotes mixing [12]. The torsion is a measure of twisting strains on the fluid elements which also promotes the break up of the clumps and enhances mixing. Torsion can be positive or negative, but the degree of mixing in the stirred tank depends primarily on the magnitude of the torsion rather than its sign. Therefore, for visualizing torsion, we will show its absolute value (magnitude), neglecting whether the twist is clockwise or counterclockwise.

The curvature κ and torsion τ are given by

$$\kappa = \frac{|\ddot{q} \times \dot{q}|}{|\dot{q}|^3} \quad . \tag{1}$$

$$\tau = \frac{\dddot{q} \cdot (\ddot{q} \times \dot{q})}{|\ddot{q} \times \dot{q}|^2} \quad . \tag{2}$$

where $\dot{q}$ is the velocity of the fluid (vector field) and $\ddot{q}$ and $\dddot{q}$ are the first and second derivatives of the vector field with respect to time. In our approach we compute pathlines from the vector field first, based on user-specified initially defined seeding points, and then calculate curvature and torsion along the pathlines as a post-processing step.

Another measure of fluid mixing is the specific measure of length generation [13] defined as

$$\frac{d\ln(\lambda)}{dt} \tag{3}$$

where $\lambda = \frac{\delta}{L}$, δ being the increase in path length (which is proportional to the magnitude of the velocity) and L being the total path length (before the δ increment) at a particular time t. The proportional change of the path length will be larger at the start of the pathlines and smaller at their ends, thus this mixing indicator will be dominated by variations of the velocity as the pathlines get longer. The greater the specific measure of length generation, the greater the mixing. This quantity is also calculated as a post-processing step on the pathlines. It should be noted that in addition to the mixing indicators described above, there are also other principles that can be used to investigate the mixing of fluids.

1.3 Related Work

Automated generation of color maps has been the subject of research in scientific visualization for decades. Yet, still many visualization tools rely on manual specification of colors as the problem is highly application and problem specific. Taking over full control of color map generation from the user is usually not even desirable, leaving semi-automated assistance of color specification as the best compromise. A well known approach is the method of finding material boundaries by G. Kindlmann [8]. In this approach a Histogram Volume Structure is created which measures the relationship between the data values and their derivatives. In this volume, the three axes are f, f' and f'' , each axis is divided into some number of bins which contains the number of voxels falling in the same combination of ranges of these variables. This information facilitates a high-level interface to opacity function creation. In the winning entry of the IEEE Visualization 2004 Contest [6], the visualization system VisSym was highlighted, where the subset of data items were manually selected using techniques for information visualization such as scatter plots and histograms. These values were related to the dataset, e.g., for detecting the eye of the hurricane area, cells which exhibit low velocity and pressure regions were brushed out.

Our main objective is the depiction of scalar-valued fluid flow mixing indicators along pathlines. Fluid flow quantities such as curvature and torsion are common subjects of study. Weinkauf [15] demonstrated computation of curvature and torsion as a field over the entire data domain, allowing one to identify regions of interest within the full spatial domain. Due to the sheer amount of dynamic data in our application, which comprises about 300GB of binary data, this approach is not feasible and we need to restrict ourselves to displaying such field quantities on precomputed pathlines. One method for doing this is to use a geometric object [5] to visualize the fluid fields along the pathlines, however, if the number of pathlines is large, this approach is not visually appealing because the image is more cluttered and does not give global information about the flow. In the approach proposed here, we employ a simple but efficient way to display data values with a wide numerical range by specifying color mapping via histogram percentage rather than absolute data values. Rendering of these pathlines using the illuminated streamlines method [17] gives a 3D view of the lines using Phong Shading. The depth perception can be further enhanced by using the halos described in [7, 11] and we plan to implement this halo feature in future work on color coding line structures.

2 OUR APPROACH

The vector field in the stirred tank simulations produced scalar fields, such as curvature and torsion, which have a wide range of values in some parts of the stirred tank. The use of uniform color coding for the whole range of these scalar values often gives a single predominant color along most of the length of the pathlines, making it impossible to visualize small variations in these values. This paper describes the use of a multi scale color coding scheme which not only shows small variations in the scalar field but also enables one to see the entire range of values of the scalar field by just looking at the color coded pathlines. In this paper, the curvature and torsion, and also a new mixing indicator, the specific measure of length generation, are calculated for 300 time steps using more accurate higher order difference formulas and are then visualized using the multi scale color coding technique. Also another scalar quantity of interest, the pressure, is visualized using this approach. This multi scale color coding scheme enables one to easily draw conclusions about the behavior of the scalar fields without any uncertainty about the variations in the small values.

2.1 Higher Order Differentiation Schemes

The formulas for the curvature, eqn. (1), and torsion, eqn. (2), contain the first and second derivatives of the velocity with respect to time, and the specific measure of length generation is defined as the first derivative of $\ln(\lambda)$ with respect to time, eqn. (3). In this work we computed these derivatives with more accurate higher order difference formulas compared to previous work [3,4]. We used the following central difference formulas with fourth order accuracy for the first and second derivatives:

$$f'(x_0) \approx \left(\frac{1}{12} f(x_{-2}) - \frac{2}{3} f(x_{-1}) + \frac{2}{3} f(x_1) - \frac{1}{12} f(x_2) \right) \Big/ h_x \tag{4}$$

$$f''(x_0) \approx \left(-\frac{1}{12} f(x_{-2}) + \frac{2}{3} f(x_{-1}) - \frac{5}{2} f(x_0) + \frac{4}{3} f(x_1) - \frac{1}{12} f(x_2) \right) \Big/ h_x^2 \tag{5}$$

2.2 Multi Scale Color Coding

Designing a multi scale for color coding depends on several parameters like the number of divisions in the scale, the range of the scalar values in each division and the selection of the color range for each division. Generally one needs to visualize the entire range of scalar field values including small, medium and large values, but if the range of values is large, as in the case of curvature and torsion, uniform color coding often gives a single predominant color for most of the pathline which makes it difficult to distinguish between these values. For the mixing indicators described in this paper, we need to see the small and medium values in addition to the large values in order to determine where the mixing speed in the stirred tank is slow, intermediate and fast. Once the ranges of scalar values that need to be visualized in detail have been determined, one can then specify the number of divisions in the multi scale and the range of values and the color range for each division. There are a few considerations for the colors [9,16] that need to be addressed, including the fact that the human eye perceives more variation in the green than in the other colors. As an example of this, Figs. 6 and 7 contain a lot of green which enables one to readily see the variations in the scalar values. Other than these considerations, the colors can generally be chosen as the user desires.

Various techniques can be used to construct the multi scale. In the method presented here, the number of divisions in the multi scale and the range of scalar values in each division are determined manually so as to facilitate the visualization of the scalar field variations of interest. A cumulative histogram of the scalar values is used to determine the boundaries of the divisions so that each one contains the desired percentage of scalar values. In this paper, all of the multi scales have three divisions and the same range of colors, but these are just examples to illustrate the multi scale method. In general the user can choose any number of divisions and specify any range of scalar values and any range of colors for each division in order to enhance the visualization of the scalar values of interest. Currently this process is not automated but rather is done manually by the user. There also exists other work in this area which describes ways to come up with ranges of values from the scalar field. These involve automatic [10], semi-automatic [8] and manual generation of the functions to do this task [14].

Here, since we need to visualize small, medium and large values of the scalar field, three divisions were selected for the multi scale color coding. An example of these three divisions is given in Fig. 1, where the first, second and third divisions go from 0% to 10%, 10% to 30%, and 30% to 90% of the scalar values, and the third division is capped at 90%, with values above 90% being set to the 90% value. Capping the third division at 90% enables one to see the variation of the values between 30% and 90% in greater detail, which is more relevant to the mixing process than the variation of the large values between 90% and 100%. Each division has linear color gradients of R, G and B going from the left end point to the right end point of the division, but the overall color variation over all three divisions is not uniform but rather is multi scale in order to better show the variations in the small and medium scalar values. In the example shown in Figure 1, in the first division R varies linearly from 0 to 255, G=0, and B varies linearly from 255 to 0, in the second division R=255, G varies linearly from 0 to 255, and B=0, and in the third division R varies linearly from 255 to 0, G=255, and B=0. These color ranges for the three divisions were used to color code all four of the scalar fields shown in the paper, while the percents used for the boundaries between the divisions were different for each scalar field.

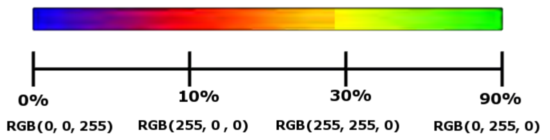


Figure 1: An example of Multi Scale Color Coding with different ranges of scalar values to distinguish regions of low values and medium values.

2.3 Rendering

For the final rendering step, we employ the method of illuminated streamlines [17], which provides improved depth perception as compared to homogeneously colored lines. While in its original form only monochrome illumination could be supported by OpenGL 1.0 hardware, modern OpenGL implementations support multi-texturing and thus allow an additional one-dimensional lighting model as proposed in [17]. Illumination of lines is done by using the Phong Reflection model which defines the intensity (I) at any point as

$$\begin{aligned} I &= I_{ambient} + I_{diffuse} + I_{specular} \\ &= k_a + k_d L \cdot N + k_s (V \cdot R)^n \end{aligned} \tag{6}$$

where L denotes the light direction, V the viewing direction and R the unit reflection vector (the vector in the L-N-plane with the same angle to the surface normal as the incident light). k_a, k_d and k_s are the ambient, diffuse and specular reflection constants respectively. For lines the normals and reflection vectors are undefined. So, intensity can be made dependent on the tangential vector T instead as

$$L \cdot N = |L_N| = \sqrt{1 - (L \cdot T)^2} \tag{7}$$

and

$$\begin{aligned} V \cdot R =& (L \cdot T)(V \cdot T) - \\ & \sqrt{1-(L\cdot T)^2}\sqrt{1-(V\cdot T)^2} \quad . \end{aligned} \tag{8}$$

In order to enhance the color resolution, we use procedural color maps that are merely parametrized through linear functions between each key value. Generally, this linear mapping will be sufficient for each range of scalar values, but if a particular range needs to be subdivided, then that particular range can have more than one linear scale and additional colors. In the approach we have discussed, some idea of the scalar values can be derived from the histogram where, if more points are biased to a single value in that range, then nonlinear functions can be applied to map values to color for the single range. Only at the rendering step is the color map discretized into a texture, for instance one with 256 entries if sufficient, but a higher color resolution can be used if required.

3 RESULTS

We used the the multi scale color coding technique described above to color code the pathlines in the stirred tank. The main idea is to visualize the pathlines and color code them with different scalar quantities that indicate the mixing of the fluids. The quantities that were color coded are curvature, torsion, specific measure of length generation, and pressure. These are shown in separate sections which compare the uniform and multi scale color coding methods for each quantity. The pathlines are generated by introducing 36 seeding points or 516 seeding points on two spheres placed symmetrically about the z-axis in the stirred tank and traced for 300 timesteps totaling about 1.8 secs.

3.1 Curvature

The curvature and torsion values cover a large range and thus uniform color coding cannot represent the whole range efficiently and can be biased to a particular range of values. Even if the values are capped at a certain value, small variations are not visible. Figs. 2 and 3 compare the uniform and multi scale color coding methods for curvature images, and a lot of visible differences can be found.

From Fig. 2 it is clear that the uniform and multi scale color coding methods generate different images even when the range of scalar values is the same. One can see that, in general, multi scale color coding (Fig. 2(b)) gives more noticeable changes in color for the different ranges of scalar values as compared to uniform color coding. This is because variations within specified ranges of scalar values are highlighted in the multi scale technique but remain visibly indifferent in the case of uniform color coding.

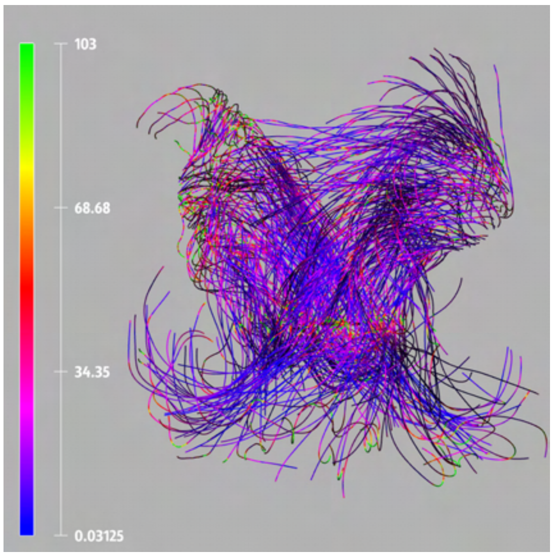
3.2 Torsion

Fig. 4 shows the uniform and multi scale color coded values of the magnitude of the torsion along pathlines in the stirred tank, and a wide range of colors can be seen in the multi scale image, indicating large variations in the torsion values even for pathline points that are close together. In contrast, the uniform color coded image is dominated by a small range of color along most of the length of the pathlines and hence one cannot see the large variations in the torsion.

