ISSN 1213-6972 *Volume 27, Number 1, 2019*

Journal of WSCG

An international journal of algorithms, data structures and techniques for computer graphics and visualization, surface meshing and modeling, global illumination, computer vision, image processing and pattern recognition, computational geometry, visual human interaction and virtual reality, animation, multimedia systems and applications in parallel, distributed and mobile environment.

Vaclav Skala – Union Agency

Journal of WSCG

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

Managing Editor: Vaclav Skala

Printed and Published by:
Vaclav Skala - Union Agency
Na Mazinach 9
CZ 322 00 Plzen
Czech Republic

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

Hardcopy: ***ISSN 1213 – 6972***
CD ROM: ***ISSN 1213 – 6980***
On-line: ***ISSN 1213 – 6964***

Journal of WSCG

Editor-in-Chief

Vaclav Skala

c/o University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering
Univerzitni 8, CZ 306 14 Plzen, Czech Republic
http://www.VaclavSkala.eu

Journal of WSCG URLs: http://www.wscg.eu or http://wscg.zcu.cz/jwscg

Editorial Board

Journal of WSCG

Board of Reviewers
2019

Anderson,M. (Brazil)
Assarsson,U. (Sweden)
Ayala,D. (Spain)
Baranoski,G. (Canada)
Benes,B. (USA)
Benger,W. (Austria)
Bilbao,J. (Spain)
Birra,F. (Portugal)
Bouatouch,K. (France)
Boukaz,S. (France)
Bourke,P. (Australia)
Cakmak,H. (Germany)
Carmo,M. (Portugal)
Carvalho,M. (Brazil)
Coquillart,S. (France)
Dachsbacher,C. (USA)
De Martino,J. (Brazil)
Durikovic,R. (Slovakia)
Eisemann,M. (Germany)
Feito,F. (Spain)
Ferguson,S. (United Kingdom)
Galo,M. (Brazil)
Gavrilova,M. (Canada)
Gdawiec,K. (Poland)
Gerhards,J. (France)
Giannini,F. (Italy)
Goncalves,A. (Portugal)
Gudukbay,U. (Turkey)
Guthe,M. (Germany)
Hansford,D. (USA)
Hermosilla Casajus,P. (Germany)
Chaudhuri,D. (India)
Chover,M. (Spain)
Ivrissimtzis,I. (United Kingdom)
Jeschke,S. (Austria)
Joan-Arinyo,R. (Andorra)
Jones,M. (United Kingdom)
Juan,M. (Spain)
Kanai,T. (Japan)
Klosowski,J. (USA)
Komorowski,J. (Poland)
Krueger,J. (Germany)
Kurt,M. (Turkey)
Kyratzi,S. (Greece)
Larboulette,C. (France)
Lee,J. (USA)

Lisowska,A. (Poland)
Liu,S. (China)
Lobachev,O. (Germany)
Manoharan,P. (India)
Manzke,M. (Ireland)
Mastermayer,A. (Germany)
Max,N. (USA)
Meyer,A. (France)
Mohd Suaib,N. (Malaysia)
Molla,R. (Spain)
Montrucchio,B. (Italy)
Muller,H. (Germany)
Nishio,K. (Japan)
Oliveira,J. (Portugal)
Oyarzun Laura,C. (Germany)
Pan,R. (China)
Papaioannou,G. (Greece)
Paquette,E. (Canada)
Pasko,A. (United Kingdom)
Pedrini,H. (Brazil)
Ramires Fernandes,A. (Portugal)
Richardson,J. (USA)
Rodrigues,N. (Portugal)
Rojas-Sola,J. (Spain)
Sarfraz,M. (Kuwait)
Savchenko,V. (Japan)
Segura,R. (Spain)
Shum,H. (United Kingdom)
Sik-Lanyi,C. (Hungary)
Skala,V. (Czech Republic)
Sousa,A. (Portugal)
Stuerzlinger,W. (Canada)
Szecsi,L. (Hungary)
Thalmann,D. (Switzerland)
Todt,E. (Brazil)
Tokuta,A. (USA)
Toler-Franklin,C. (USA)
Torrens,F. (Spain)
Trapp,M. (Germany)
Tytkowski,K. (Poland)
Vanderhaeghe,D. (France)
Weber,A. (Germany)
Wuensche,B. (New Zealand)
Wuethrich,C. (Germany)
Cai,Y. (Singapore)
Yin,Y. (USA)
Yoshizawa,S. (Japan)
Zwettler,G. (Austria)

Journal of WSCG

Vol.27, No.1, 2019

Contents

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.27, No.1, 2019

Fast Adaptive Undersampling for Volume Rendering

Sergey Belyaev
Peter the Great St. Petersburg Polytechnic University
Polytechnicheskaya 29
195251 St. Petersburg, Russia
bel_s_u@mail.ru

Pavel Smirnov
Peter the Great St. Petersburg Polytechnic University
Polytechnicheskaya 29
195251 St. Petersburg, Russia
psmirnov@spbstu.ru

Natalia Smirnova
EPAM Systems, Inc.
41 University Drive, Suite 202
Newton, PA 18940, USA
Natalia_Smirnova@epam.com

Vladislav Shubnikov
Peter the Great St. Petersburg Polytechnic University
Polytechnicheskaya 29
195251 St. Petersburg, Russia
vlad.shubnikov@gmail.com

ABSTRACT

Adaptive undersampling is a method for accelerating the rendering process by replacing the calculation of a volume integral with an interpolation procedure for a number of pixels. In this paper, we propose a method for accelerating the volume integral calculation for the rest of the pixels, i.e. those pixels for which interpolation cannot be done with sufficient accuracy. This method requires two passes through the input data. On the first pass, rendering is done into a low-resolution texture. At this stage, the values of the volume integral on a set of intervals of a given length are calculated and saved into a special G-buffer alone with the pixel's color. On the second pass, these values are used to determine colors of the pixels. For those pixels whose result is not precise enough, the volume integral is calculated on one or several intervals, rather than the whole ray. The proposed method allows one to accelerate adaptive undersampling by a factor of 1.5 on average, depending on the input data.

Keywords

Volume rendering, Ray casting, Adaptive sampling.

1. INTRODUCTION

The main visualization method for volumetric scientific data (e.g. medical data) is direct volume rendering, which calculates the value of the volume integral for each screen pixel. This approach uses scanning of the large volumes of data efficiently using various transfer functions, but this process is computationally expensive. Its running time is proportional to the number of pixels in the visualization window, so its optimization for high-resolution screens and devices with low computing power is a relevant problem. Examples of such devices include mobile phones, laptops and PCs with slow video cards, as well as VR devices, which require a minimum of 60 FPS while rendering into two cameras at the same time.

The method is usually implemented on GPUs in conjunction with various optimization techniques—discarding regions on which the transfer function is zero [LCDP12], varying the integration step [CCF15], pre-integrated volume rendering [KE04], and adaptive undersampling (or screen undersampling) [KRHH11]. Adaptive undersampling makes use of the coherency of the scene in order to minimize the number of volume integrals to be calculated to determine the color of pixels in the image. This is achieved by an iterative procedure. On the first iteration, only part of the pixels is sampled (one for each $n \times n$ block), and then an attempt is made to recover the colors of the rest of the pixels with the information thus obtained (for example, by interpolating bilinearly between the colors of adjacent pixels). If this does not produce the required image quality, then the set of pixels being sampled is expanded. In practice, most input data sets (including medical data) have high levels of spatial coherence, which means that after the first iteration, only around

10% to 20% of the total number of pixels need to be sampled additionally. Unfortunately, when the algorithm is implemented on a GPU, additional calculations for some pixels lead to increase of the processing time for all pixels of the image, as if the optimization is completely absent. This is because the pixels are processed concurrently in groups, and the time it takes to process a group is equal to the maximum of the times to process each pixel. Thus, if at least one pixel in the group calls for the calculation of a volume integral, then the whole group will take exactly as much time to process as if every pixel's volume integral had to be calculated.

In this paper, we propose a two-pass algorithm which solves the problem by making the calculation of the volume integrals on the second pass much faster for pixels whose colors cannot be interpolated. To do this, on the first pass, the domain of integration for the volume integral is broken up into M pieces, and the values of the integral on each piece are saved into the G-buffer. On the second pass, the volume integral is calculated by summing its values on M pieces. These values are determined either by bilinearly interpolating the corresponding G-buffer values, or, if that is not possible, by integration.

Below is an overview of related work (Section 2), followed by a discussion of what we consider to be our main contribution: a method for accelerating volume rendering by pre-computing the volume integrals on multiple intervals for part of the pixels (Section 3). We discuss the results in Section 4 and make conclusions in Section 5.

2. RELATED WORK

The most flexible and widespread method for direct volume rendering is raycasting. GPU-based raycasting was proposed in [KW03]. It uses cube proxy geometry (the bounding box of the dataset) to determine the starting and ending points of the way. However, the method is slow, as it requires the volume integral to be calculated for every pixel by going down the whole ray from start to end with some step. Adaptive sampling can be used for raycasting optimization. This allows to obtain the output image by calculating the volume integral for only part of the pixels. This was first proposed in [Lev90], in which the volume integral is calculated in the corners of equally sized blocks into which the image is partitioned. If the values in these corners do not differ significantly, then the colors of the interior pixels of the block are interpolated bilinearly. Otherwise, the block is partitioned into four parts, and the procedure is applied recursively to each part. Kratz et al. [KRHH11] present a variation of Levoy's approach for GPU-based rendering. They replaced the comparisons of the integrals at the blocks' corners with a more sophisticated technique based on finite element methods (FEM) to achieve explicit error control. In their implementation, raycasting was done on the GPU, while the hierarchical data structures of the blocks (quadtree) were stored on the CPU. [KSK*16] and [BSSS18] examine methods for excluding artifacts which arise due to the fact that volume integrals are not calculated for all pixels.

On a GPU, recursive division leads to multi-pass algorithms which turn out inefficient due to the architecture of a GPU. Thus, [L15] uses a two-pass algorithm, in which the colors of interior pixels are calculated either via bilinear interpolation or by calculating the volume integral. The two-pass algorithm is also used in [BFE16] in order to optimize raytracing on mobile devices.

In [BSM18] the second rendering pass is accelerated by saving (on the first pass) volume integral values in the ray intervals, where the transfer function value is not zero. Unfortunately, this algorithm is effective only when number of intervals is relatively small.

3. ALGORITHM

3.1. Overview

The volume integral for each pixel gives the fraction of light passing through the volume along the pixel's view ray. The discrete form of this integral can be efficiently computed via compositing, which replaces a Riemann sum with a recurrence relation:

$$C_{i+1} = C_i + (1 - A_i) \cdot a_i \cdot c_i, \qquad (1)$$
$$A_{i+1} = A_i + (1 - A_i) \cdot a_i.$$

In the above equations, C_i is the composited color on the i'th step along the ray, A_i is the composited transparency, and a_i and c_i are, respectively, the transparency and color in the given sample.

The proposed algorithm is based on two-pass adaptive undersampling. The set of pixels is partitioned into $n \times n$ blocks, and on the first pass one pixel from each block is processed. However, unlike the method above, our algorithm divides the interval of integration into M equal pieces, and the values of the integral over these pieces are saved into the G-buffer along with the color of the pixel.

More details concerning M value will be explained in section 4. In Figure 1, which depicts the

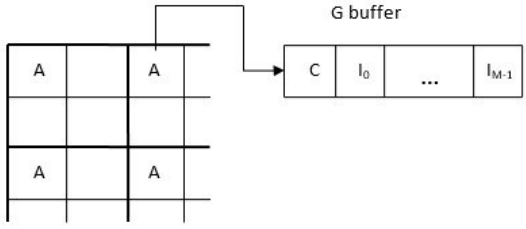


Figure 1: The G-buffer contains the color (C) of the pixel and the values (I_i) of the volume integral on a set of equal length intervals.

case $n = 2$, the pixels being processed are marked with an A. The volume integral over the i'th interval is denoted with I_i ($i < M$). C and I_i are 4-component vectors, containing the colors C_i and transparencies A_i. Colors C_i contain three components: red, green, blue.

The rest of the pixels are processed on the second pass. If the colors of their adjacent pixels are sufficiently close, then the color C^P of the current pixel is interpolated bilinearly. Otherwise, it is calculated with the following recurrence relation, according to [HLSR09]:

$$C^P{}_{i+1} = C^P{}_i + (1 - A^P{}_i) \cdot A_i^* \cdot C_i^*. \quad (2)$$
$$A^P{}_{i+1} = A^P{}_i + (1 - A^P{}_i) \cdot A_i^*, \quad 0 \le i \le M - 1.$$

In the above equations, A_i^* and C_i^* are interpolated bilinearly from the values of A_i and C_i in the adjacent pixels if those values are close enough and are calculated from the volume integral otherwise. Thus, on the second pass the volume integral is only calculated over the part of the ray in the worst case, which significantly accelerates the generation of the whole image.

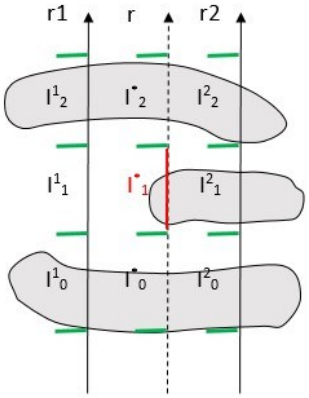


Figure 2: The volume integral only needs to be calculated on the red interval.

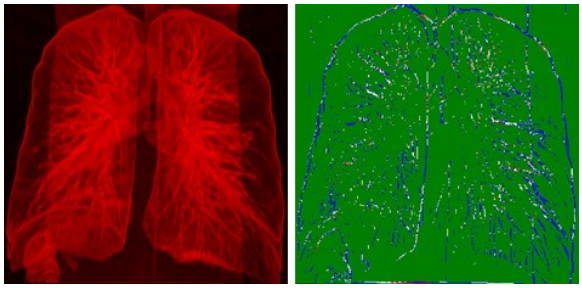


Figure 3: Left: the resulting image. Right: the pixels whose volume integrals were interpolated bilinearly are shown in green (91.7%); those for which the volume integral had to be calculated over one interval are shown in blue (5.7%); over two intervals, in white (2.1%); over three intervals, in yellow (0.1%); and more than three intervals in red (0.4%).

The algorithm accelerates rendering by reducing the length of the interval of integration. In Figure 2 (drawn in two dimensions for simplicity), an example is shown for the second pass of the algorithm for the case $M = 3$, where r_1 and r_2 denote the rays passing through pixels processed on the first pass. The integrals I_i^1 and I_i^2 have been calculated and are stored in the G-buffer. The current pixel being processed is on the ray r. The values of A_i^* and C_i^* for $i = 0, 2$ are interpolated from I_i^1 and I_i^2, while A_1^* and C_1^* are calculated via the volume integral I_1^* on the given interval. Figure 3 shows the number of pixels in a real dataset for which the volume integral needs to be calculated on the second pass, and the number of intervals on which it must be calculated. The pixels whose volume integrals were interpolated bilinearly are shown in green. Those for which the volume integral had to be calculated are colored based on how many intervals it had to be calculated on: blue for 1, white for 2, yellow for 3 and red for more than 3. As can be seen from the figure 3, in most cases the integral only needed to be calculated over one interval, which is what makes the algorithm so efficient. The following is a detailed description of the algorithm.

3.2. Algorithm details

In the first pass, the algorithm fills the M parallel textures (in the G-buffer) which have a resolution n times less than the viewport (along each side). Algorithm 1 shows a pseudocode for each ray calculation. The function *GetDistanceForStart*(s, f) called in line 1 finds the distance from the starting point s of the ray to the point v where it first meets the domain where the transfer function is not zero (Figure 4). Here, f is a final point on ray, both s, f are 3D vectors, step initialization is explained below.

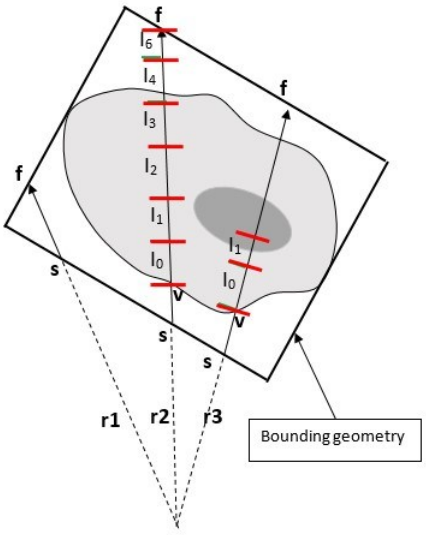


Figure 4: No integration is done over r_1. Over r_2 and r_3 integration begins at the points marked with v.

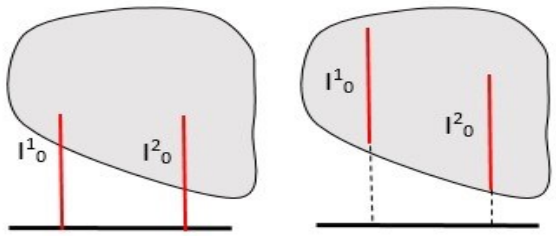


Figure 5: On the left, $I^1_0 \neq I^2_0$; on the right, $I^1_0 = I^2_0$.

Condition ($A < 1$) in line 18 means that total opacity has not reached 1, i.e. integration process along ray has not stopped at this point. Important detail: last integration calculation has another "*lastStep*", not equal to "*step*" and will be explained in detail below. *VolumeIntegral*(v, *step*) is a function, calculating the partial sum along a ray, starting from point v with "*step*" length. We find the point v for two reasons. First, the probability that the first integrals I_0 coincide on adjacent rays increases (see figure 5). Second, this helps remove "woodgrain" artifacts, especially in cases where the derivative of the opacity function is high in a neighborhood of v. This is explained in more detail in [LJKY13]. If v is not found, the function returns −1 and the algorithm halts (as in the case of the ray r_1 in Figure 4). Otherwise, the algorithm calculates the color of the pixel and the volume integral over intervals of equal length (r_2).

All calculations are done in the texture space of the 3D texture which stores the data to be visualized. All samples are contained in a cube with sides equal to 1, so the longest ray in the texture space has length $\sqrt{3}$ (the diagonal of the cube). This value is used to calculate the interval length in line 6, where M is the user-selected maximum number of intervals. The integrals are calculated in line 12 and are stored in the G-buffer; the integration itself can be done using any known method. The color of the pixel is stored in G[0].rgb in line 13. As can be seen from Figure 4, the last interval of integration can be shorter than the rest; this interval is processed in lines 18–21.

On the second pass, the colors are calculated for those pixels which were not processed on the first pass. Shown below is the Algorithm 2 that does this. It uses data from the G-buffer which was created on the first pass for the four neighboring pixels. In line 2, the current pixel's color is interpolated bilinearly if the neighbors' colors are sufficiently close.

Index i in G_i means neighborhood texel, calculated on the first pass. Index i can be in range [0…3], due to four neighborhood texel for current ray, calculated during second pass. Condition for simple bilinear interpolation is based on comparison maximum color difference for neighborhood pixels with some parameter *delta*.