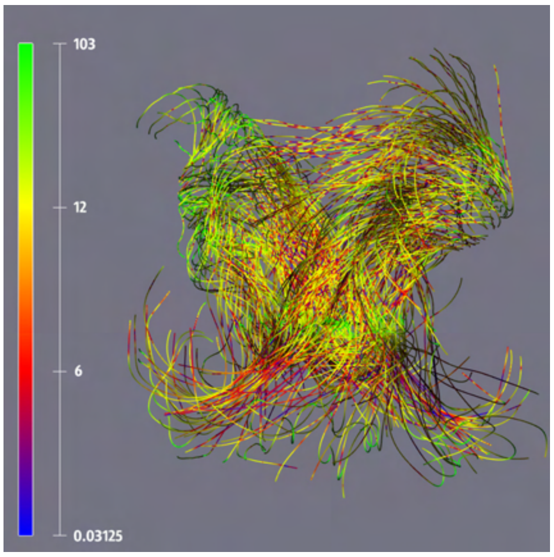
Alternatively, we may also employ Frenet Ribbons to geometrically encode the curvature and torsion [2] of a line in addition to its colorization. Here, the ribbons are rendered instead of the lines, which provide one more degree of freedom to the geometry, i.e., surface. Torsion can easily be visualized using ribbons, employing a geometry shader to generate the actual rendering primitives on the GPU with minimal performance impact as compared to rendering lines. In Fig. 5 we have used multi scale color coding to represent torsion. Areas with a lot of twist correlate to high torsion whereas the areas which seem consistently straight might have different colors due to changes in their torsion values.

3.3 Specific Measure of Length Generation

Fig. 6 shows the specific measure of length generation color coded using the uniform and multi scale methods.



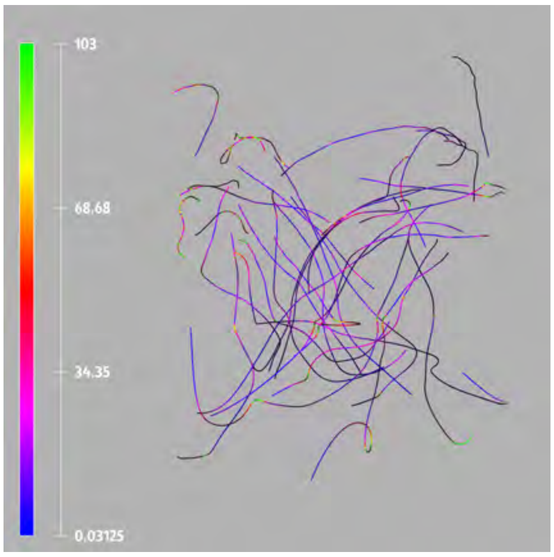
(a) Uniform color coding for curvature



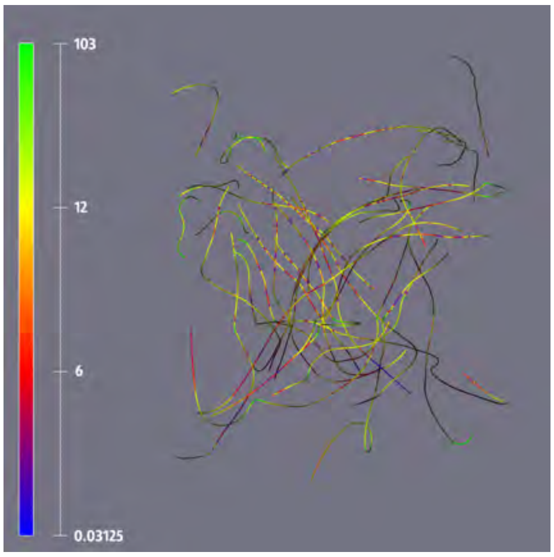
(b) Multi scale color coding for curvature

Figure 2: Comparison of uniform and multi scale color coding of curvature values on 516 pathlines for 300 time slices of stirred tank data. The curvature values were capped at the 95% value in both of the images. For the multi scale technique, the scale has three partitions: 0% to 11%, 11% to 32%, and 32% to 95%, which have the color ranges blue to red, red to yellow and yellow to green respectively.

In the uniform color coded image, almost the entire image is a single color so that one cannot see any variation in the specific measure of length generation values, while the multi scale image has more colors and shows variations in the values of this quantity.

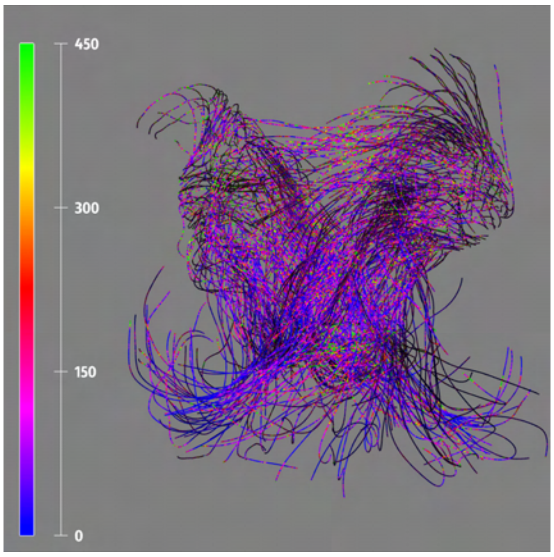


(a) Uniform color coding for curvature

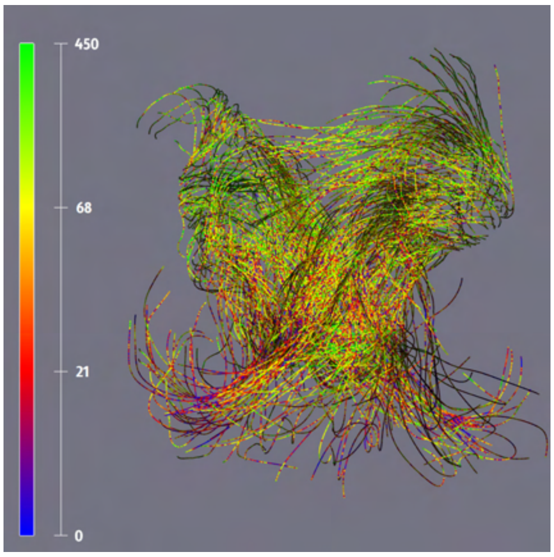


(b) Multi scale color coding for curvature

Figure 3: Comparison of uniform and multi scale color coding of curvature values on 36 pathlines for 300 time slices of stirred tank data. The curvature values were capped at the 95% value in both of the images. For the multi scale technique, the scale has three partitions: 0% to 11%, 11% to 32%, and 32% to 95%, which have the color ranges blue to red, red to yellow and yellow to green respectively.



(a) Uniform color coding for torsion



(b) Multi scale color coding for torsion

Figure 4: Comparison of uniform and multi scale color coding of torsion values on 516 pathlines for 300 time slices of stirred tank data. The values were capped at the 90% value in both the images. For the multi scale technique, the scale has three divisions: 0% to 15%, 15% to 40% and 40% to 90%, which have the color ranges blue to red, red to yellow and yellow to green.

3.4 Pressure

The pressure field is a scalar quantity of interest which is given at all of the grid points in the stirred tank simulation data set. Fig. 7 displays the values of the pressure using the uniform and multi scale color coding methods. As in Fig. 6, the uniform image is mostly a single color and thus shows very little variation in the pressure, while the multi scale image has more colors and shows significant variations in the pressure.

4 DOMAIN EXPERT REVIEW

Images for all of the fluid mixing indicators considered here are generated using both the multi scale and the uniform color coding techniques. The multi scale images can be observed for analysis of mixing performance as they are capable of displaying both the well

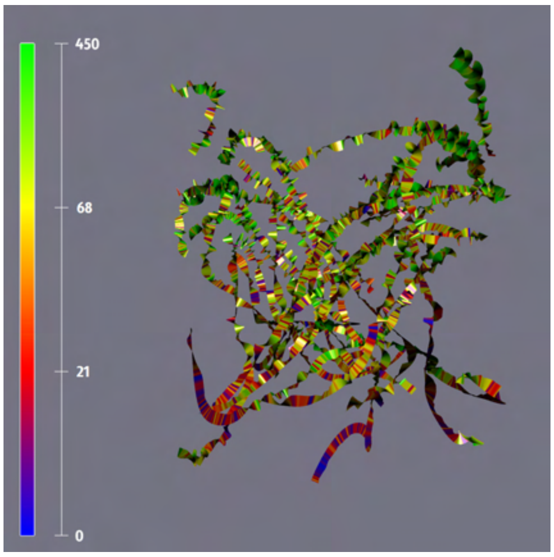
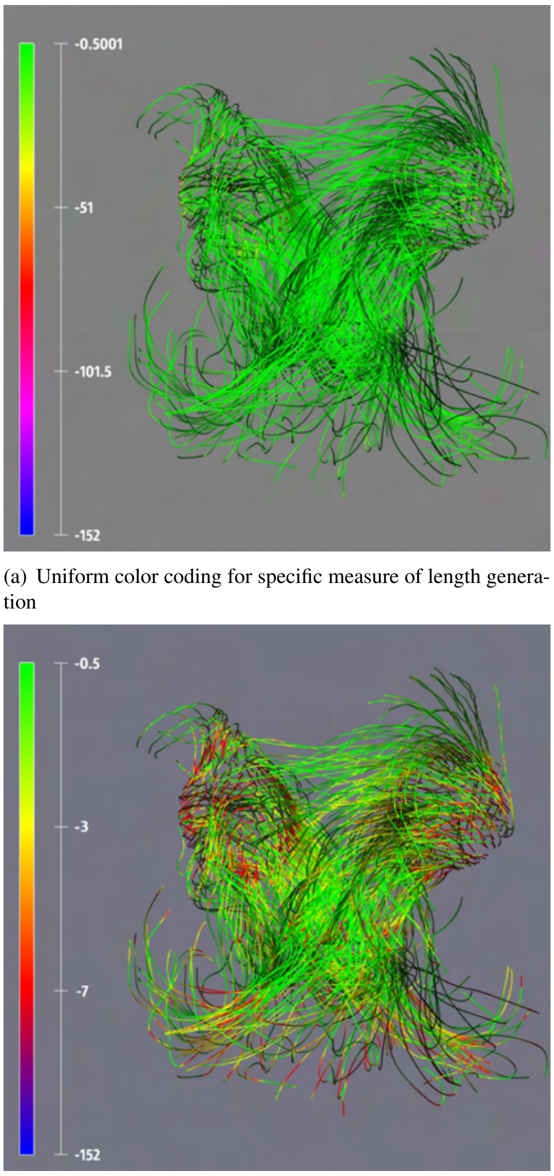


Figure 5: Multi scale color coding of torsion values on the pathlines visualized as ribbons. Values used to color code are the same as in Fig. 4(b)

and poorly mixed zones. The larger values of the torsion, curvature or specific measure of length generation of the pathlines indicate the areas where the mixing is greater. There are some lines which can be observed to have the same green color in all of the multi scale images, indicating high mixing along that particular locus of the tracer particle. Similarly, certain lines can be found which have low values for all three mixing factors, indicating lower mixing in those regions of the stirred tank. These types of observations are not possible in the corresponding uniformly color coded images as they are visually incapable of providing a definitive analysis of the scalar fields.

5 CONCLUSION

The benefits of using a multi scale color coding scheme for the study of fluid flow, such as flows in large scale computational fluid dynamics simulations of a stirred tank, have been described in this paper. This technique allows one to identify areas of local variations in quantities of interest while also obtaining a global view of these quantities. Color coding of various mixing indicators along the fluid flow lines has been demonstrated. We provided a comparative study between the uniform and multi scale color coding methods for these pathline images, demonstrating that the mixing in the stirred tank can be visualized more clearly and effectively using the multi scale technique. This method has been implemented in the VISH visualization system [1] and yields an interactive way to study the fluid system in 3D.



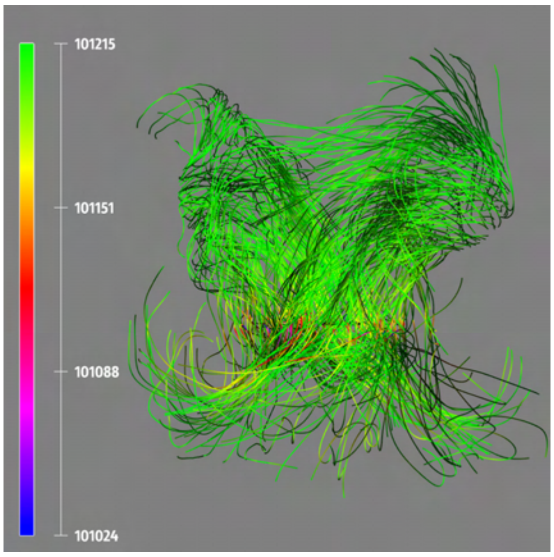
(a) Uniform color coding for specific measure of length generation

(b) Multi scale color coding for specific measure of length generation

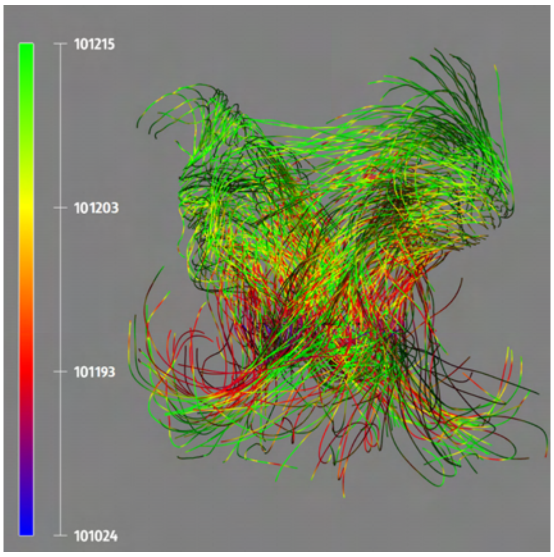
Figure 6: Comparison of uniform and multi scale color coding of specific measure of length generation on 516 pathlines for 300 time slices of stirred tank data. The values were capped at the 70% value in both of the images. For the multi scale technique, the scale has three partitions: 0% to 10%, 10% to 30%, and 30% to 70%, which have the color ranges blue to red, red to yellow and yellow to green respectively.