Algorithm 1 The first pass algorithm

```
1:  G[0].a = GetDistanceForStart(s, f);
2:  if G[0].a ≤ 0 then
3:    G[0].rgb = BackgroundColor;
4:  else
5:    G = 0, i = 0, C = 0, A = 0;
6:    step = √3 / M;
7:    imax = floor(length(f − s) / step);
8:    lastStep = length(f − s) − step ∗ imax;
9:    r = normalize(f − s);
10:   v = s + G[0].a · r;
11:   while A < 1 and i ≤ imax do
12:     G[i + 1] = VolumeIntegral(v, step);
13:     C = C + (1 − A) · G[i + 1].rgb ×
          × G[i + 1].a;
14:     A = A + (1 − A) · G[i + 1].a;
15:     v = v + step · r;
16:     i = i + 1;
17:   end while
18:   if A < 1 then
19:     G[imax + 1] = VolumeIntegral(v, lastStep);
20:     C = C + (1 − A) · G[imax + 1].rgb ×
          × G[imax + 1].a;
21:   end if
22:   G[0].rgb = C;
23: end if
```

Algorithm 2 The second pass algorithm

```
1:  if for all i,
        max_j||G^j[0].rgb|−|G^i[0].rgb|| < delta then
2:    C = BilinearInterpolation(G_i.rgb);
3:  else
4:    A = 0, C = 0;
5:    t = r · min{G_i[0].a};
6:    v = s + r · GetDistanceForStart(s + t, f);
7:    for k = 1 … M do
8:      if for all i,
            max_j||G^j[k].rgb|−|G^i[k].rgb|| < delta/M then
9:        I = BilinearInterpolation(G_i);
10:     else
11:       if k = M then
12:         Length = step;
13:       else
14:         Length = lastStep;
15:       end if
16:       I = VolumeIntegral(v, Length);
17:     end if
18:     C = C + (1 − A) · I.rgb · I.a;
19:     A = A + (1 − A) · I.a;
20:     v = v + step · r;
21:   end for
22: end if
```

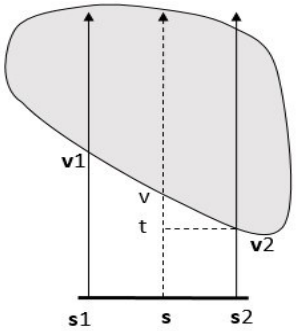


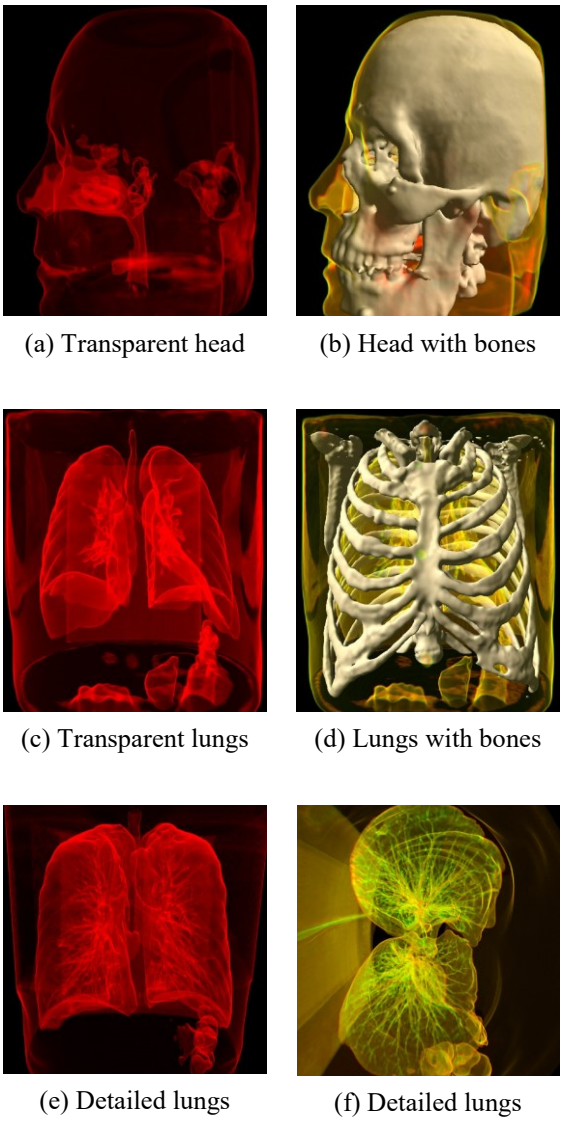
Figure 6: Calculating the starting point for integration.

In line 6, the algorithm finds the first point along the ray where the transfer function is not zero. This uses the same *GetDistanceForStart* function as in the first pass, but in order to accelerate its execution the ray is cast from $s + r \cdot \min\{G_i[0].a\}$, rather than from s (see Figure 6). The loop (lines 7–21) implements the recurrence relations in formula 2. The integral calculation function in line 16 coincides with the function used in the first pass.

4. RESULTS AND DISCUSSION

All tests were performed on a 3.4GHz Intel Core i7 2600 PC with 4.0GB of main memory with NVidia GForce GTX 780 Ti graphics hardware with 3072MB of texture memory and implemented using Unity3D, using OpenGL ES 3.1. Three CT data sets were used as testing data; their characteristics and screenshots are given in Figure 7.

Table 1 contains the framerate achieved in visualizing the data sets. The volume integrals were calculated using the standard method [KW03] with ¼ of the voxel size as the step size. The bounding volume was chosen to be a box. The viewport was 1200×900 pixels. The value of M was chosen as 8, which is the maximum possible size of the G-buffer on the video card used. The value of delta (see Algorithm 2) was chosen as 0.05.



(a) Transparent head (b) Head with bones

(c) Transparent lungs (d) Lungs with bones

(e) Detailed lungs (f) Detailed lungs

Figure 7: Data sets; resolution is 256×256×256 for (a)–(d) and 512×512×136 for (e)–(f).

Data Set	OpenGL ES 3.1 FPS			B/A	C/A	C/B
	A	B	C			
(a)	32	72	118	2.25	3.69	1.64
(b)	48	92	116	1.92	2.42	1.26
(c)	34	68	112	2.00	3.29	1.65
(d)	40	66	74	1.65	1.85	1.12
(e)	18	32	50	1.78	2.78	1.56
(f)	20	42	74	2.10	3.70	1.76

Table 1: The framerate achieved in visualizing the data sets.

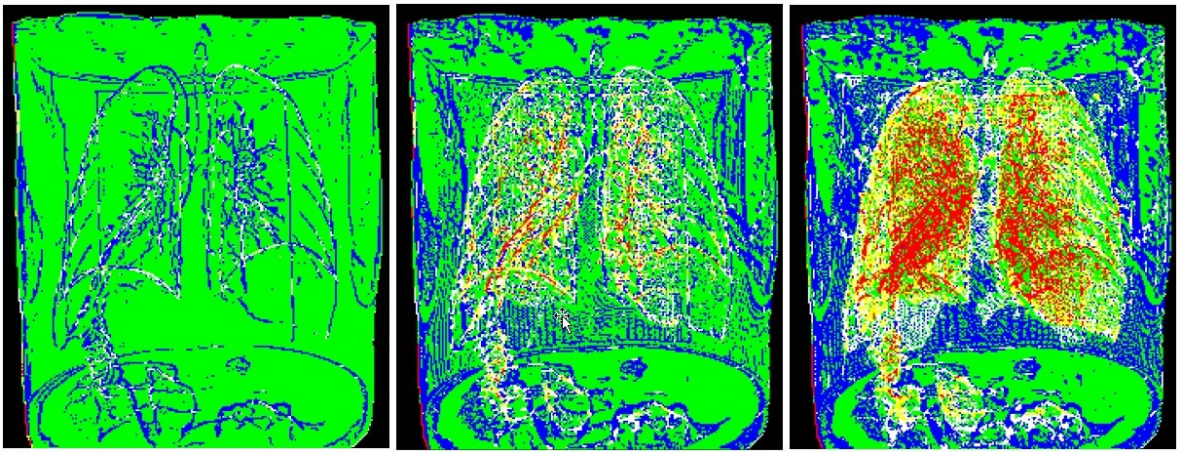


Figure 8: The number of intervals of integration per pixel for delta = 0.05, 0.01, 0.005 (left to right). Color key: green = 0 (bilinear interpolation), blue = 1, white = 2, yellow = 3, red = more than 3.

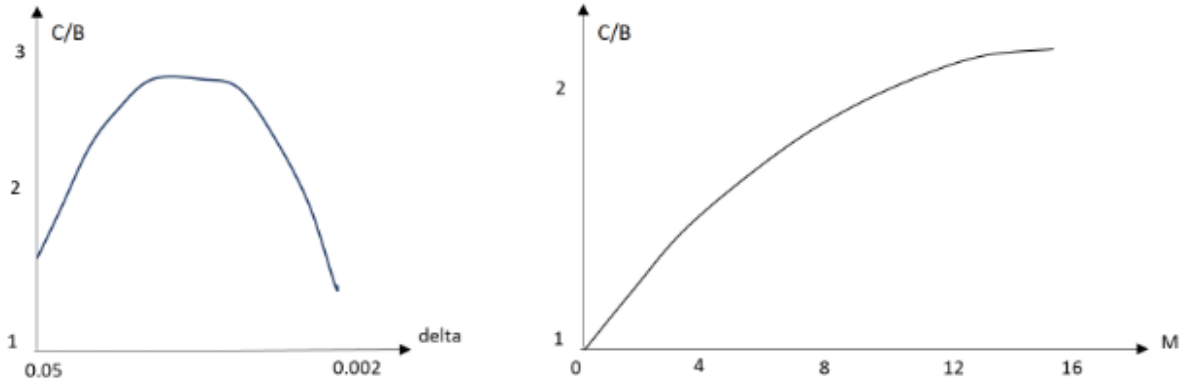


Figure 9: left: The acceleration factor as a function of delta; right: The acceleration factor as a function of M.

Table 1 is organized as follows: the first column lists the reference to the dataset from Figure 7, the columns labeled A, B, C contain the framerates obtained with the following optimization methods:

A. No optimization.

B. Two-pass adaptive screen sampling.

C. Two-pass adaptive screen sampling plus partitioning the interval of integration into M = 8 pieces.

The next two columns contain the acceleration factors achieved using, respectively, two-pass adaptive screen sampling and the proposed algorithm. The last column contains the acceleration factors achieved only by using the proposed algorithm.

You can note from the last column of the Table 1 that proposed method, by itself, increases FPS by a factor of 1.5 on the data sets used. This factor becomes smaller if a significant number of rays end early (for example, for the bones see Figure 7 in screenshots (b) and (d)). The reason for this is that most intervals of integration are short and are harder to partition into smaller ones.

The efficiency of the method also depends on how coherent the dataset is. The less coherent it is, the more pixels need to be processed on the second pass. This can happen if the value of delta is lowered. Thus, to a first-order approximation, the dependence on the data sets' coherence can be replaced with a dependence on delta. Figure 8 shows the pixels processed on the second pass in visualizing data set 3 with delta values 0.05, 0.01 and 0.002. (Blue pixels are those for which the integral had to be computed over one interval; white, over two; yellow, over three; and red, over more than three.)

Figure 9 (left) shows the acceleration factor (relative to standard adaptive undersampling) as a function of delta. It can be seen from the graph that the efficiency of our method is at its maximum for medium levels of coherence. The reason for the decrease in performance on low coherence is that volume integrals need to be calculated for more pixels (red pixels in Figure 8). The decrease in

performance for high coherence is because a small number of pixels need to be processed on the second pass.