6 ACKNOWLEDGMENTS

This research employed resources of the Center for Computation & Technology at Louisiana State University, which is supported by funding from the Louisiana Legislature's Information Technology Initiative. This work was supported by the Austrian Science Foundation FWF DK+ project Computational Interdisciplinary



(a) Uniform color coding for pressure



(b) Multi scale color coding for pressure

Figure 7: Comparison of uniform and multi scale color coding of pressure on 516 pathlines for 300 time slices of stirred tank data. The values were capped at the 71% value in both of the images. For the multi scale technique, the scale has three partitions: 0% to 11%, 11% to 30%, and 30% to 71%, which have the color ranges blue to red, red to yellow and yellow to green respectively.

Modeling (W1227) and grant P19300. It was also supported by the Austrian Ministry of Science BMWF as part of the UniInfrastrukturprogramm of the Forschungsplattform Scientific Computing at LFU Innsbruck. In addition it was supported by an Oil Spill grant from BP through LSU, and by a grant from the Louisiana Board of Regents Post Katrina Support Fund Initiative.

REFERENCES

[1] W. Benger, G. Ritter, and R. Heinzl. The Concepts of VISH. In 4th *High-End Visualization Workshop, Obergurgl, Tyrol, Austria, June 18-21, 2007*, pages 26–39. Berlin, Lehmanns Media-LOB.de, 2007.

[2] W. Benger and M. Ritter. Using Geometric Algebra for Visualizing Integral Curves. In E. M. Hitzer and V. Skala, editors, *GraVisMa 2010 - Computer Graphics, Vision and Mathematics for Scientific Computing*. Union Agency - Science Press, 2010.

[3] W. Benger, M. Ritter, S. Acharya, S. Roy, and F. Jijao. Fiberbundle-based visualization of a stir tank fluid. In 17th *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 117–124, 2009.

[4] B. Bohara, F. Harhad, W. Benger, N. Brener, B. Karki, S. Iyengar, M. Ritter, K. Liu, B. Ullmer, N. Shetty, V. Natesan, C. Cruz-Neira, S. Acharya, and S. Roy. Evolving time surfaces in a virtual stirred tank. *Journal of WSCG*, 18(1-3):121–128, 2010.

[5] W. de Leeuw and J. van Wijk. A probe for local flow field visualization. In *Visualization, 1993. Visualization '93, Proceedings., IEEE Conference on*, pages 39 –45, oct 1993.

[6] H. Doleisch, P. Muigg, and H. Hauser. IEEE Visualization 2004 Contest Entry - Interactive Visual Analysis of Hurricane Isabel with SimVis. `http://vis.computer.org/vis2004contest/vrvis/`, 2004.

[7] M. Everts, H. Bekker, J. Roerdink, and T. Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1299 – 1306, nov.-dec. 2009.

[8] G. Kindlmann and J. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Volume Visualization, 1998. IEEE Symposium on*, pages 79 –86, oct. 1998.

[9] G. Kindlmann, E. Reinhard, and S. Creem. Face-based luminance matching for perceptual colormap generation. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 299–306, Washington, DC, USA, 2002. IEEE Computer Society.

[10] R. Maciejewski, A. Pattath, S. Ko, R. Hafen, W. Cleveland, and D. Ebert. Automated box-cox transformations for improved visual encoding. *Visualization and Computer Graphics, IEEE Transactions on*, PP(99):1, 2012.

[11] O. Mattausch, T. Theussl, H. Hauser, and E. Groller. Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. In *Proceedings of the 19th spring conference on Computer graphics*, SCCG '03, pages 213–222, New York, NY, USA, 2003. ACM.

[12] E. Nauman. *Handbook of Industrial Mixing: Science and Practice*. Wiley-Intersciences, 2003.

[13] J. M. Ottino. *The Kinematics of Mixing: Stretching, Chaos and Transport*. Cambridge Texts in Applied Mathematics(No. 3). Cambridge University Press, 1989.

[14] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. Avila, K. Raghu, R. Machiraju, and J. Lee. The transfer function bake-off. *Computer Graphics and Applications, IEEE*, 21(3):16 –22, may/jun 2001.

[15] T. Weinkauf and H. Theisel. Curvature measures of 3d vector fields and their applications. *Journal of WSCG*, 10(2):507–514, 2002.

[16] M. Wijffelaars, R. Vliegen, J. J. van Wijk, and E.-J. van der Linden. Generating Color Palettes using Intuitive Parameters. *Computer Graphics Forum*, 27(3):743–750, undefined.

[17] M. Zockler, D. Stalling, and H.-C. Hege. Interactive visualization of 3d-vector fields using illuminated stream lines. In *Visualization '96. Proceedings.*, pages 107 –113, 27 1996-nov. 1 1996.

Benchmarking of De-noising Techniques for Streaking Artifacts in Industrial 3DXCT Scan Data

Hammad Qureshi,
SEECS, NUST,
H-12, Islamabad,
Pakistan.
hammad.qureshi@seecs.edu.pk

Muddassir Malik,
SEECS, NUST,
H-12, Islamabad,
Pakistan.
muddassir.malik@seecs.edu.pk

Malik Anas Ahmad
SEECS, NUST,
H-12, Islamabad,
Pakistan.
anas.ahmad@seecs.edu.pk

Christoph Heinzl
University of Applied Sciences
Upper Austria
Stelzhamerstraße 23
4600 Wels/Austria
Christoph.Heinzl@fh-wels.at

ABSTRACT

De-noising is one of the most important applications of image processing which has been applied to a wide variety of real world problems. De-noising allows for improving image quality in imaging modalities that are noise prone. A lot of research work has gone in to improving quality of 2D images using various de-noising techniques but new modalities of imaging such as industrial 3D X-ray computed tomography (3DXCT) have received little attention. Industrial 3DXCT scanning is used these days to acquire detailed images of the internal construction of industrial components and machinery so that defects within these can be identified non-destructively and non-intrusively. However, 3DXCT imaging is prone to artifacts and noise. One solution to the problem is to use increased number of projections which results in reduced noise but increased costs. A more cost effective solution is to de-noise the images. In this paper, we show how various de-noising techniques may be used to de-noise 3DXCT scan images acquired using a lower number of projections. We also benchmark these techniques using various picture quality measures. Our investigation shows that good results may be obtained using wavelet shrinkage and anisotropic diffusion.

Keywords
3D X-ray Computed tomography, 3D image de-noising, image enhancement.

1. INTRODUCTION

Industrial 3DXCT scanning is an emerging imaging technique that has revolutionized industrial inspection and quality control. 3DXCT scans are used to generate a highly accurate 3D image of industrial components and machinery, revealing important information about the internal material decomposition, density of the scanned material and internal construction of the scanned object. 3DXCT allows for non-destructive testing of industrial components for discovery of internal defects that are not possible to detect otherwise. Stress testing is usually used to detect such defects which often involves applying force on the object of interest which may lead to impairment of the machinery or industrial component. Using 3DXCT no stresses need to be applied on the object of interest and hence the technique can also be used to analyze elastic or deformable specimens. Moreover, industrial 3DXCT generates a 3D model that can be used to perform nominal comparisons between surface model of the scanned part and the corresponding CAD model, for assembly defect analysis, void analysis and can also be used for generation of CAD data for reverse engineering applications.

3DXCT scan inherently uses X-rays to acquire images of an object. The specimen is rotated and X-ray attenuation images are recorded typically at each angular step of a full rotation. This leads to a collection of X-ray images stored as a image stack. This image stack is subsequently used to reconstruct a 3D volumetric representation corresponding to the specimen being scanned. Although 3DXCT scanning has improved considerably since its invention, the technique still suffers from noise. The quality of the 3DXCT scan is directly dependent on the noise inherent in 3DXCT images, which has a strong relation to the strength of the radiation used to acquire the images referred to as dose. Keeping the radiation low is of importance in the medical domain but in industrial 3DXCT high radiation poses no problems. However, there is a clear trend visible in industrial 3DXCT towards higher resolutions and smaller voxel sizes. Unfortunately due to this trend 3DXCT images tend to show increased levels of noise. A potential solution to the problem is the increased number of projections which results in reduction of noise but increases the cost of the scans greatly. Industrial 3DXCT images suffer from various types of noise, which includes streaking artifacts and beam hardening. Streaking artifacts are introduced due to limited detector dynamics as well as insufficient projection data or local highly changing attenuation coefficients of the scanned materials. Beam hardening is the result of the lower energy X-ray photons getting absorbed by the specimen being scanned. In this paper, we will only focus on removal of streaking artifacts. Figure 1 shows an example of streaking artifacts seen in a 3DXCT scan image.

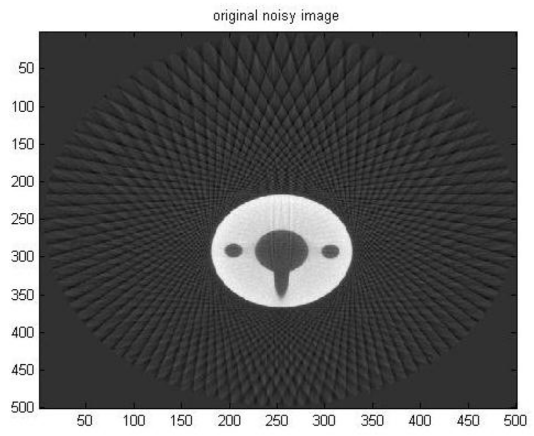


Figure 1. Streaking artifacts observed in an Industrial CT Scan image.

2. RELATED WORK

Although image de-noising has been a field of extensive study for the past many decades but no one technique has been proven to be the best for all types of noise. Different techniques have been shown to provide better results in different applications. Chang et al. [Chang2000] use adaptive wavelet thresholding to de-noise images and compare Bayes shrink with Sure shrink while Kivanc et al. [Kivanc1999] employ adaptive statistical modeling of the wavelet coefficients to perform image de-noising. Other wavelets-based methods reported in literature include the work of Portilla et al. [Portilla2003], Sendur and Selesnick [SendurSelesnick2002] and Rajpoot et al. [Rajpoot2004]. Some other transforms that have found application in de-noising is the curvelet transform [Starck2002] and the fast Fourier transform [Jain1989]. On the other hand some experts have chosen to work with multiwavelets [Bui1998]. Some spatial methods have also found application in de-noising namely non-local algorithms by Buades et al. [Baudes2005] , MDL by Rissanen [Rissanen2000] , median filtering by Yang et al. [Yang1995] and Hamza et al. [Hamza1999] , Wiener filtering by Jain [Jain1989] and bilateral mesh de-noising by Fleishman et al. [Fleishman2003] . Other methods used for de-noising include learning [Elad2006] and fractal dimensions [Ghazel2003]. In a review, Buades et al. contend that the performance of most de-noising methods is highly dependent upon the assumptions taken and the corresponding data-set used [Baudes022005]. Slight variation in the data-set results in lowering of de-noising accuracy. Motwani et al. [Motwani2004] present a survey of various techniques from the spatial as well as the transform domain and conclude that wavelets work best. However, Motwani contend that the noise in images is often assumed to be gaussian which is often not the case in the real world. One good example of such noise is the streaking noise in industrial 3DXCTs. Some work on 3DXCT scan de-noising has already been done which includes Mayer et al.'s [Mayer2007] work that compares non-linear diffusion with wavelets-based methods in de-noising medical XCT. Li et al. have found anisotropic diffusion to work well in reducing noise in industrial 3DXCTs [Li2009].

In this paper, we compare the performance of Otsu method, Fourier transforms, wavelet shrinkage and anisotropic diffusion in removing streaking artifacts from industrial 3DXCT scans. We employ a variety of data-sets and provide de-noising results using various picture quality metrics. Such a comparison for industrial 3DXCT has not been carried out in the literature before. In this paper we compare various de-noising techniques for industrial 3DXCT and present which technique works best. We also present how Otsu method may be used for de-noising. Another important contribution of the paper is the evaluation of the results using multiple quality metrics. Mean squared error used frequently for picture quality estimation has proven to be not very accurate [WangBovik2009] especially in situations where a stable ground truth is not available. Therefore, we choose to compare the results using multiple quality metrics.