C/B on horizontal axis shows how much rendering frame rate is increased (in times) when using proposed interval partitioning in a comparison with usual two-pass adaptive screen sampling algorithm.

Figure 9 (right) shows the graph of the average efficiency of the method as a function of *M*, where the average is taken over data sets 1, 3 and 5. (Since the video card used allows only 8 elements in the G-buffer, only one component was used for color in this case, which allowed two integrals to be stored in one G-buffer, effectively increasing the value of *M* to 16.) As can be seen from the graph, changing *M* from 8 to 16 can lead to a further increase in the algorithm's performance—around 21% on average—but given the shape of the curve, we consider it ineffective to increase *M* further.

5. CONCLUSION

We have proposed an algorithm which allows adaptive undersampling to be increased by a factor of 1.5 on average. The algorithm is most efficient when the input data has medium coherence and the transfer function given excludes early ray termination. If these conditions are satisfied, the acceleration factor can exceed 2.5. This algorithm inherits the limitation of adaptive undersampling: the input data must be spatially coherent.

6. REFERENCES

[BFE16] Bruder V., Frey S., and Ertl T.: Real-time performance prediction and tuning for interactive volume raycasting. In SIGGRAPH ASIA 2016 Symposium on Visualization, SA '16, pp 7:1–7:8, New York, NY, USA, 2016. ACM.

[BSSS18] Belyaev S., Smirnov P., Shubnikov V., and Smirnova N.: Adaptive algorithm for accelerating direct isosurface rendering on GPU. Journal of Electronic Science and Technology, Volume 16, Issue 3, 2018, pp 222–231.

[BSM18] Belyaev S., Shubnikov V., and Motornyi N.: Adaptive Screen Sampling Algorithm Acceleration for Volume Rendering. IADIS International Conference Interfaces and Human Computer Interaction 2018 (part of MCCSIS 2018), conference proceedings, pp 377–381.

[CCF15] Campagnolo L. Q., Celes W., and de Figueiredo L. H.: Accurate volume rendering based on adaptive numerical integration. In 2015 28th SIBGRAPI Conference on Graphics, Patterns and Images, pp 17–24, Aug 2015.

[HLSR09] Hadwiger M., Ljung P., Rezk Salama C., and Ropinski T.: Advanced illumination techniques for gpu-based volume raycasting. In ACM SIGGRAPH 2009 Courses, SIGGRAPH '09, pages 2:1–2:166, New York, NY, USA, 2009. ACM.

[KE04] Kraus M. and Ertl T.: Pre-integrated volume rendering. In Visualization Handbook, 01 2004.

[KRHH11] Kratz A., Reininghaus J., Hadwiger M., and Hotz I.: Adaptive screen-space sampling for volume ray-casting. Technical Report 11–04, ZIB, Takustr. 7, 14195 Berlin, 2011.

[KSK*16] Kim Y., Seo W., Kim Y., Lim Y., Nah J., and Ihm I.: Adaptive undersampling for efficient mobile ray tracing. Vis. Comput., 32(6-8):801–811, June 2016.

[KW03] Kruger J. and Westermann R.: Acceleration techniques for GPU-based volume rendering. In Proceedings of the 14th IEEE Visualization 2003 (VIS'03), page 38, IEEE Computer Society, Washington, DC, USA, 2003.

[L15] Lesar Z.: Real-time ray casting of volumetric data. In IEEE EUROCON 2015—International Conference on Computer as a Tool (EUROCON), pp 1–6, Sept 2015.

[LCDP12] Liu B., Clapworthy G.J., Dong F., and Prakash E.C.: Octree rasterization: Accelerating high-quality out-of core GPU volume rendering. In IEEE transactions on visualization and computer graphics, volume 19, pp 1732–1745, 07 2012.

[LJKY13] Lindholm S., Jonsson D., Knutsson H., and Ynnerman A.: Towards data centric sampling for volume rendering. In SIGRAD, pp 55–60, 2013.

[Lev90] Levoy M.: Volume rendering by adaptive refinement. The Visual Computer, 6(1):2–7, Jan 1990.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.27, No.1, 2019

Silhouette Extraction for Shadow Volumes Using Potentially Visible Sets

Jozef Kobrtek
Brno University of Technology
Czech Republic
ikobrtek@fit.vut.cz

Tomáš Milet
Brno University of Technology
Czech Republic
imilet@fit.vut.cz

Adam Herout
Brno University of Technology
Czech Republic
herout@fit.vut.cz

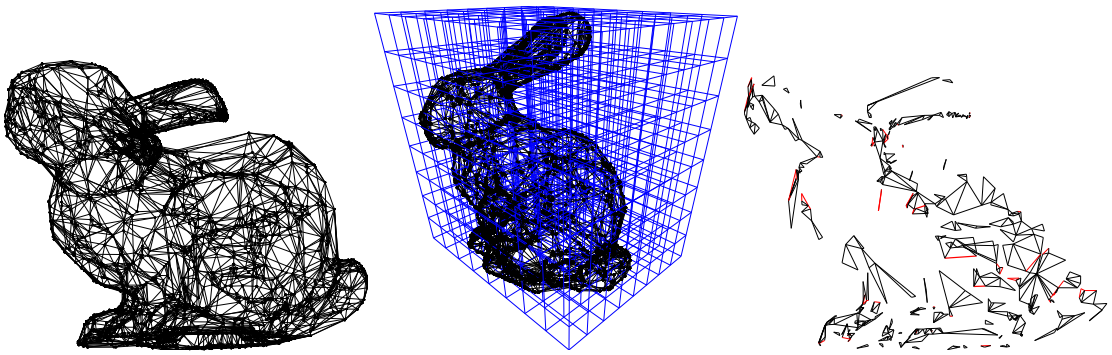


Figure 1: **left:** Wireframe representation of a given model. **middle:** We voxelize the space around the model. One voxel on the lowest octree level is selected, based on the light position, and all potentially-silhouette (need to be tested) and silhouette edges (guaranteed to be silhouette) can be collected by ascending the octree hierarchy. **right:** Red coloured edges are those that are a part of the silhouette after testing the set of potentially silhouette edges (all red and black ones). Only a small subset of model edges need to be tested, which considerably reduces the computational complexity.

ABSTRACT

In this paper, we present a novel approach for accelerated silhouette computation based on potentially visible sets stored in the octree acceleration structure. The scene space, where the light source can appear, is subdivided into voxels. The octree voxels contain two precomputed sets of edges that potentially or always belong to the silhouette. We also propose a novel method of octree compression for reduction of the memory footprint of the resulting acceleration structure. Using our novel technique we were able to considerably decrease the computational complexity of finding the silhouette and reduce its sensitivity to the number of edges.

Keywords

Silhouette Extraction, Octree, Compression, Shadow Volumes

1 INTRODUCTION

Solving surface visibility from a light source (or another point in space in general) is a very fundamental problem of computer graphics. Determining, whether a point on a surface is lit from a point light source has been subject of research for decades, as documented by Woo and Poulin [Woo12]. Over the course of history, two major techniques were developed to address this problem in the field of rasterization – Shadow Maps and Shadow Volumes. Although the majority of the derived methods of the above mentioned techniques are based on Shadow Maps, Shadow Volumes still provide an important option for scenarios requiring sample-precise shadows, which can be problematic when Shadow Maps are involved due to their discrete nature and limited resolution.

Crucial part of the algorithm of Shadow Volumes is silhouette extraction, i.e. finding the subset of edges that have both visible and non visible triangles connected to them from the light's perspective. Such edges are subsequently extruded as the shadow volume side and rendered into the stencil buffer on the GPU. Usually, all the edges are tested during the rendering of a single frame to determine whether they get extruded or not.

In this paper we focused on improving the silhouette extraction performance of the Shadow Volumes by reducing the number of edges that need to be tested during the Shadow Volumes rendering of an arbitrary triangle soup. Edge indices are stored in an octree-like structure created by voxelizing selected scene space. This octree is later traversed by the GPU to acquire two sets of edges – one set that requires further testing (potentially silhouette edges, **PE**), the second set is known to be silhouette (silhouette edges, **SE**) and edges inside the set are extruded immediately.

The remainder of the paper is organized as follows. Section 2 outlines the previous work, focusing on Shadow Volumes and silhouette extraction. The following section 3 introduces the reader to the details of our algorithm, being divided into octree construction, compression and traversal. Section 4 discusses implementation details and problems. Performance and practical analysis as well as limitations are described in Section 5. Finally, we conclude our findings and results in Section 6.

2 RELATED WORK

Shadow Volumes were first introduced by Crow [Crow77]. The core of the algorithm is casting rays from camera to the scene and incrementing/decrementing their value on intersection with extruded shadow volume sides. When geometry is hit by the ray, the fragment is considered lit or shadowed based on the ray value being zero or non-zero. The first GPU implementation came with the introduction of stencil buffer by Heidmann [Hei91]. The drawback of this method is that when the camera is in the shadow, the stencil test must be inverted. The camera problem of Heidmann's method was solved simultaneously by Everitt and Kilgard [Eve02] and by Bilodeau and Songy [Bil99] in the so-called "z-fail" method, which reverses the stencil test, but requires the shadow volumes to be capped. In order to draw an arbitrary triangle soup, Kim et al. [Kim08] introduced the concept of edge multiplicity, so a single quad cast from an edge is rendered multiple times.

Silhouette extraction methods can be divided into 3 categories – image space, object space, and hybrid (computing in object space, displaying in image space), as categorized by Isenberg et al. [Ise03]. The majority of these methods were used to provide object contours for non-photorealistic rendering, but some of the object-based method are interesting from the perspective of Shadow Volumes. Johnson and Cohen [Jon01] use a hierarchy of normal cones to determine edge visibility. Olson and Zhang [Ols06] propose octree as an acceleration structure to store Hough transform of a 3D mesh. Gooch [Goo99] and Benichou and Elber [Ben99] designed a preprocessing method based on projecting face

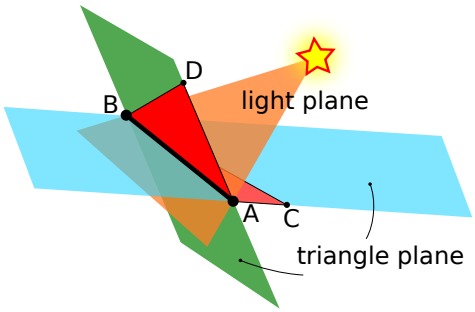


Figure 2: Silhouette edges. Edge AB is not a silhouette edge because triangles ABC and ABD do no lie on the same side of the light plane. Two triangles partition the world space into four subspaces.

normals onto a Gaussian sphere. However, all of these methods are limited to 2-manifold objects.