3. METHODS AND TECHNIQUES

In this work, we address two important aspects related to image de-noising: The first aspect is comparison of various de-noising techniques and the second aspect is an assessment of the picture quality metrics to evaluate the results. We discuss each aspect in the sections below:

De-noising Methods

We have used various techniques to carry out de-noising. In the subsequent sections, we discuss each technique independently:

3.1.1 Otsu Thresholding

Since the use of Otsu method for barcode de-noising by Shellhammer et al. [Shellhammer1999], not much work has been carried out using the Otsu method for de-noising. We propose to use the Otsu thresholding in a scheme similar to the one proposed by Shellhammer et al. to de-noise a 3DXCT scan image. Since there is a great variation in the Hounsfield units obtained in industrial 3DXCT, adaptive automatic thresholding as proposed by Otsu [Otsu1975] can provide interesting insights as to how to de-noise images. We use the method to acquire a de-noised version of the industrial 3DXCT scan.

3.1.2 Fourier Transform

Fourier transform converts a signal from the time domain in to the frequency domain referred to as the Fourier space. In this paper, we use the Fourier space just as in the curvelet transform to acquire the frequency pattern for streaking artifacts and then remove the artifacts by removing frequencies associated with the noise. Application of 2D Fourier transform for image de-noising is carried out by removing the high frequency components as described by Gonzalez and Woods [Gonzalez2002]. Low pass filtering is applied using a 7 pixel circular filter.

3.1.3 Wavelet Shrinkage

We use the standard wavelet shrinkage method as proposed by Donoho and Johnstone [Donoho1995]. The selection of threshold is carried out using the Birge and Massart method [BirgeMassart1997]. We use various shrinkage functions namely soft, hard and garrote as suggested by Fodor and Kamath [FodorKamath2003]. The results for each are presented.

Daubechies 4-tap filter is used for wavelets analysis. Other wavelets were also tried but Daubechies combined with Birge and Massart method produces the best results. Other threshold calculation methods such as Sure, Bayes and Penalized were also attempted but no improvements in results were seen.

3.1.4 Anisotropic Diffusion

Anisotropic diffusion also referred to as Perona-Malik diffusion [Perona1990] is a powerful technique for removing noise from an image while preserving important information such as edges and other details. It has been known to perform well for medical imaging. The filtering is carried out for 100 iterations.

Picture Quality Metrics

We use various quality metrics for acquiring an estimate of the quality of the picture obtained after de-noising. The ground truth is obtained from scanning the object in question at high resolutions. However the ground truth is not entirely noise free. The performance of the various techniques is measured using six metrics, namely Peak Signal to Noise Ratio (PSNR), Structural Similarity Index Metric (SSIM), Chou and Li Metric (CLM) [Mayache1998], Czenakowski Distance (CZD) and Spectral Magnitude Distortion (SMD) [DosselmannYang2005]. The reason for using multiple metrics is that each represents the de-noising performance of the technique used, in terms of different properties of the acquired image. PSNR and SSIM consider the global differences between the image and the ground truth, while CLM represents the visual quality of the image with the help of JND (Just Noticeable Distortion). On the other hand, CZD and SMD consider global correlation and global spectral analysis respectively.

4. RESULTS AND DISCUSSION

We tested three data-sets to analyze the effect of the various de-noising algorithms. Some images from each data-set are shown in the Figure 2. These image-sets will be referred to as *Kalibrierkorper_LR_0* (simple geometrically shaped mono-material), *KF9_real* (complex geometrically shaped mono-material) and *Testkörper_TK08* (simple geometrically shaped bi-material, metal and plastic).

Completely noiseless data is not available to us, as even with very high number of projections industrial CT-scanned images still contain some noise. In our analysis we took the least noisy image as the ground-truth and evaluated the performance of de-noising methods by comparing the ground truth image with the de-noised one.

Ground truth is obtained using increased number of projections. As stated earlier, one of the major source of streaking artifacts is the insufficient number of projections taken during a CT scan. Industrial 3DXCT machines take multiple x-ray images (projections) of a specimen in a single 360 degree rotation of the machine about the specimen. These projections are then used to compute a 3D volumetric data set for the specimen. The number of projections taken in one 360 degrees rotation about the specimen directly affect the amount of streaking artifacts present in the data set. Less number of projections result in more streaking artifacts due to insufficient sampling rate whereas streaking artifacts reduce as the number of samples (projections) are increased.

The ground truth in our analysis is obtained using a very high number of projections.

In Table 1, 2 and 3, the results of different de-noising techniques on data sets *Kalibrierkorper_LR_0*, *KF9_real* and *Testkörper_TK08* using various quality metrics are shown respectively.

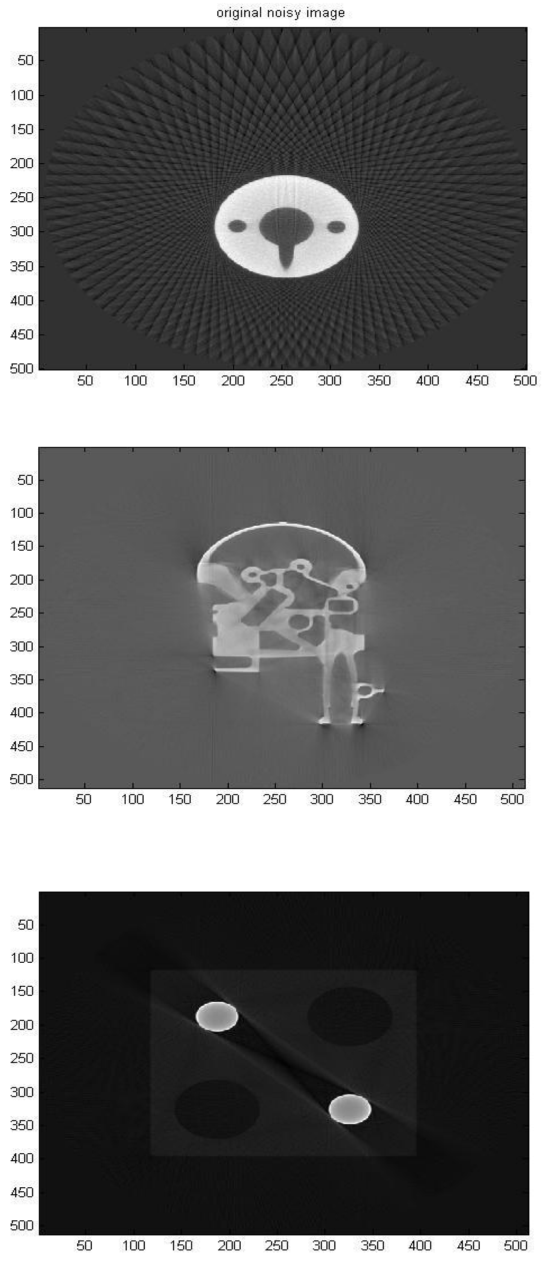


Figure 2. 50th Slice of Kalibrierkorper_LR_0, 300th slice of *KF9_real*, 150th slice of *Testkörper_TK08*

The various quality metrics represent different aspects pertaining the de-noised results. PSNR is a very good metric for estimating the quality of images reconstructed from lossy image CODECs and estimating the quality of transmitted data. It may not be the best metric in this instance due to the nature of the problem as described earlier and also due to the fact that PSNR is closely related to MSE which is not the best metric in problems such as ours. For image and video compression PSNR in the range of 30dB to 50 dB is generally considered good. The PSNR values for the de-noised image in Table 1 and Table 3 indicates that the technique has improved the quality of the image. However, the results for second data-set i.e. *KF9_real* are not very good as they lie between 7 dB to 25.4 dB. As per PSNR Otsu method is consistently the worst method. PSNR values indicate that although some de-noising has been obtained for *KF9_real* but it is not too much improvement. This is also evident from de-noised images in Figures 1-8.

	O	**F**	**Ws**	**Wh**	**Wg**	**A**
PSNR	6.5	27.8	43.9	43.3	43.7	**48.2**
SSIM	0.13	0.96	**0.99**	0.98	0.98	**0.99**
CLM	3555	25.13	0.23	0.30	0.23	**0.07**
CZD	0.875	0.018	0.003	0.004	0.003	**0.002**
SMD	2.5X 10^{14}	1.9X 10^{12}	5.9X 10^{10}	5.3X 10^{10}	**5.2X 10^{10}**	1.7X 10^{16}

Table 1. Metric results for *Kalibrierkorper_LR_0* are shown where 'O' stands for Otsu threshold, 'F' for fourier filtering, 'Ws' for soft wavelet thresholding, 'Wh' for hard wavelet thresholding, 'Wg' for garrotte thresholding and 'A' for anisotropic diffusion

	O	**F**	**Ws**	**Wh**	**Wg**	**A**
PSNR	7.79	23.3	**25.4**	**25.4**	**25.4**	**25.4**
SSIM	0.03	0.94	**0.98**	**0.98**	**0.98**	**0.98**
CLM	2663	69	**41**	**41**	**41**	**41**
CZD	0.97	0.08	**0.07**	**0.07**	**0.07**	**0.07**
SMD	1.9X 10^{14}	5.3X 10^{12}	**3.3X 10^{12}**	**3.3X 10^{12}**	**3.3X 10^{12}**	**3.3X 10^{12}**

Table 2. Metric results for *KF9_real* are shown where 'O' stands for Otsu threshold, 'F' for fourier filtering, 'Ws' for soft wavelet thresholding, 'Wh' for hard wavelet thresholding, 'Wg' for garrotte thresholding and 'A' for anisotropic diffusion

	O	**F**	**Ws**	**Wh**	**Wg**	**A**
PSNR	0.21	**33.2**	32.2	32.2	32.2	32.0
SSIM	0.21	**0.98**	**0.98**	**0.98**	**0.98**	0.97
CLM	359	**4.6**	9.1	9.1	9.1	9.5

CZD	0.81	**0.03**	0.06	0.06	0.06	0.06
SMD	2.6X 10^{13}	**5.9X 10^{11}**	7.4X 10^{11}	7.4X 10^{11}	7.4X 10^{11}	7.7X 10^{11}

Table 3. Metric results for *Testkörper_TK08* are shown where 'O' stands for Otsu threshold, 'F' for Frequency Filtration, 'Ws' for Soft wavelet Thresholding, 'Wh' for hard Wavelet Thresholding, 'Wg' for Garrotte Thresholding, and 'A' for anisotropic diffusion

SSIM or the Structural Similarity Index Metric as the name suggests measures the structural similarity between the noisy and de-noised image. It is considered an improvement over PSNR and MSE as it does not compare the whole image but looks for structural similarity. A value of 1 for SSIM indicates a complete match i.e. the images being compared are identical. Results in Table 1, 2 and 3 indicate that soft thresholding based wavelet shrinkage performs very well for all data-sets producing a SSIM of 0.99, 0.98 and 0.98 followed by anisotropic diffusion. Different thresholding techniques for wavelet shrinkage also produce high SSIM values. Fourier transform has also produced good results but not as good as wavelet shrinkage and anisotropic diffusion. The results indicate that wavelet shrinkage, anisotropic diffusion and Fourier transform maintain the image structure in the de-noising process. However, Otsu is the worst technique as the results indicate that Otsu method has significantly modified the image structure in the de-noised image.

CLM provides an estimate of image quality in terms of distortion accrued in the de-noised image in the process of de-noising in comparison with the ground truth. Lower the distortion the better the results. The values for CLM vary greatly for different data-sets as can be seen in Table 1, 2 and 3. The CLM metric indicates that de-noising works best for the first data-set i.e. *Kalibrierkorper_LR_0*. This fact is interestingly also evident from the results in Figures 1-8. The best value for CLM in the case of *Kalibrierkorper_LR_0* is obtained for anisotropic diffusion which is 0.07 which contrast greatly from 41 and 4.6 for other data-sets. An important aspect to note is that CLM metric also indicates that Fourier transform works best for the last data-set i.e. *Testkörper_TK08.* Very high values for CLM are obtained in the case of Otsu method indicating that the technique does not work well at all for 3DXCT scanned images de-noising.

CZD or Czenakowski Distance measures the percentage of similarity between two images which in this case are the de-noised image and the ground truth. As the difference between two images tends to zero the more similar the images are. The results in Tables 1, 2 and 3 indicate that except for Otsu method all other techniques produce very low values for CZD with anisotropic diffusion producing the best results for *Kalibrierkorper_LR_0***.** However, the difference between de-noising results for *Kalibrierkorper_LR_0* and other data-sets is quite large in terms of CZD as it is more than 10 times as compared to *Kalibrierkorper_LR_0* (with the exception of Otsu method which again produces very poor results). This result also indicates that de-noising works best for data-sets such as *Kalibrierkorper_LR_0* which can also be seen from the images in Figures 1-8.