With the introduction of programmable graphics pipeline, research focused more on the ad-hoc algorithms, mostly due to the fact that the new pipeline provided mechanisms to determine the silhouette during the rendering process with zero or minimal preprocessing. Silhouettes can be computed in almost any programmable shader stage, specifically vertex [Bren02, Mil14], geometry [Sti07], requiring practically no preprocessing, tessellation [Mil15] or compute (OpenCL) [Peč13] shaders.

Gerhards et al. [Ger15] use BSP trees constructed from per-triangle frusta. Fragments are then tested against this structure, whether they are lit or shadowed. This method, however, needs to rebuild the data structure each time the light source or geometry is changed by – even a rigid – transformation.

The proposed method does not require rebuilding (unless the light source moves outside the targeted space) and it is also invariant to affine transformation – the correct voxel in the octree is selected by applying an inverse transformation of the object to the light source and then traversing from the corresponding voxel.

3 ALGORITHM

Our algorithm is based on the concept of the *potentially visible set* (PVS) introduced by Airey et al. [Air90]. It precomputes the results of brute force silhouette extraction for a discrete set of world-space voxels. The brute force extraction process therefore does not need to be executed on all scene edges but only on a small subset that cannot be precomputed, see Figure 1. This section will describe the construction of a compression structure for storing the PVS in an effective manner. It will also describe the modified extraction process.

The algorithm can be broken down into two major stages: **construction** and **traversal**, but first let us summarize the brute force silhouette extraction process.

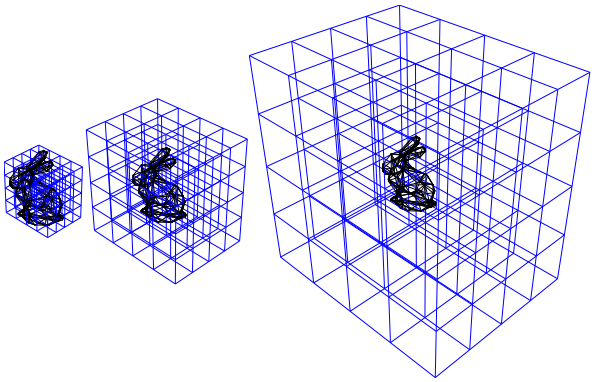


Figure 3: The algorithm supports custom scales of the scene bounding box. If a larger scale is selected, the light can be moved farther from the model. The image shows three different scales with the same voxelization level (in this case 2 levels of octree, $4 \times 4 \times 4$ voxels).

3.1 Silhouette Extraction

A model is composed of vertices that are connected by edges/triangles. An edge is considered as belonging to the silhouette if all triangles adjacent to this edge lie on the same side of the light plane, see Figure 2. In general, from 1 to N triangles can be connected to a single edge. Kim et al. [Kim08] proposed a technique that computes the difference in the number of triangles on the left and the right side of the light plane called edge multiplicity $m \in [-N, N]$.

Without loss of generality, edges with more then 2 connected triangles can be transformed into several simpler edges by splitting and duplicating. If an edge is connected to only one triangle, it is considered a silhouette edge in every case. Our method works with edges having maximum 2 adjacent triangles connected to them.

3.2 Octree Construction

We base the voxelization space on scaling the scene's axis-aligned bounding box (AABB) by a user-specified scaling factor, as seen in Figure 3. The scaling factor depends on the user's needs and on the type of the scene (closed-space scenes will do even with factor 1, open scenes or simple models require larger factors, around 5–10).

This scaled bounding volume circumscribes all the possible light positions. The user can then choose the maximal level of the octree hierarchy, see Figure 4. AABB scaling and maximum octree depth define the octree granularity and voxel size on the deepest level.

We found that depth level of 3–5 is suitable in most scenarios. Larger scales tend to consume too much memory (as described in Chapter 5.1, each octree level increases the amount of memory by a factor of 4). The next step is to find two sets (**SE** and **PE**) for every voxel in the lowest level of the octree. The algorithm tests each edge against all voxels on the lowest level

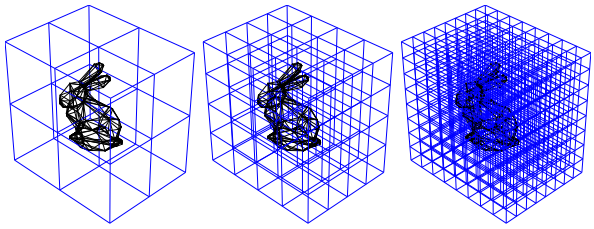


Figure 4: The algorithm supports a custom level of voxelization. The image shows three levels (1,2,3) of depth of the octree for the same scale of the scene bounding box.

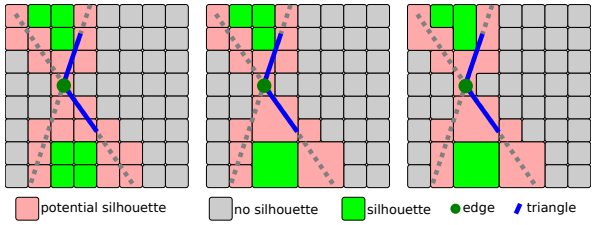


Figure 5: Overview of the proposed approach in 2D. The image shows voxels for one edge. The left side of the images shows the first step of the voxel building algorithm. Voxels are classified into 3 categories – no silhouette, silhouette and potentially silhouette. The next step is to propagate this classification into higher levels (middle image). The right image shows the improvement of compression stage of building algorithm. The octree is transformed into a tree with nodes containing many different subsets of edges defined by bitmasks.

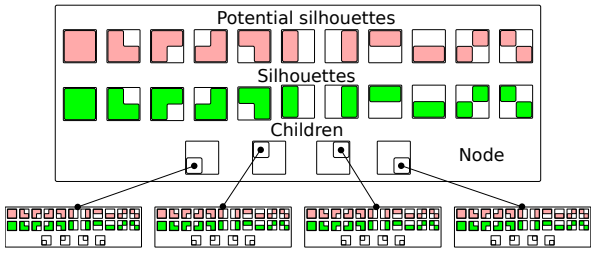


Figure 6: Node data in 2D space for the 8-bit compression. One node contains sets of silhouette and potentially silhouette edges, each addressed by its bitmask value. If a set shape does not intersect the triangle planes of an edge, the edge is stored into the set. The largest set shape is chosen if multiple set shapes do not intersect the triangle planes. A node also contains pointers to child nodes.

of the octree, as seen in Figure 5. If any plane constructed from the triangles adjacent to edge E intersects the voxel, E is considered a **PE**. If none of the triangle planes intersects the voxel and multiplicity of E is nonzero, it is stored among **SE** (set of silhouette edges). The multiplicity can be computed against any point inside the voxel because the whole voxel lies within one of the four subspaces, as demonstrated in Figure 2.

The next step is to propagate **PE** and **SE** into higher levels of the octree. An edge can be propagated to

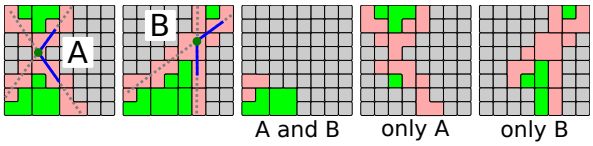


Figure 7: The first two images show two edges – *A* and *B*. Each edge partitions all voxels into voxel shapes for silhouette case and of potential silhouette case. If some voxel shapes are the same for both edges, the edge subsets of those voxel shapes contain both edges (middle image). Otherwise, voxel shapes contain only one edge.

the parent node if it is contained in all of its children. Both types of edges are propagated. The propagation process already significantly reduces the memory footprint. We refer to this propagation scheme as "basic compression".

The last optional step in octree construction is advanced compression. It extends the propagation step by allowing edges to be moved to their parent node even if not all of them are contained in all of its children. These sets of edges are marked with bitmasks corresponding to voxel shapes, see Figure 6. Every subvoxel in these voxel shapes contains the same set of edges, see Figure 7. We call this extended propagation "8-bit compression" as we propagate the edges from children to parent and the bitmask is 8-bit integer. Edges can also be propagated into grandparents (from 64 sibling voxels) which can further improve the compression ratio. This compression scheme is referred to as "64-bit compression". Octree node data are shown in Figure 6.

3.3 Traversal

The traversal part of the algorithm has to copy **SE** and **PE** subsets from the octree into two continuous buffers. The light position determines which subsets of edges have to be copied to the linear buffers, see Figure 8.

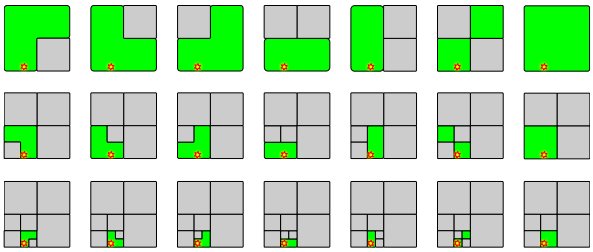


Figure 8: 2D illustration of all edge subsets that contain silhouette edges for a given light position. The hierarchy level is 3. The union of all subsets forms the set of all precomputed silhouette edges. Similar subsets are selected for all potentially silhouette edges. Note that some subsets could be empty. A single edge is contained only in one of the subsets (the largest possible).

The **PE** linear buffer is in the final part of the brute force silhouette extraction process. However, the **PE** set is very small compared to the set of all edges which leads to performance improvement.

4 IMPLEMENTATION

We implemented the whole process both on CPU and an GPU (OpenGL). The construction process relies on two compute shaders: the first one is used to load the data to the octree, the second one for propagation to upper levels.

Hypothetically, every voxel of the octree could contain more than 50 % of all the edge indices in both **PE** and **SE**. Provided we use Crytek's Sponza model as the reference which breaks down to 431 246 edges with maximum multiplicity of 2, octree maximum depth of 5 ($8^5 = 32\,768$ leaf voxels) and indices stored as 32-bit integers, we may end up, in theory, with more than 50 GB of memory. For larger models, adding edges to the lowest level of the octree runs in batches. The batch size is limited by the GPU memory size.