SMD measures the quality of the de-noised image in the spectral i.e. frequency domain. The lower the difference the better the results. It can be seen that very high values for SMD are obtained for all data sets indicating that there is still a presence of considerable noise. The values for SMD confirm for the two data-sets namely *KF9_real* and *Testkörper_TK08* that wavelet shrinkage and anisotropic diffusion produce similar results and Fourier transform is the best technique for the latter. However, the SMD results for *Kalibrierkorper_LR_0* are most interesting in the sense that it indicates that garrote thresholding based wavelet shrinkage produces the best results which is a deviation from what other metrics indicate. This may be due to the fact that SMD is unable to detect localized errors as the Fourier transform estimates the total energy of the signal.

From the above discussion it can be concluded that best de-noising results may be obtained for data-sets such as *Kalibrierkorper_LR_0*. However, for other data-sets the improvement in quality is only marginal but some techniques tend to keep the structural integrity intact while others such as Otsu compromise it.

The results in Tables 1, 2 and 3, clearly indicate that the Otsu's metric results are not very good for all types of 3DXCT scan images. Figure 3 shows the performance of the Otsu method on 50th slice of *Kalibrierkorper_LR_0*, 300th slice of *KF9_real* and 150th slice of *Testkörper_TK08*. In the *Testkörper_TK08* filtered result, one can clearly see that Otsu removed the non-metal intensity value along with the air considering it to be noise.

Moreover, for mono materials (in Table 1 & 2) the metrics index for Fourier filtering are much better when compared with Otsu. However, blurring is observed as some important edges are removed but in bi-materials case i.e. *Testkörper_TK08* (Table 3), Fourier filtering was found to be the best method based upon most of the metrics (except SSIM). In the case of bi-material based structure, the object is so badly corrupted by streaking noise that according to our metrics, no other de-noising technique improved the quality of the image.

We applied wavelet filtering using Daubechies 2 wavelet up to level 9 while threshold was calculated using Birge and Massart method. It improved our results for the geometrical shaped mono material *(Kalibrierkorper_LR_0)* to a certain level but improvement for complex geometrically shaped mono-materials (*KF9_real*) was very low and it proved almost useless for simple geometrically shaped bi-material (*Testkörper_TK08*). Among wavelets we found soft thresholding results to be the best for the first data set. For the other two data-sets, the different wavelet thresholding methods produced identical results. Anisotropic diffusion provides the best results for the first two image data sets. Therefore, anisotropic diffusion is the best method for de-noising of mono material 3DXCT scanned images. But it did not work well for *Testkörper_TK08*.

The reason why anisotropic diffusion and wavelet shrinkage work better than other techniques for removing streaking artifacts is probably because both techniques are multi-resolution in nature and allow for removal of noise artifacts at different scale-space resolutions. This is particularly important in the case of streaking artifacts as they represent noise at multiple frequencies which is captured well using various scale-space resolutions. Hence, once the noise is de-correlated at various scale-space resolutions, the de-noising is more accurate as the noise is removed separately at each scale space. At the same time, another additional feature of anisotropic diffusion is that it adapts to the local noise variations by suppressing the noise and enhancing the texture. Therefore, anisotropic diffusion is able to surpass wavelet shrinkage in removing streaking artifacts.

Moreover, we have also compared the methods in terms of the time they took to perform the de-noising. Table 4 shows the processing time for each technique. Otsu is the fastest but as discussed before does produce very good results. Wavelet shrinkage offers the best compromise with acceptable results and very quick processing time of 40 seconds. Fourier filtering takes 5 minutes but Anisotropic Diffusion takes the most amount time of more than an hour although it produces the best results in terms of quality. Speed of a technique is important as a typical 3DXCT scan contains hundreds and thousands of images.

	Processing Time
Anisotropic Diffusion	1 hour 2 sec
Fourier Filtering	5 min
Wavelets	40 sec
Otsu	28.6 sec

Table 4. Processing time for various techniques for de-noising 256 slices of a 512X512 XCT Scan

5. CONCLUSIONS

This paper compares established techniques in de-noising literature for removing noise in industrial 3DXCT scan data. 3 different data-sets are used and the efficacy of the technique on these 3 data-sets is investigated and compared using different picture quality metrics.

The analysis shows that anisotropic diffusion produces the best results while wavelet shrinkage provides the second best results. Often the wavelet shrinkage results are as good as anisotropic diffusion. Frequency space based techniques such as the Fourier transform works well in situations where the original image has been greatly compromised by streaking artifacts otherwise anisotropic diffusion and wavelet shrinkage are better techniques. Otsu thresholding has been found to be not a very good technique for industrial 3DXCT de-noising. However, it is important to note that the improvement in quality is only marginal for most data-sets included in the study. Great improvement in quality can be seen for the first data-set. In terms of speed and accuracy, wavelet shrinkage provides the best compromise.

Our future work would include comparing other de-noising techniques with anisotropic diffusion and wavelet shrinkage based methods and also to come up with new techniques that can produce good and robust de-noising results.

6. ACKNOWLEDGMENTS

Our thanks to Higher Education Commission Pakistan for funding this research work.

7. REFERENCES

[Chang2000] Chang, S.G., Yu, B. and Vetterli, M. Adaptive wavelet thresholding for image de-noising and compression. IEEE Transactions on Image Processing, , pp. 1532–1546, (2000).

[Kivanc1999] Kivanc, M., Kozintsev, I. Ramchandran, K. and Moulin, P. Low-complexity image de-noising based on statistical modeling of wavelet coefficients. IEEE Signal Processing Letters, Vol. 6, pp. 300–303, (1999).

[Portilla2003] Portilla, J., Strela, V., Wainwright, M.J. and Simoncelli, E.P. Image de-noising using scale mixtures of Gaussians in the wavelet domain. IEEE Transactions on Image Processing, Vol. 12 pp. 1338-1351, 2003.

[SendurSelesnick2002] Sendur, L. and Selesnick, I.W., Bivariate shrinkage functions for wavelet-based de-noising exploiting inter-scale dependency. IEEE Transactions on Signal Processing, vol. 50, no. 11, pp.2744-2756, 2002.

[Rajpoot2004] Rajpoot, N., Yao, Z. and Wilson, R., Adaptive wavelet restoration of noisy video sequences. International Conference on Image Processing, 2004. ICIP'04. 2004 vol. 2, pp.957-960, 2004.

[Starck2002] Starck, J.L., Candes, E.J., Donoho, D.L. The curvelet transform for image de-noising. IEEE Transactions on Image Processing, vol. 11, pp.670-684, 2002.

[Jain1989] Jain, A.K. Fundamentals of digital image processing. Prentice-Hall, Inc. 1989.

[Bui1998] Bui, T.D. and Chen, G. Translation-invariant de-noising using multiwavelets. IEEE Transactions on Signal Processing, vol. 46, pp.3414-3420, 1998.

[Baudes2005] Buades, A., Coll, B. and Morel, J.M. A non-local algorithm for image de-noising. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005. vol. 2, pp.60-65, 2005.

[Rissanen2000] Rissanen, J. MDL de-noising. IEEE Transactions on Information Theory, vol. 46, pp. 2537-2543, 2000.

[Yang1995] Yang, R., Yin, L., Gabbouj, M., Astola, J. and Neuvo, Y. Optimal weighted median filtering under structural constraints. IEEE Transactions on Signal Processing, vol. 43, no. 3, pp.591-604, 1995.

[Hamza1999] Hamza, A.B., Luque-Escamilla, P.L., Martinez-Aroza, J. and Roman-Roldan,R. Removing noise and preserving details with relaxed median filters. Journal of Mathematical Imaging and Vision, vol.11, pp.161-177, 1999, Springer.

[Fleishman2003] Fleishman, S., Drori, I. and Cohen-Or, D. Bilateral mesh de-noising. ACM Transactions on Graphics, vol.22, pp.950-953, 2003.

[Elad2006] Elad, M. and Aharon, M. Image de-noising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image Processing, vol.15, pp.3736-3745, 2006.

[Ghazel2003] Ghazel, M., Freeman, G.H. and Vrscay, E.R. Fractal image de-noising. IEEE Transactions on Image Processing, vol. 12, pp.1560-1578, 2003.

[Baudes022005] Buades, A., Coll, B. and Morel, J.M. A review of image de-noising algorithms, with a new one. SIAM Journal on Multiscale Modeling and Simulation, vol. 4, pp.490-530, 2005.

[Motwani2004] Motwani, M.C., Gadiya, M.C., Motwani, R.C. and Harris Jr, F.C. Survey of image de-noising techniques. GSPx, pp.27-30, 2004.

[Mayer2007] Mayer, M., Borsdorf, A., Köstler, H., Hornegger, J. and Rüde, U. Nonlinear Diffusion vs. Wavelet Based Noise Reduction in CT Using Correlation Analysis. Vision, Modeling, and Visualization, pp.223-232, 2007.

[Li2009] Li, J., Wang, L. and Bao, P. An industrial CT image adaptive filtering method based on anisotropic diffusion. International Conference on Mechatronics and Automation, 2009. pp.1009-1014, 2009.

[WangBovik2009] Wang, Z. and Bovik, A.C. Mean squared error: Love it or leave it? A new look at signal fidelity measures. Signal Processing Magazine, IEEE, pp.98-117, 2009.

[Shellhammer1999] Shellhammer, S.J. and Goren, D.P. and Pavlidis, T. Novel signal-processing techniques in barcode scanning. IEEE Robotics & Automation Magazine, vol. 6, pp.57-65, 1999.

[Otsu1975] Otsu, N. A threshold selection method from gray-level histograms. Automatica, vol. 11, pp.285-296, 1975.

[Gonzalez2002] Gonzalez, R.C. and Woods, R.E. Digital image processing. Prentice Hall, 2002.

[Donoho1995] Donoho, D.L., Johnstone, I.M., Kerkyacharian, G. and Picard, D. Wavelet shrinkage: asymptopia? Journal of the Royal Statistical Society, Series B (Methodological), pp.301-369, 1995.

[BirgeMassart1997] Birgé, L., Massart P. From model selection to adaptive estimation. D. Pollard (ed), Festchrift for L. Le Cam, Springer, pp. 55–88, (1997).

[FodorKamath2003] Fodor, I.K. and Kamath, C. De-noising through wavelet shrinkage: an empirical study. Journal of Electronic Imaging, volume 12, pp.151, 2003.

[Perona1990] Perona, P. and Malik, J. Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, pp.629-639, 1990.

[Mayache1998] Mayache, A., Eude, T. and Cherifi, H. A comparison of image quality models and metrics based on human visual sensitivity. International Conference on Image Processing, pp.409-413, 1998.

[DosselmannYang2005] Dosselmann, R. and Yang, X.D. Existing and emerging image quality metrics. Canadian Conference on Electrical and Computer Engineering, pp.1906-1913, 2005.

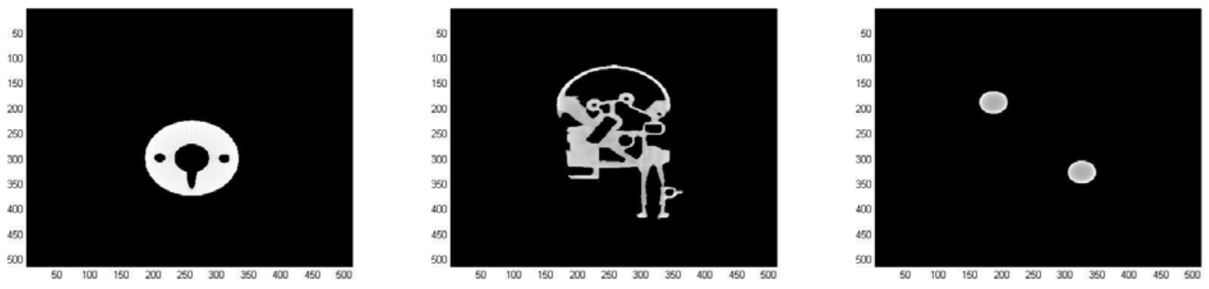


Figure 3. Denoising results for Otsu Method.

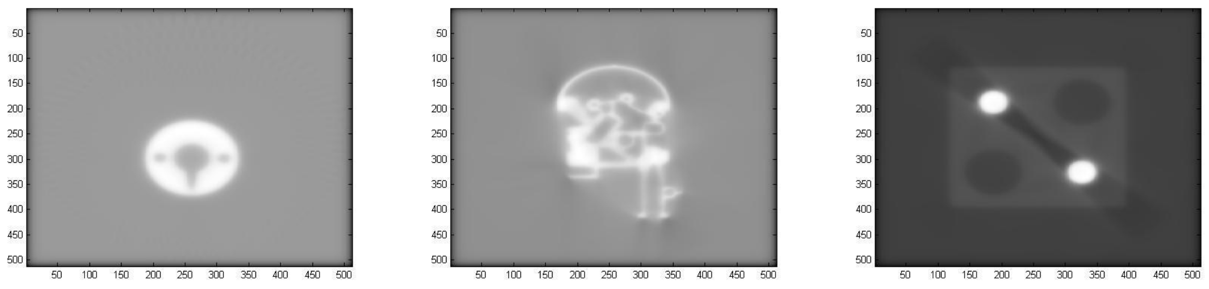


Figure 4. Denoising results for Fourier filtering.

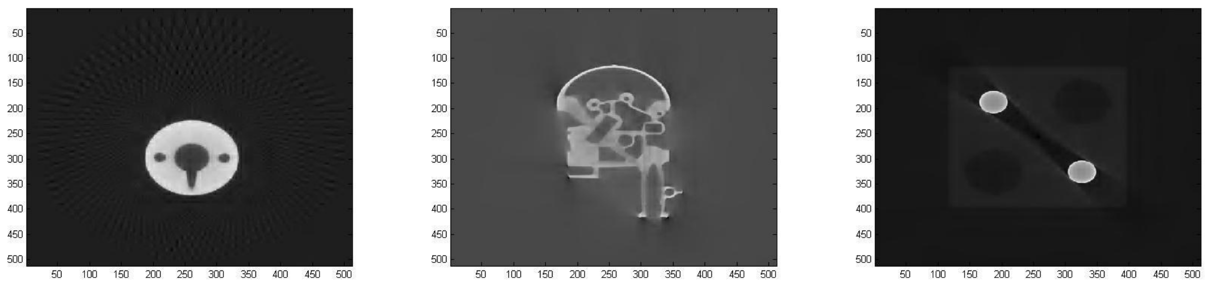


Figure 5. Denoising results for Wavelets Soft Thresholding (Daubechies filter+Birge-Massart threshold calculation).

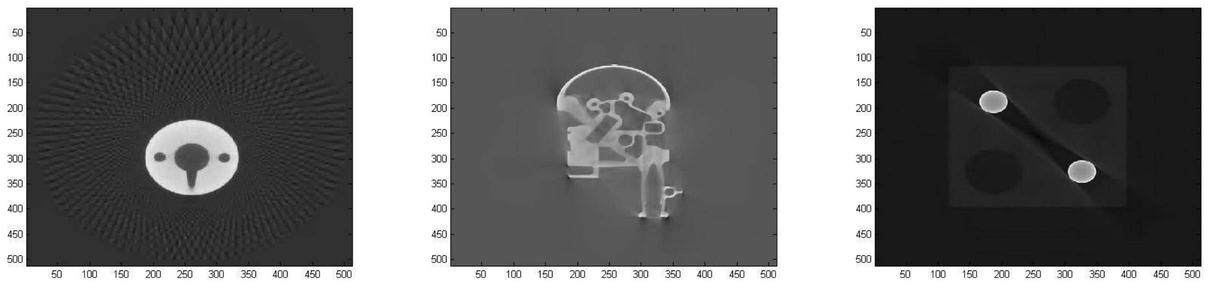


Figure 6. Denoising results for Wavelets Hard Thresholding (Daubechies filter+Birge-Massart threshold calculation).

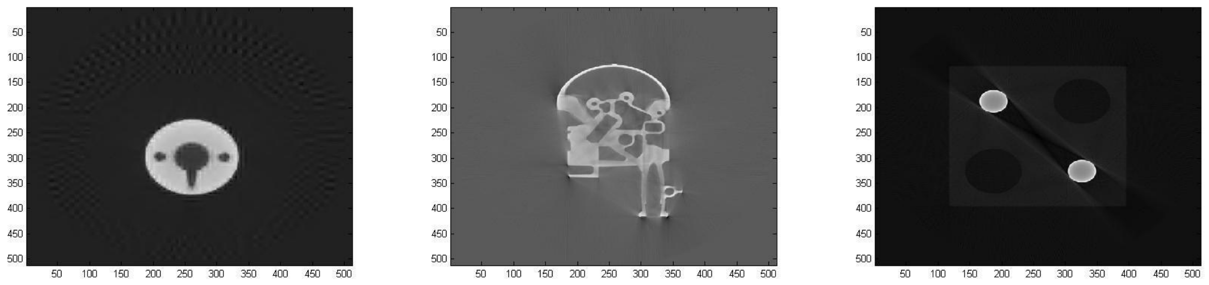


Figure 7. Denoising results for Wavelets Garrotte Thresholding (Daubechies filter+Birge-Massart threshold calculation).

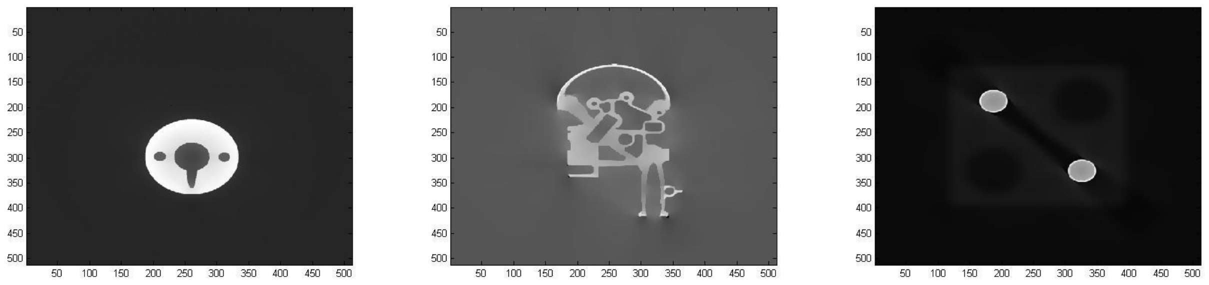


Figure 8. Denoising results for Anisotropic Diffusion.

WorldPlus: *An Augmented Reality Application with Georeferenced content for smartphones - the Android example.*

Sérgio Graça[1] João Fradinho Oliveira[1] Valentim Realinho[1]

sergiogr@gmail.com jfoliveira@estgp.pt valentim.realinho@estgp.pt

[1] C3i Centro Interdisciplinar de Investigação e Inovação/Instituto Politécnico de Portalegre

Lugar da Abadessa, Apartado 148, 7301-901 Portalegre

ABSTRACT

In the last few years there has been a significant evolution in the mobile devices hardware capabilities, this evolution is very important as it allows for more complex applications and services to be developed for this type of devices. In this paper we describe the concepts and key issues that arise when developing an Augmented Reality system in a localization application for smartphones in general. WorldPlus presents solutions and implements these concepts using the Android smartphone as an example. In addition, WorldPlus allows programmers to develop their own content providers for both an online and an offline mode.

Keywords

Augmented Reality, Smart phone, Android

1. INTRODUCTION

In the last few years there has been a significant evolution in the mobile devices hardware capabilities, this evolution is very important as it allows for more complex applications and services to be developed for this type of devices.

Platforms like IOS (Mobile Operating System from Apple) and Android (Mobile Operating System developed by the Open Handset Alliance with Google Inc. leadership) are examples of platforms that were developed for mobile platforms; they include a set of tools that enable one to take advantage of the hardware capabilities of the new mobile devices.

Whilst mobile localization Augmented Reality applications are not new, in this paper we describe the concepts and key issues that arise when developing an Augmented Reality system in a localization application for smartphones in general, and offer solutions to some of the problems. We present available tools that can be used in ordinary smartphones and implement an augmented reality system using the Android system as an example, where the user can browse the world through the mobile device and find additional information about his surroundings, with these tools. In this paper we review the main concepts of an Augmented Reality based on Localization in Section 2, including application requirements and the description of two sources of online content or points of interest POI that are viewed with our system (Panoramio/photos, Wikipedia/documents). In Section 3 we present the WordPlus application, which describes how different components and services from the OS of mobile devices can be used to gather geo referenced points of interest, and how these points are managed and projected in our system. We show results in Section 4, propose further solutions in Section 5, and conclude in Section 6.

2. BACKGROUND

In this section we review key concepts that need to be modeled in order to build our augmented reality system based on localization, we outline the application requirements, and review two online sources of georeferenced information that will be accessed and viewed with our system.

2.1 Augmented Reality Based on Localization

Augmented Reality is a concept contained in a larger concept that is Mixed Realities, Mixed Reality is a concept that represents a system where various realities are mixed together, usually mixing the reality that we all know, the world where we live, and realities generated by computational processes To facilitate the comprehension of these concepts, and relations between them, Paul Milgram has created a spectrum that clarifies it called Reality-Virtuality Continuum [Mil94a] (see Fig. 1), the spectrum goes from Reality (left) that comprises the world where we live in, with no extra elements computationally generated, to Virtual Reality (right) where the "world" is completely computer generated and doesn't have any elements from the real world. In the middle of these two is the Mixed Reality that consists in Augmented Reality and Augmented Virtuality, these two AR and AV are opposite concepts, while AR is the real world augmented with computer generated elements, AV is a virtual world augmented with real world components.

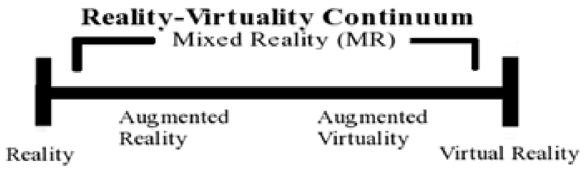


Figure 1: Reality-Virtuality Continuum.

Now that these concepts have been introduced, Augmented Reality based on localization will be described in more depth. Augmented Reality is the mixture between the real world and computer generated elements, this concept has the following elements:

- Caption of real world;
- Computer generated elements;
- In a 3D environment.

AR Reference Model

The cost and effort in developing mobile AR systems is quite high. To reduce these, a number of different software architectures and toolkits have been proposed. A survey of different mobile and wireless technologies and how they have impact AR, including software architectures for mobile AR can be found in Papagiannakis et al. [Pap08a]. The authors compare several models and conclude that there is no single ideal mobile AR system approach but rather different AR systems according to location (indoors or outdoors), type of display (handheld or head mounted displays), content augmentation (static 3D, virtual characters) as dictated by each application domain. Reicher [Rei04a] presents a reference architecture for augmented reality systems organized into logical subsystems as follows: (i) a *tracking* subsystem that in smartphones is typically based on location sensors in the device such as GPS, compass and accelerometer allowing 6 degrees of freedom in displaying a digital object; (ii) an *application* subsystem responsible for the main control flow logic of the application and coordinating communication between other subsystems; (iii) a *world model* subsystem that stores and provides access to a digital representation of the world, including points of interest, 3D objects and metadata about the model itself; (iv) a *presentation* subsystem responsible for all output, including reality view streams, 2D and 3D rendering and audio and tactile outputs; (v) an *interaction* subsystem which gathers and processes any input that the user makes deliberately; and (vi) a *context* subsystem that provides the application with context about the status and situation of the user. We use an architecture that can be mapped to the one presented by Reicher.

AR Browsers for Smartphones

Layar [Lay12a] is the most prominent AR browser designed for smartphone devices. It offers animated 3D rendering, location based tracking, and has a highly flexible API and a useful set of tools for developers. Wikitude Worlds [Wik12a] is a general purpose AR browser with location based tracking, support for 2D images and it is one of the most easy to use browsers for developers. The Wikitude Worlds AR browser is based on the Wikitude API, which is an open source framework for developing users own standalone AR applications on iPhone, Android and some Symbian based devices. Junaio [Jun12a] is a powerful AR browser which includes features to support 3D object rendering, location based tracking, marker and markerless image recognition and a powerful API for developers. Sekai Camera [Sek12a] is another example of an AR browser which is a social augmented reality mobile location-based service. Wikitude is the only one that supports an offline mode which is an advantage where the availability of an internet connection is an issue. Our approach was to support both online and offline mode. We provide an architecture that enables programmers to develop their own content providers for both online and offline modes. For evaluation purposes, we provide in the current version of WorldPlus an offline access of the geo-referenced Wikipedia articles, and an online access of the geo-referenced Panoramio photos.

2.2 Application Requirements

In order to implement an Augmented Reality Localization application on a smartphone, we identified the following five requirements:

1. ***Caption of real world*** – this requirement will be resolved with the camera that comes with the mobile device, this camera will help on capturing a stream of real world images in real time like the AR concept requires.
2. ***Gather Points of Interest*** – these points are places in the world that are represented by a latitude and a longitude, these elements will represent the real world places that the application detects when the user points the camera at them, they will consist basically on a bitmap image/tag for each point of interest with which the user can interact and retrieve additional information about the place represented. These graphic elements will be superimposed in the view of the camera, like a layer on top of the captured real world images of the camera in which the images representing the points are drawn.

3. ***Process Sensors Data*** – modern mobile devices have a set of sensors that can track the position and direction of the device, the accelerometer sensor and magnetic sensor respectively.

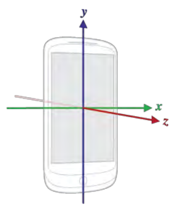


Figure 2: Three axis system in a device.

The accelerometer´s main objective is to map the device's position (axis) in relation to the real world, it measures the force in G's applied in the 3 axis (Figure 2) and provides values that help one determine for example if the device is leaning forwards facing the ground or leaning backwards and facing the sky. In addition, these values can represent if the device is in a portrait position or in a landscape position, enabling an application to decide to perform some processing based on this information, perhaps for flipping the user interface elements so that the user can view them always in a correct position. Furthermore these values will later be used in the calculation of the projection of the points of interest in the device.

The magnetic sensor helps to determine what direction the device is facing, much like a compass in reading which direction is north. This sensor is important to determinate if a given point of interest is in the same direction that the device is facing, in this case the point has a high chance of being drawn on the device's display. The magnetic sensor provides the values in radians that can be converted to degrees, it measures from 1 to 360 degrees. 0/360 degrees being the direction of north.