Usually, not all edges get stored neither in **PE** or **SE** buffer of a voxel, thus their size can be limited to a percentage of total edge count (we found that a factor of 0.8 works for most cases and can be seen for example in Table 1). This increases the batch size and speeds up the building process.

Data are then copied back to the system memory. Before the edge propagation, which is also implemented in a compute shader, the algorithm needs to sort the edges, which is carried out on the CPU in parallel.

We first tried the compression as a multi-core post-processing of the octree, but such implementation, although parallelized, was $20-170\times$ slower than 8-bit compression, based on particular scene. For the 8-bit scenario, we moved the advanced compression to the compute shader which tests the edges against octree voxels, because it is in this very step that the bitmask is already known. However, porting the 64-bit compression scheme to GPU seemed problematic as the potential number of sub-voxels is 2^{64}, which would lead to excessive memory footprint, thus we perform the 64-bit compression as CPU post-processing. Due to implementation reasons, compressed nodes using bitmasks other than all-bits-set are located only in `max_depth-1` for 8-bit compression or `max_depth-2` for 64-bit.

During traversal, the algorithm first determines the (X,Y,Z) voxel coordinates within the octree from the light position (flooring the floating point coordinates), which are then converted to linear voxel index. If a light source lies on the boundary between two or more voxels, only one voxel is selected according to eq. (1) where l is the light position and A and B are minimal and maximal corners of the voxel:

$$l_x \in [A_x, B_x) \wedge l_y \in [A_y, B_y) \wedge l_z \in [A_z, B_z). \qquad (1)$$

The traversal process depends on the compression level. For non-compressed and 8-bit compression scenarios, we first traverse the octree and compute an exclusive scan of sizes of all necessary sub-buffers, which serves as the input to the second stage that performs the actual data copy from selected subsets to two linear buffers, one for **PE**, the other for **SE**. Traversal for 64-bit compression splits the pre-processing and prefix scan into two steps as the amount of sub-voxels increases to several thousands. Splitting the stage also helps reducing the count of the global memory reads.

5 RESULTS

Evaluation took place on the following test setup: Intel Core i5 6500, 16GB of DDR4, nVidia GeForce RTX 2080Ti (11GB of GDDR6, driver version 419.17), Windows 10 Pro x64. The test application was built using Visual Studio 2015 x64.

5.1 Build and Compression Tests

We conducted comprehensive build tests on the Šibenik, Conference, and Sponza scenes. The aim was to evaluate the building time and the size of the octree structure, based on the octree deepest level, size of the voxelization area, and compression type.

Tables 1, 2, and 3 show the performance evaluation of the building process under various octree settings, with respect to the selected light source position inside the scene's bounding box. We compared 3 types of the build – with basic compression only (first two coloured columns), 8-bit GPU compression (c8) and 64-bit CPU compression (c64).

One of the first things the reader may notice is that 8-bit compression on GPU performs actually faster than the non-compressed version of the algorithm. This is due to fact that GPU compression occurs in the very first stage of the algorithm thus the following stages have to process a smaller amount of data. However, the 64-bit compression is performed as a postprocessing step and it happens on the CPU, thus being very slow, even though the algorithm was written using OpenMP. We tested the 64-bit compression also on the AMD ThreadRipper system with 24 cores, which improved the 64-bit compression build time by around 60 %, but other two methods performed significantly slower, probably due to different architectures of the two processors.

It can be seen that the amount of memory required to store the octree increases with a factor of 4 with each octree level, but also the average amount of extracted SE increases by around 10 % and the average number of PE that needs to be tested is almost halved with increasing octree depth, for each of the tested scenes.

This test, however, shows the biggest weakness of the algorithm, the memory consumption which, for a particular model, is strongly dependant on the algorithm's settings. For practical use, the 8-bit compression scheme seems to be the the best choice, in terms of both the building speed and the size of the resulting octree structure.

The memory consuption can be approximated using eq. (2), where S is an approximation of the resulting size of the octree structure in MB, e is the number of edges in millions, d is octree depth and c is compression ratio:

$$S(e,d,c) = e \cdot 8^d \cdot V_d \cdot c \tag{2}$$

Based on the results in Tables 1, 2, and 3, we estimated the average compression ratios to 0.32 for 8-bit compression scheme and 0.11 for 64-bit scheme by dividing the compressed octree size with non-compressed octree. Values V_d define the approximate size of a single voxel per 1 million edges. These values were calculated as $V_d(d,e) = S_m/8^d/e$, where S_m is the measured size of non-compressed octree. Values obtained by this equation are $V_3 = 0.93$, $V_4 = 0.53$ and $V_5 = 0.30$.

The average relative deviation of eq. (2) is 6 %.

5.2 Silhouette Extraction Tests

We compared our new method (with 8-bit compression) to a brute-force compute shader implementation of silhouette extraction, based on an OpenCL implementation and multiplicity theorem described in [Peč13]. Both methods output edge indices as their result. For this test, we compiled 26 models in total, mixing popular models (Sponza, Šibenik, Buddha, Conference, Gallery, Bunny[1]) with two types of synthetic scenes that we created – the first type were scenes consisting of uniform grid of increasing amount of spheres, having 33 750 to 1 574 640 edges. The second type consisted of randomly positioned spheres differing in numbers, having 124 200 to 933 120 edges. We evaluated our algorithm with two levels of octree depth (3 and 5) posing as best and worst case, and scene scales 1, 2, 4, 8, and 16.

In a single test run, we moved the light source through the octree volume in a $10 \times 10 \times 10$ grid, both for our method and the brute-force approach. From each light position we evaluated the traversal time as average of 5 repetitions. In total, we made 75 000 measurements in each scene: 50 000 for our approach and 25 000 for bruteforce method. As mentioned above, we tested 2 octree levels; that is why our method has twice as many measurements per scene.

The result can be seen in Figure 9. Our accelerated approach has reduced the sensitivity of the algorithm to the number of edges, compared to the bruteforce approach. Our method performs always better on models

[1] freely available at https://casual-effects.com/data/

Octree Depth	Scale	Size (MB)	Build (s)	Size c8 (MB)	Build c8 (s)	Size c64 (MB)	Build c64 (s)	Pot Avg	Sil Avg	Pot % Avg	Sil % Avg
3	1	52	0.62	16	0.58	5	11.39	19668	16105	16.76	72.20
	2	57	0.60	18	0.57	5	14.69	21586	15649	18.40	69.91
	4	58	0.61	18	0.57	5	17.12	22088	15514	18.82	69.25
	8	58	0.61	18	0.57	5	17.98	22179	15494	18.90	69.26
	16	58	0.61	18	0.57	6	18.35	22147	15498	18.87	69.19
4	1	235	1.77	77	1.54	27	21.37	10291	18801	8.77	84.28
	2	256	1.75	84	1.55	29	29.13	11359	18531	9.68	82.85
	4	262	1.75	86	1.54	30	32.38	11610	18469	9.89	82.51
	8	263	1.75	86	1.53	30	35.10	11688	18444	9.96	82.37
	16	263	1.74	86	1.53	30	35.32	11690	18448	9.96	82.42
5	1	1022	7.67	341	6.82	127	62.17	5304	20405	4.52	91.52
	2	1122	7.71	376	6.78	139	76.00	5876	20266	5.01	90.62
	4	1147	7.78	385	6.78	141	81.60	6008	20246	5.12	90.45
	8	1155	7.69	388	6.79	143	83.72	6038	20236	5.15	90.41
	16	1155	7.75	388	6.77	142	86.69	6051	20237	5.16	90.39

Table 1: Build test of Sibenik scene, consisting of 117 342 edges. We evaluated the build times and resulting octree size under various voxel sizes and scales. The 3rd and 4th columns contain results for octree build with basic compression scheme. Columns tagged "c8" and "c64" show build times and sizes when using 8-bit or 64-bit advanced compression schemes. The numbers in "Pot Avg" and "Sil Avg" columns show the average number of PE and SE acquired during octree traversal, tested from each lowest level voxel. The second column from the last tells the average amount of edges from the full edge count that needs to be tested, the last column describes the average amount of SE acquired from octree as the percentage of all silhouette edges observed from light position in the middle of each lowest level voxel.

Octree Depth	Scale	Size (MB)	Build (s)	Size c8 (MB)	Build c8 (s)	Size c64 (MB)	Build c64 (s)	Pot Avg	Sil Avg	Pot % Avg	Sil % Avg
3	1	80	0.84	25	0.79	8	34.95	34338	19426	17.61	65.87
	2	93	0.84	29	0.78	9	50.55	39952	18481	20.49	62.37
	4	96	0.84	30	0.79	10	59.29	41323	18288	21.19	61.56
	8	97	0.84	31	0.78	10	62.89	41656	18265	21.36	61.46
	16	97	0.84	31	0.78	11	64.99	41794	18243	21.43	61.32
4	1	379	2.55	121	2.54	42	69.81	18675	23055	9.58	78.19
	2	431	2.64	139	2.53	48	106.65	21857	22317	11.21	75.35
	4	443	2.62	144	2.52	50	124.19	22595	22187	11.59	74.71
	8	446	2.64	145	2.53	51	134.23	22783	22154	11.68	74.54
	16	447	2.65	146	2.54	52	136.04	22832	22149	11.71	74.51
5	1	1786	10.66	581	8.80	209	150.24	9894	25738	5.07	87.29
	2	2044	11.49	668	8.92	238	222.80	11698	25234	6.00	85.18
	4	2102	11.12	687	8.61	245	265.10	12123	25151	6.22	84.69
	8	2120	11.23	694	8.81	248	284.60	12219	25141	6.27	84.58
	16	2121	11.28	694	8.96	249	291.80	12241	25135	6.28	84.56

Table 2: Build test of Conference scene, consisting of 195 019 edges. Check table 1 for column description.

Octree Depth	Scale	Size (MB)	Build (s)	Size c8 (MB)	Build c8 (s)	Size c64 (MB)	Build c64 (s)	Pot Avg	Sil Avg	Pot % Avg	Sil % Avg
3	1	192	1.56	60	1.40	19	270.59	83074	31640	19.26	58.98
	2	202	1.57	63	1.38	20	332.78	86744	30845	20.11	57.30
	4	205	1.59	65	1.37	20	361.90	87829	30661	20.37	57.08
	8	206	1.62	65	1.37	21	373.43	88157	30644	20.44	57.08
	16	206	1.60	66	1.38	21	379.24	88382	30618	20.49	57.02
4	1	893	5.31	289	5.10	96	586.34	44817	39567	10.39	73.72
	2	930	5.36	302	4.62	101	705.98	47044	39029	10.91	72.58
	4	943	5.40	306	4.62	102	783.30	47414	38933	10.99	72.34
	8	946	5.42	306	4.64	103	819.03	47559	38906	11.03	72.34
	16	948	5.41	307	4.61	103	837.29	47636	38899	11.05	72.34
5	1	4111	21.17	1359	15.45	498	1210.22	23895	45123	5.54	84.18
	2	4305	21.37	1425	15.90	504	1517.91	25094	44867	5.82	83.47
	4	4345	21.20	1438	16.18	522	1635.41	25321	44782	5.87	83.34
	8	4351	21.25	1441	16.36	511	1735.49	25374	44819	5.88	83.30
	16	4357	21.13	1443	16.26	519	1789.33	25376	44782	5.88	83.31

Table 3: Build test of Sponza scene, consisting of 431 246 edges. Check Table 1 for column description.

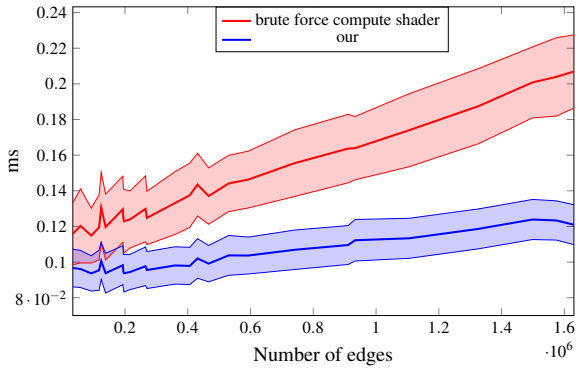


Figure 9: Average extraction times for compilation of 26 scenes (sorted by the number of edges). Red line represents bruteforce compute shader method, blue line represents our new proposed method. The area around the lines represents (+-) mean absolute deviation.

Compression	Average (ms)	Max Abs Deviation (ms)
basic	0.098	0.011
8-bit	0.102	0.012
64-bit	0.134	0.013

Table 4: Comparison between compression levels on Sponza scene. Average octree traversal time calculated from 1000 different light positions in the scene and maximum absolute deviation from the average.

having more than 200 000 edges and it is also more stable. Its average absolute variance is almost half, compared to the bruteforce method.

We also evaluated the performance difference when using different compression ratios. We used the Sponza model and moved the light source around in the same way as described in the previous test. The results can be seen in Table 4. The complexity of the 64-bit compression traversal outweighs its benefits in the form of lower size, thus the 8-bit compression scheme seems to be the best choice.

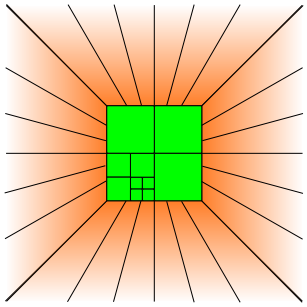


Figure 10: One of the possible extensions to the algorithm. Instead of using one hierarchical structure, the algorithm would use two – one for the close vicinity of the model and one for all the other space around. The green part shows the hierarchical structure as presented, the orange parts is the second hierarchical structure. The second structure uses angles instead of voxels.

We can estimate the extraction time for brute force approach as $t_b = E \cdot K$ where E is the number of edges and K is extraction complexity. Our method yields $t_t = P \cdot E \cdot K + T$ where T is traversal cost and P is the ratio of potential edges, which can be seen in 2nd last column of Tables 1 - 3. Based on the build test results, we estimated the P to be 0.2, 0.1 and 0.05 for octree levels 3, 4 and 5. According to our measurements, T was 0.075 ms in average and was not dependant on the number of edges.

6 CONCLUSION

We presented a novel approach to accelerated silhouette extraction by storing pre-computed PVS in an octree, as well as novel octree compression schemes.

The majority of the building process was implemented on GPU using OpenGL and compute shaders. The building process is reasonably fast when using 8-bit compression scheme, as it processes less data then the basic compression scheme. The resulting octree can be stored in a file to avoid repetitive builds in subsequent runs. Experimentally, we were able to reduce the octree size using 64-bit compression to around 12 % of basic compression scheme, but the compression itself was used as a post-processing step on CPU, thus not performing as fast as the GPU implementation. In terms of performance, 64-bit compression also lagged behind due to having a more complicated traversal. The 8-bit compression scheme provides the best results in terms of octree size and traversal speed.

Compared to the brute-force approach, our method is less sensitive to the number of edges. It was also more stable, in terms of maximal absolute deviation. The biggest drawback of the method is its memory consumption and also spatial limitation due to the nature of voxelization. The method also would not work well on scenes with dynamic geometry (f.e. morphing).

This method could be further improved by storing triangle indices instead of edges, which, in theory, could reduce memory footprint of the method even more. The whole structure does not need to reside on the GPU but can be streamed as needed. Future research could also evaluate usage of homogeneous coordinates, which may create hierarchy with unlimited spatial span, see Figure 10.

7 ACKNOWLEDGMENTS

The work was supported by the Ministry of Education, Youth and Sports, Czech Republic, V3C (Visual Computing Competence Centre) TE01020415 research program and Technology Agency of the Czech Republic.

8 REFERENCES

[Mil14] Milet, T., Kobrtek, J., Zemčík, P, Pečiva, J. Fast and Robust Tessellation-Based Silhouette Shadows. In WSCG 2014 - Poster papers proceedings, 2014.

[Peč13] Pečiva, J., Starka, T., Milet, T., Kobrtek and Zemčík P., Robust Silhouette Shadow Volumes on Contemporary Hardware. In Conference Proceedings of GraphiCon'2013, pp 56–59, 2013.

[Wil78] Williams, L., Casting curved shadows on curved surfaces. In SIGGRAPH Comput. Graph., pp 270–274, 1978.

[Jon01] Johnson, D. E. and Cohen, E., Spatialized Normal Cone Hierarchies. In SI3D '01, pp 129–134, 2001.

[Crow77] Crow, F. C., Shadow algorithms for computer graphics. In SIGGRAPH '77, pp. 242–248, 1977.

[Woo12] Woo, A. and Poulin, P., Shadow Algorithms Data Miner. CRC Press, ISBN 978-1-4398-8023-4, 2012.

[Mil15] Milet, T., Tóth, M., Pečiva, J., Starka, T., Kobrtek, J. and Zemčík, P., Fast robust and precise shadow algorithm for WebGL 1.0 platform, In ICAT-EGVE 2015, pp 85–92, 2015.

[Ols06] Olson, M., Zhang, H., Silhouette Extraction in Hough Space, In Computer Graphics Forum 25(3), pp 273–282, 2006.

[Sti07] Stich, M, Wächter, C, Keller, A., Efficient and Robust Shadow Volumes Using Hierarchical Occlusion Culling and Geometry Shaders, In Nguyen, R. (Ed.) GPU Gems 3, ISBN 978-0321515261, pp 359–378, 2007.

[Bren02] Brennan, C., Shadow Volume Extrusion using a Vertex Shader, In Engel, W. E. (Ed.) ShaderX, pp 188-194, 2002.

[Hei91] Heidmann, T., Real shadow real time, In Iris Universe 18, pp 23–31, 1991.

[Eve02] Everitt, C. W., Kilgard, M. J., Practical and Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering, In CoRR, 2002.

[Bil99] Bilodeau, B. and Songy, M., Real time shadows, Creativity, 1999.

[Ise03] Isenberg, T., Freudenberg, B., Halper, N., Schlechtweg, S., Strothotte, T., A developer's guide to silhouette algorithms for polygonal models, In IEEE Computer Graphics and Applications, vol. 23, no. 4, pp. 28-37, July-Aug. 2003.

[Kim08] Kim, B., Kim, K., Turk, G., A Shadow Volume Algorithm for Opaque and Transparent Non-Manifold Casters, In Journal of Graphics Tools 13, pp 1–14, 2008.

[Goo99] Gooch, B., et al., Interactive Technical Illustration, In Interactive 3D Graphics, pp. 31–38, 1999.

[Ben99] Benichou, F. and Elber, G., Output Sensitive Extraction of Silhouettes from Polygonal Geometry, In Proc. 7th Pacific Graphics Conf., pp. 60–69, 1999.

[Ger15] J. Gerhards, J., Mora, F., Aveneau, L., Ghazanfarpour, D., Partitioned Shadow Volumes, In Computer Graphics Forum, volume 34 issue 2, pp 549–559, 2015

[Air90] Airey, John M. and Rohlf, John H. and Brooks,Jr., Frederick P., Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments, SIGGRAPH Comput. Graph., volume 24, 1990

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.27, No.1, 2019

Evolutionary Generation of Primitive-Based Mesh Abstractions

Markus Friedrich, Felip Guimerà Cuevas, Andreas Sedlmeier, André Ebert
Institute for Computer Science
LMU Munich
Oettingenstr. 67
80538 Munich, Germany
{markus.friedrich|andreas.sedlmeier|andre.ebert}@ifi.lmu.de, felip.guimera@campus.lmu.de

ABSTRACT

The procedural generation of data sets for empirical algorithm validation and deep learning tasks in the area of primitive-based geometry is cumbersome and time-consuming while ready-to-use data sets are rare. We propose a new and highly flexible framework based on Evolutionary Computing that is able to create primitive-based abstractions of existing triangle meshes favoring fast running times and high geometric variation over reconstruction precision. These abstractions are represented as CSG trees to widen the scope of possible applications. As part of the evaluation, we show how we successfully used the generator to create a data set for the evaluation of neural point cloud segmentation pipelines and additionally explain how to use the system to create artistic abstractions of meshes provided by publicly available triangle mesh databases.