These two sensors are processed at the same time using methods provided by the Android SDK library, and it returns 3 values:

- **Azimuth** – this value corresponds to the angle between the true north and the direction that the device is facing.
- **Pitch** – this value corresponds to the angle that represents the inclination of the device, if the device is leaning forward or backward.
- **Roll** – this value represents the rotation in the device that helps one determine for example, if the device is in a portrait position or in landscape.

4. ***Localization System*** - the application has to know where the user/device is located on the surface of the Earth, it needs to know this so that amongst other tasks it can detect geo referenced points surrounding the user or get distances from the users to points with a latitude and a longitude. This can easily be achieved by a the GPS system that most devices have already built in, where the system using a method of triangulation with satellites can locate the position of the user on the surface of the Earth. In the particular case of the android platform this localization system uses a mixture of the GPS and network systems.

5. ***Network*** – to gather points of interest, our system will use online services such as Panoramio[1] that need an Internet connection for retrieving information. And as mentioned in the "Localization System" requirement, in the particular case of the Android platform, network protocols help to geographical locate the device in a combination of broadband systems and Wi-fi access points[2], this could be useful, for example, when the GPS service is not available in certain conditions like inside buildings. We point out however that in our experience these alternative localization systems have a much higher error than GPS, serious limiting their use.

2.3 Online Content (Points of Interest)

Our system accesses content / points of interest from two online services: Panoramio for photographs and Wikipedia for documents, both systems hold information that is geo-referenced.

2.3.1 Panoramio

Panoramio is an online service that permits people to submit pictures that are geo referenced by a latitude and a longitude, the pictures are submitted with some additional information like the author of the picture, the title given to the picture by the author, and so on, people then can share their photos with all the users of Panoramio. The objective of using this service in our application is to obtain the pictures taken in the surroundings of the location of the user and show these places to the user with all the information attached to that particular place. When the user turns on the application, it makes a call to the Panoramio API that consists on a REST[3] call for obtaining the points near the user of the application. All the points and their information are obtained formatted in a JSON object that will be parsed in order to retrieve and store in memory all the data that is needed. This type of service is very simple and easy to process with a simple HTTP Request with a query string

[1] Panoramio – Panoramio is an online website that permits the users to submit geo referenced photographs to the system, add additional information about the place where the photograph was taken among other functionalities.

[2] More information about this method of localization can be accessed in the Android SDK Reference Guide in: http://developer.android.com/reference/android/location/LocationManager.htm

[3] REST – Is a method to obtain information from a web server with a simple HTTP Request, much like RPC calls or SOAP but without their formalisms.

containing all the parameters to filter the data. Services using the same method are spreading because of its simplicity. A call at the Panoramio's API is much like the next example.

```
"http://www.panoramio.com/wapi/template/l
ist.html?tag=mountain&width=280&height=140&r
ows=2&columns=4&orientation=horizontal"
```

2.3.2 Wikipedia

As is common knowledge, the Wikipedia is a website that strives to store articles from all branches of human knowledge, some of these articles are about places for instance historical places, Museums, and much more. Articles regarding places often contain references of latitude and longitude, these articles become points of interest for our application. These places will show on the device's display as images (a tag) with which the user can interact. The Wikipedia is a good source of points of interest because of the shear coverage of articles that it stores. Unfortunately Wikipedia was not designed for fast retrieval of geo-referenced data like Panoramio. In order for information on Wikipedia to be used as a source of points of interest, one needs to download "dumps" that the Wikipedia shares online in various forms, in XML files that store all the articles in a given time stamp, or for example SQL dumps. This way of sharing all that information is not very mobile phone friendly as those dumps are released almost every day and consist of files of a few gigabytes. We filter offline only the georeferenced points of interest of such dump files and make the database file available physically in the mobile for offline querying of GPS coordinates.

3. WORLDPLUS

WordPlus was implemented using one of the platforms that is growing rapidly in users, the Android platform, however we believe that with the concepts described in this paper it should be easy to implement an augmented reality application on other similar platforms like the IOS. In this section we will first present the architecture of the application and describe some of the key classes created for the application along with other classes that were reused from the Android SDK libraries. We will then describe in detail how we create the points of interest and tackle problems such as the size of the data. We then describe how the sensor data is processed, noise removed and finally describe the projection method that enables points to be mapped on to the display.

3.1 Application Architecture

Figure 3 illustrates the main classes of our system and how they interact between themselves. There are two central classes that have the most relations with other classes, the *WorldPlus* and *PoiViewManager* classes, these classes are the foundations of the application. The *WorldPlus* class is the base class of the application, it is the "main" class, responsible for instantiating the necessary classes when they are needed and manipulating them as needed in the cycle of the application by using their methods.

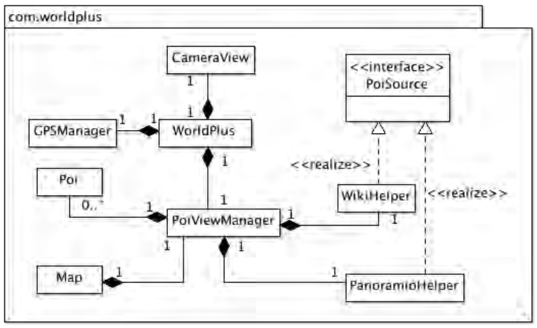


Figure 3: Class diagram of the application.

This class has the following set of tasks:

- Create an instance of the CameraView, the CameraView is itself responsible for displaying the view of the device's camera, real images in real time that are being captured by the camera.
- Create an instance of the GPSManager to get the user´s location at any time.
- Create an instance of the PoiViewManager that is the other central class, and is responsible for managing all that has to do with the points of interest.
- Create an instance of a Sensor Listener that receives notifications from the SensorManager every time that alterations in the state on the sensors of the device are detected. The sensor listener and sensor manager are classes used from the Android SDK libraries.
- Process the data obtained from the sensors, this includes the task of eliminating the noise created by the sensors. Values undergo an averaging procedure (described in section 3.3) before being passed to the method that deals with the calculations of the projection, in this way the projection of points is more stable and prevents points from constantly jumping between positions on the screen due the instability of the sensors.
- "Order" the PoiViewManager to calculate the projection of the points when all the data needed from the sensors is gathered, this includes data from the accelerometer, and from the magnetic sensor more commonly referred as the compass.

The next class discussed will be the PoiViewManager, this class as the name describes, is responsible for managing all the process of dealing with the information contained in the points of interest (Poi – the class that represents a point of interest, this class consist on a set of attributes and interface methods to access them), from the process of loading the information in memory until the points are presented to the user, drawn on the device's display. This class has to accomplish these tasks:

- Load all the information of the points of interest, this consists on gathering the poi's information from the poi source mechanisms that were discussed earlier, like Panoramio or Wikipedia. For this purpose an interface class was created that helps one load ubiquitously different types of poi. Every type of poi source implemented had to implement the methods necessary to load their associated information. In this way, PoiViewManager does not need to know the manner in which the points are loaded, all it knows is that it has to call the same function and the points are loaded in their particular way. This helps in the scalability of the application, with this interface it is easy to implement other forms of gathering points of interest with different processes, and these processes are then transparent to the application.
- Maintain in memory the information loaded.
- Update that information when it is required, for example, when the user walks to another location it is necessary to update the points of interest in memory with the ones gathered in the new location and the previous data become outdated.
- Calculate the positions on screen of the tags that represent the real world places of the points of interest, the projection is described in section 3.4.

The classes *PanoramioHelper* and *WikiHelper*, are the implementations of the sources of poi, these classes implement the methods that are necessary to load the information of the points of interest from their systems, in this case the Panoramio and Wikipedia sites. Both classes implement the interface *PoiSource* that consist of a set of methods to load points information, their implementations are different, as the way in which they load the points is different as described in the previous section. The *GPSManager* class deals with the localization of the user, the geographical location of the device. This class has the task to manage the location providers and to update the location in the application. This class extends the class LocationManager provided by the Android SDK, that has all the necessary methods to manage the localization of the device.

3.2 Gathering Points of interest

The process of developing an application of this kind relies on the use of several services and systems, like the ones listed section 2, that have their limitations, and as a consequence limit the application itself. It is up to the developer to overcome these limitations with a design solution that renders itself still useful. In this section we list the most relevant problems when dealing with online data for a mobile device, along with our design solutions that were implemented. The key problems were: Size of Data; Age of Data; Dynamic Data;

The ***Size of Data*** in any mobile application is crucial, and in the context of this application it has a great impact on it's design. First of all, the data, in the context of WorldPlus, is composed by all the information associated with the points of interest: latitude and longitude, a title that represents the place, images of the place, a description such as historical facts of that place, along with many other characteristics that can be associated with a point of interest and could be important information for the user experience. The objective of the application is to gather points of interest that are in the surrounding environment where the user is located, as we don't know where a user will use the application, it is important to have a large number of points of interest located all around the world so that people from all places in the world can experience the application.

One of the sources of points of interest are the Wikipedia articles as mentioned previously, for a few years now, wikipedia has started to geographically reference some of their articles. One problem arises: how can we process all this data?, Wikipedia has a very large number of articles which means dealing with gigabytes of information in the application, something which is impossible to process with the mobile devices of nowadays, where the space of an application of this kind normally amounts to only a few megabytes. So the challenge is not to store the data in the device, but to create a way in which the application can retrieve this data, filtered in some way, so that it could be processed by these devices and the information presented to the user with a nice fluid experience. Our main interest was on the creation of the AR system itself, we did not solve the problem of transferring and parsing vasts amounts of data from wikipedia, instead we stored in the application parsed content from a few points of interest in order to test basic rendering functionality. In the case of the Panoramio API, we do have direct access to all the available online content. One needs to make the request to the API and process the response, this revealed itself as the most time expensive process in the execution of the application. This process is composed by the request, parsing the data of the response and transfer of the image files which are typically photos of the respective locations and can have arbitrary resolutions.

Age of Data. It is crucial to have data to present in the application, the basic objective of the application, but it is equally important that the information is correct, that it represents exactly the state of the point of interest in the moment that the user requests it's information. For example a file showing train delays. It is important to have a system for point source that is easy to update so the user always sees correct information, and not useless outdated data, the lack of a system that support this issue, could cause failure of some applications.

Dynamic Data In a tourism hotspot, for example Rome, it would be useful to have a system that can show the photos just taken by others in key places of the city, so other people can visit those places too. A person could be submitting a photo and another person nearby could see that photo through the application, for instance to help navigation. But with this dynamic aspect arise some problems that consist on the amount of photos in a given area, e.g. if we were to define a 1 km radius around the location of the user and retrieve all the photos taken inside that radius, the amount of photos retrieved in a place like Rome could still be prohibitive interaction wise. A simpler solution that worked well is to limit the number of points that are computed and drawn with a fixed number, some tests have been done to determine empirically the limit of points of interest that can be computed interactively in the application, these tests will be described in the results section. But another issue arises, how can we choose which points of interest to discard and which ones are the best to present to the user. This can be achieved by determining the distance at which the point of interest is from the position of the user, and then discard the ones that are more far, compute and present just those near the user´s location. This is a straightforward solution to overcome the problem of having to compute too many points of interest but there are more elegant solutions that could be implemented in the future. For example implementing machine-learned ranking[4] techniques that can determine the most relevant points, as used in Information Retrieval[5] systems, such as search engines, etc. Points of interest are ranked by an algorithm that computes a group of special characteristics of the points and determines their relevance to the user, this relevance then determines which points are more interesting/relevant to be presented. The application could profile the user by saving user interests, and then the application could assign different scores to poi categories based on those interests, some categories would have more relevance than others or one could create a way of scoring based on the distance the poi is from the user.

3.3 Sensors

Another problem encountered when developing this application was the noise that the sensors report in their readings. The accelerometer sensor of the devices returns values of the acceleration that are imposed in the device to realize the position of the device in relation to the real world, as stated in the previous section. These values are returned in time intervals of milliseconds, and sometimes they return values that are not consistent with each other, if we put a device in a rest position on top of a table for example, we can see that the sensors don't give always the same value, the values oscillate around the real value. Our solution consists on simply saving the values returned by the sensor in a temporary collection of data and before the projection of the points, an average of the values is calculated. Three read outs are taken and averaged in under 100 ms. With processing units with multiple cores one envisages that Kalman filtering in a dedicated core could be used to further improve results.

3.4 Point Projection

The point projection is the process that maps the real places on the display of the device, using tags that consist of a bitmap that represent the points. Our calculations are based on the work of Matuscheck [Mat11a]. The projection is divided in two different calculations, the horizontal projection and the vertical projection, the first consists on calculating the screen coordinate in the *X* axis (horizontal) and the second consists on calculating the Y screen coordinate. To draw a point in the device's display it is necessary to have an *X* and an *Y* coordinate *P(x,y)*. The horizontal projection is based on two similar but different values, the azimuth that we mentioned earlier, that the magnetic sensor provides, and the bearing, that is the angle formed between the true north and a given point on Earth that the device is facing. Before describing these calculations it is important to define the *angle of view*, the angle of view in this application is defined by a 45 degree angle, in the horizontal and in the vertical projections. In the case of the horizontal projection this angle will define two boundaries, like two "arms" that embrace the field of view of the application and determine what is inside of the field of view and shown to the user. Figure 4 illustrates these concepts. As can be seen the azimuth is the angle formed between North and a point that the device, in this case the little man is facing. The bearing is calculated in the same way but it requires an additional point, but if we assume that the point that the little man is facing is a point of interest we can say that the bearing to that point has the same value as the azimuth. In the projection if the azimuth and bearing to a point is the same or very close, that point is a good candidate to be drawn on the display, it means that the device is facing in the direction of that point. After knowing the two angles needed we can define a horizontal screen coordinate from them, we know that the azimuth is always in the middle of the screen, if the bearing has the same value as the

[4] Machine-Learning Ranking - is a type of supervised or semi-supervised machine learning technique where the objective is to create a ranking model with gathered data.

[5] Information Retrieval – area of study with methodologies for searching documents, information inside documents, etc. In these methodologies there are concepts of scoring special characteristics of the documents that determine their relevance.

azimuth the position of the point will be in the middle of the display.

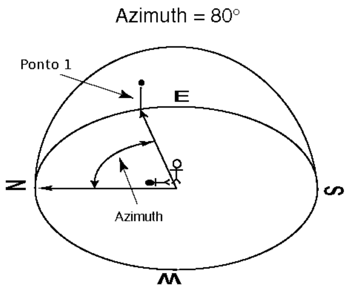


Figure 4: Azimuth calculation (adapted [Phy12a]).

In the case that the bearing to a point is greater than the azimuth then we know that the point is in the right half of the screen, or else, if the bearing is less than the azimuth we know that the point is in the left half of the screen. With this we have a horizontal screen coordinate. Now that we have the horizontal screen coordinate for the point, it is necessary to calculate the vertical coordinate for it, the vertical projection, that consists in calculating the *Y* coordinate that the point will have on the device's display. Like in the horizontal projection, the vertical angle of view is defined with an angle of 45 degrees.

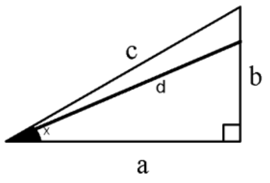


Figure 5: Altitude culling calculation.

Figure 5 shows the triangle *(a,b,c)* that represents half of the vertical field of view. As mentioned before the full vertical angle of view is 45 degrees, the full vertical field of view would have a similar triangle mirrored in the bottom of the one presented.

It is important to define some variables before we calculate the vertical projection, first of all the position of the user and it's device, the user is located in the vertex formed by *ca* facing to *b*, *c* is the upper boundary, *a* is the line that connects the position of the user with the position of the point of interest, *a* represents the distance between those two points. The *b* line represents the height of the point in relation to the height of the position of the user. For example, if the user is at 500 meters high and the point has an altitude of 630 meters *b* will be the difference between them, 130 meters. Representing the position of a point is defined by a line *d* that also represent the angle formed between the position of the user and the position of the point of interest. This is the angle that we need to determine if the point of interest is in the field of view or not, if it will be drawn in the device's display or not. We calculate this angle using the Law of Cosines, this law trivially determines a set of rules that can be used to calculate angles and lengths of the sides of a triangle that has a 90 degree angle.

We have the distance between the two points (*a*), we have the height of the point in relation to the height of the user position (*b*), we know that the angle formed between *a* and *b* is a 90 degree angle, what is missing is the angle formed between the user position and the point of interest position. This is what we are going to calculate next using a rule of the Law of Cosines. The Law of Cosines is an generalization of the Pythagorean Theorem, the base is the next formula.

$$\boldsymbol{d}\hat{}2 = \boldsymbol{a}\hat{}2 + \boldsymbol{b}\hat{}2$$

Here ***d*** is our hypotenuse (squared) and is equal to the sum of the squares of the two sides (***a*** and ***b***). We can now obtain the length of ***d*** that will help to determine the angle that we are looking for, using the next formula.

$\boldsymbol{a} = \boldsymbol{d} * \cos(y)$, where *y* is the angle that we need.

By transforming the equation we obtain the angle.

$cos(y) = \boldsymbol{a} / \boldsymbol{d}$ or, $y = cos(\boldsymbol{a}/\boldsymbol{d})$

In this way the angle formed between where the user is looking at and the height of the point is obtained. Finally, and similar to the horizontal projection, we just need to compare angles, the angle ***y*** (calculated just now) and the angle that restricts the vertical field of view represented as ***c***, if the angle *y* is lesser than the boundary angle it means that the point is in the angle of view, if it is greater it means that the point is outside the angle of view and it will not be drawn in the device's display. In this way we can determine if the point is inside the field of view and calculate the exact screen coordinate.

4. RESULTS

In this section we present results from WordPlus, our augmented reality application. Results were carried out on a HTC Wildfire, with a 528 Mhz CPU, 512MB ROM and 384MB RAM. Fig 6, shows how the application is shown to the user. Namely the application consist in showing the user points of interest (poi) that are located near the location of the user, in this case four points of interest are shown (in green), which were gathered from the Panoramio system, the user can interact with these tags and request additional information.

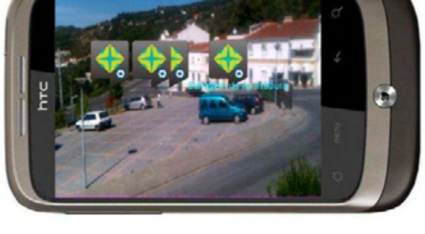


Figure 6: Points of interest (Poi) in WorldPlus.

The user "browses" the world that surrounds him and discovers points/green tags when he directs the camera at them. When the user holds the device with an inclination the points will react to that motion, pushing the points up if the inclination is towards the ground like (Figure 7).

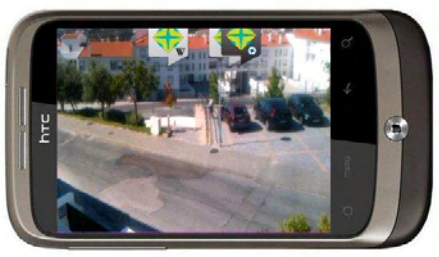


Figure 7: Poi projected with inclination.

Or with an inclination towards the sky pushing the points to the bottom of the screen (Figure 8).

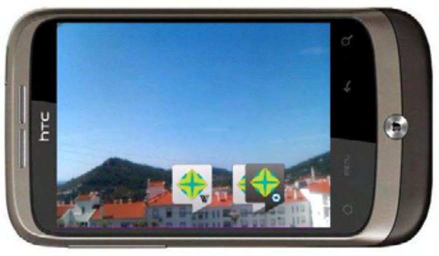


Figure 8: Poi projected with inclination.

We found that the biggest bottleneck for rendering is the projection calculation. In order to determine the maximum number points of interest that the system can compute coordinates and render whilst still being interactive, we performed some tests running the application with different amounts of points of interest in memory at a given time. Namely tests computing 25, 50 and 100 points of interest were carried out. In the first case the application runs smoothly in a range of 10-15 frames per second, the user can have a fluid navigation experience with no "freezing" effect at all. With 50 points the result was a range of 6-10 frames per second, again a good experience is provided, it is possible to move the equipment around and the points are projected almost instantly. With 100 points the result was 4-8 fps, in this case when the user moves the equipment it can take a couple of seconds to update the points on the screen. These tests led us to limit our system to 50 points of interest so that the user can have the best experience with it. On the network/transfer of images side, tests showed that 20 points of interest retrieved from the Panoramio take an average of 9 seconds to load and parse all the data. We note that with the parsed offline wikipedia dump file, these access lags are non-existent as points are loaded into main memory when the application starts.

5. FUTURE WORK

In the future, we would like to obtain more sources of points of interest, there are many services online that contain geographical referenced information that can be used in this type of application. Our application is prepared for scalability, new sources of points are easily implemented with a few lines of code. We would like to create a way for the users to create their own points of interest and submit them so that they can be shared with friends or other users of the service. We would like to find a way to more efficiently parse Wikipedia. One way would be to set up a computer that stores a database with all the information of the Wikipedia articles that are needed in the application, and create some services for accessing that data across the internet. In this way it would be possible to update the information that is used in the application in a transparent manner, on the side of the application it is not known if the information stored in the server is up to date or not, it just makes calls to retrieve the data. All the updates could be achieved by a routine on the server side that picks up the new dumps given by the wikipedia, those dumps could be parsed and the resulting information could be stored in the database that is accessed by the application.

6. CONCLUSIONS

We have presented WorldPlus. An Augmented Reality Application with Georeferenced content for smartphones. We described the requirements, concepts, components, problems and designed solutions using the Android platform as an example. We presented problems for mobile devices such as the size of the retrieved data and dynamic content that are likely to still be research topics in the future.

REFERENCES

[Jun12a] Junaio Home Page http://www.junaio.com/

[Lay12a] Layar Home Page http://www.layar.com/

[Mil94a] Milgram P., Takemura H., Utsumi A., Kishino F., 1994. Augmented Reality: A class of displays on the reality-virtuality continuum..

[Mat11a] Matuscheck J. - Finding Points Within a Distance of a Latitude/Longitude Using Bounding Coordinates:http://janmatuschek.de/LatitudeLongitude BoundingCoordinates. At 2011-08-10.

[Pap08a] Papagiannakis, G., Singh, G., Thalmann, N., 2008, "A survey of mobile and wireless technologies for augmented reality systems2, in Computer Animation and Virtual Worlds, Vol. 19, No. 1, pp.3-22.

[Rei04a] Reicher, T., 2004, "A Framework for Dynamically Adaptable Augmented Reality Systems", PhD thesis, Technische Universität München, library.

[Sek12a] http://sekaicamera.com/

[Wik12a] http://www.wikitude.org/en

[Phy12a] www.physics.csbsju.edu/astro/CS/CS.05.html

WSCG 2012

Index

Abdulla,W. 259
Ahmad,M.A. 367
Acharya,S. 357
Arora,N. 347
Aryal,J. 327
Bahnsen,C. 231
Benger,W. 357
Beran,V. 205
Bian,X. 341
Bittorf,B. 269
Blanz,V. 59
Brener,N. 357
Bruni,V. 283
Bugaj,M. 291
Crumley,Z. 113
Cyganek,B. 291
Daniel,M. 179
de Rezende,P.J. 27
Debelov,V. 189
Delmas,P. 249
Dewilde,A. 231
Drap,P. 275
Falcao,A.X. 27
François,A. 327
Gain,J. 113
Gargalik,R. 163
Gillies,D.F. 69
Gomes,J.F. 27
Graca,S. 377
Guthe,M. 59
Hast,A. 11
Hoppenheit,J. 155
Hrmo,I. 163
Hulík,R. 223
Ihrke,I. 239
Iyengar,S. 357
Jawad,M. 335
Jean-Marc Boi,J.-M. 275
Jimenez,J.R. 105
Kakimoto,M. 95
Kalra,P. 347
Karki,B. 357
Kenwright,B. 1
Khurana,S. 357
Klein,A. 53
Klein,A., 197
Klicnar,L. 205
Köppen,V. 35
Kozlov,D. 189
Krim,H. 341
Krivokuca,M. 259
Krömker,D. 87
Kršek,P. 223
Kumar,A. 347
Kurowski,M. 79
Lavoué,G. 259
Lazunin,V. 131
Lee,G.R. 45
Lee,H.C. 45
Lee,T.M. 45
Leonardi,V. 179
Lutteroth,C. 249
Maddock,S. 317
Madsen,C.B. 231
Mahiddine,A. 275
Malik,M. 367
Marais,P. 113
Marchetti,A. 11
Mari,J.L. 179
Marks,S. 169
Masik,S., 35
Merad,D. 275
Metzgar,J. 147
Minich,C. 309
Minoi,J.-L. 69
Morik,M. 35

Müller,R. 35
Müller,S. 155
Nakata,N. 95
Nguyen,M.H. 249
Nishita,T. 95
Nischwitz,A. 53 197
Noguera,J.M. 105
Obermeier,P. 53 197
Oliveira,J.F. 377
Oshita,M. 213
Pedersen,C. 231
Pimenta,W. 139
Qureshi,H. 367
Raffin,R. 327
Realinho,V. 377
Reuter,A. 239
Ritter,M. 357
Robert,A.J. 69
Rossi,E. 283
Roy,S. 357
Safdar,K. 299
Saito,P.T.M. 27
Santos,L.P. 139
Sarfraz,M.S. 335
Savchenko,V. 131
Seidel,H.-P. 239
Seinturier,J. 275
Semwal,S.K. 147
Schiffner,D. 87
Schmidt,M. 59
Schumann,M. 155
Spanlang,B. 19
Suzuki,C.T.N. 27
Tang,Y. 123
Tappert,B. 197
Tomori,Z. 163
Tranchet,G. 231
Vassilev,T.I. 19
Vidal,V. 179
Vitulano,D. 283
Warburton,M. 317
Windsor,J. 169
Wu,Z. 123
Wuensche,B. 169 249 259
Wüthrich,C. 269
Yasin,M. 335
Yoon,G.H. 45
Zhou,M. 123