Keywords

Evolutionary Algorithms, Geometry Processing, CAD, CSG, Deep Learning

1 INTRODUCTION

A plethora of empirical algorithm validation and deep learning tasks in the field of primitive-based 3D geometry processing require a diverse and sufficiently large set of 3D models as test or training input. For models represented as triangle meshes, these data sets exist and are available for free (e.g. ShapeNet [CFG+15], ModelNet [WSK+14], ABC [KMJ+18], etc.). However, if model representations based on a composition of geometric primitives (eg. spheres, cylinders, cuboids, etc. combined by Boolean set operations) are needed, data sets are rare. For example, the ABC data set contains 1000.000+ Computer Aided Design (CAD) models with primitive information but lacks cuboids as one of the considered primitive types and also does not account for compositional information like the arrangement of Boolean operators in a CSG tree.

An example use case where such data sets are needed would be a CSG tree detection pipeline based on neural networks (see [SGL+18] for example). It requires a 3D point cloud as input and delivers a CSG tree that fits the 3D point cloud best, together with the parameters of detected primitives. Traditionally, deep learning tasks need huge training sets, which are in this case hard to find or cumbersome to generate manually with off-the-shelf CAD tools.

In order to fill this gap, we propose a CSG tree generator framework based on Evolutionary Computing that transforms a triangle mesh model together with a set of constraints (e.g. frequency distribution of primitive types) into a CSG tree representation which combines a set of fitted primitives with Boolean set operations (e.g. union, intersection, difference). Note that the primary goal here lies not in finding the CSG tree which matches the input geometry as perfectly as possible but in generating a sufficiently accurate abstraction, allowing for the generation of tens of thousands of models within an acceptable time frame on currently available hardware. Another, completely different use case worth to consider is the artistic abstraction of geometry for visually appealing renderings and animations appearing in entertainment products and multimedia installations.

The proposed processing pipeline starts with sampling the input model, resulting in a 3D point cloud that is then clustered for better computational efficiency. For each cluster, primitives are fitted. Which primitive types to use for fitting is determined by sampling a user-defined frequency distribution that specifies the desired distribution of primitive types in the resulting CSG tree. Then, per-cluster CSG trees are extracted using a specific variant of an Evolutionary Algorithm (EA). Finally, resulting CSG trees are merged to a combined result.

In summary, this paper presents the following main contributions:

- A highly flexible and configurable framework for the generation of primitive-based mesh abstractions that are represented as CSG trees.

- An evaluation of three neural network architectures considering the task of primitive detection from 3D point clouds. Necessary training, validation and test data sets were generated with our proposed framework.

2 BACKGROUND

2.1 CSG Trees and Signed Distance Functions

A CSG tree represents a 3D model as a hierarchical combination of Boolean set operations and primitives (e.g. cubes, spheres, cylinders, ...). Set operations are thereby inner nodes of the tree whereas primitives are always leaves. In our case, a primitive p is described by a signed distance function f_p, where the surface of p is the zero set of f_p:$\{\mathrm{x} \in \mathbb{R}^3 : f_p(\mathrm{x}) = 0\}$.
The Boolean set operations are represented using min- and max-functions [Ric73]:

- Intersection: $p_1 \cap p_2 := \max(f_{p_1}, f_{p_2})$
- Union: $p_1 \cup p_2 := \min(f_{p_1}, f_{p_2})$
- Complement: $\overline{p} := -f_p$
- Subtraction: $p_1 \setminus p_2 := p_1 \cap \overline{p_2}$

The surface normal for a certain point $\mathrm{x} \in \mathbb{R}^3$ can be retrieved by $\nabla f_p(\mathrm{x})$.

2.2 Genetic Algorithms

A Genetic Algorithm is a population-based metaheuristic for solving optimization problems and belongs to the class of Evolutionary Algorithms. The concept is inspired by the biological phenomenon of natural selection. Initially, a randomly generated population of possible solutions is created and ranked using a problem-specific objective function. In the following iteration, the best solutions from the last iteration are selected and changed using domain-dependent modification operators (mutation and crossover). This procedure is repeated until a certain stop criterion is met (e.g. a certain objective function value or maximum iteration count has been reached). The extraction of a CSG tree from a set of fitted primitives and a shape-describing point cloud is a combinatorial optimization problem (see e.g. [FFPF18]) which we solve using a Genetic Algorithm.

3 RELATED WORK

3.1 Available Data Sets

Large model databases of triangle meshes exist (ShapeNet [CFG+15], ModelNet [WSK+14]) but do not describe distinct primitives. The data set introduced in [KMJ+18] contains primitives but does not include cuboids and CSG tree descriptions. For an exhaustive overview of available data sets, see [KMJ+18].

3.2 Procedural Model Fitting & Modeling

Procedural Model Fitting (PMF) describes the task of finding a geometric representation that fits a certain input data set (e.g. a point cloud) as precisely as possible. The primitive generator proposed by Zou et al. [ZYY+] takes point clouds as input and uses a variant of the Iterative Closest Point method (ICP) [BM92] to fit cuboids but is restricted to that single primitive type. A constrained-based PMF technique employing a Genetic Algorithm is proposed in [HSS17]. While results look promising, all models are represented using triangle meshes, which is not suitable for our use case. Other approaches use Sequential [RMGH15] or Markov Chain Monte Carlo [TLL+11] methods as well as Reinforcement Learning [TKS+13, SGL+18]. Our approach is different in that we accept arbitrary 3D meshes (not just point clouds) as input and focus on generation speed rather than precise fitting.
A related research field is Procedural Modeling (PM). There, visual content (3D models, textures, ...) is generated based on specialized algorithms with user-controlled parameters. In recent years, there has been vivid research activity in the field of procedural content generation using Machine Learning approaches (PCGML). See [SSG+18] for a comprehensive survey. For a survey on the procedural generation of complete worlds (landscapes, buildings, creatures, ...), see [FE17].

4 PROBLEM STATEMENT

The problem that is solved by our proposed generator can be described as follows: Given a 3D model represented as a closed triangle mesh and a frequency distribution of primitive types initially defined by the user, generate a CSG tree which matches the input mesh as closely as possible while the set of primitives corresponds to the selected distribution. Important to note is that the computational effort of the generation process should be kept low in order to allow for the creation of large model data sets (> 10.000 objects) in a reasonable time frame. Speed is therefore more important than visual quality.

5 CONCEPT

The proposed pipeline as depicted in Figure 1 starts with a closed 3D triangle mesh together with a user-defined primitive frequency distribution H_p as input and

results in a CSG tree together with a set of primitives and their parameters as output. Each of the following sub sections is dedicated to a particular pipeline step.

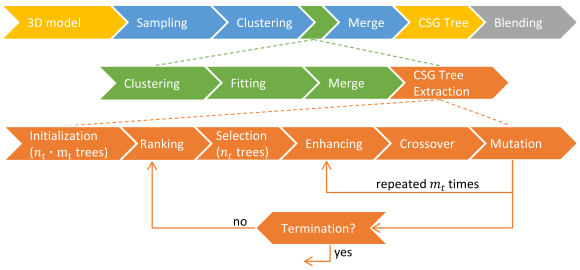


Figure 1: Overview of the CSG tree extraction pipeline. Parameters n_t and m_t are explained in Section 5.4.

5.1 Sampling

In this step, the input mesh is sampled resulting in a 3D point cloud S. The sample points are later used to measure how well a CSG tree matches the mesh's shape. Each point in S receives a label that indicates whether the point is located inside, outside or on the surface of the mesh. See Figure 2 for an example.

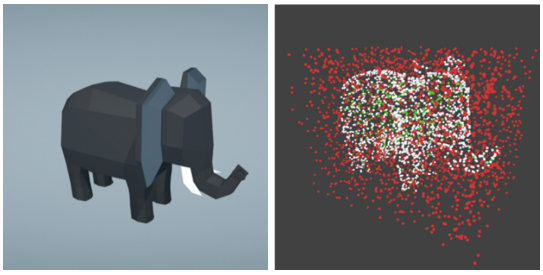


Figure 2: Elephant mesh (left) and corresponding sampling points with inside points in green, outside points in red and on-surface points in white (right).

The points and their corresponding assigned labels are retrieved using a special raycasting approach, in which rays do not detect meshes for which the origin of the raycast lies inside the mesh: First, random points within the bounding box of the mesh are selected. These points serve as origins for rays that are cast in random directions. If a ray hits the mesh, the hit point p_{hit} is added to S with an "on-surface" label. If no mesh was hit during the raycast, then any point along that cast outside the bounding box can be marked as p_{hit} with an "outside" label. In addition, the ray's origin p_{org} is added to S. Its corresponding label ("inside", "outside") is determined by casting a second ray back from p_{hit} through p_{org} and comparing the lengths of both rays. If the first ray is longer, then the label "inside" is assigned, if it is shorter, the label "outside". In case of equal ray lengths the point is marked as "on-surface. If p_{hit} and p_{org} are the same, the origin is not added to S. The label of p_{org} is therefore always uniform regardless of the initial direction of the raycast.

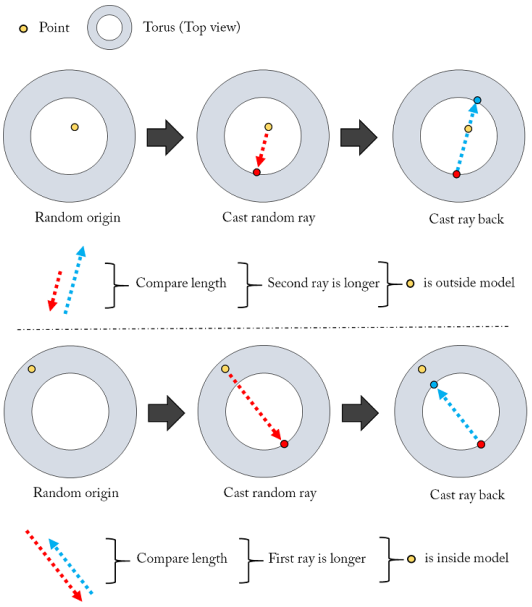


Figure 3: Explanation of the employed raycasting approach for label assignment by example: A torus is used as mesh geometry. The two cases for an "inside" (bottom) and an "outside" labeled point (top) are depicted. The label "on-surface" is assigned in case of equal ray lengths.

5.2 Clustering

In order to reduce the computational complexity of primitive fitting and CSG tree extraction, a two-level clustering approach is applied to the point cloud S ("inside", "outside" and "on-surface" points are considered). First, it is clustered into a set of n_c clusters C using the k-means algorithm [Llo82]. Then, each cluster is again clustered into m_c sub-clusters using the same technique. The method results in a set of $n_c \cdot m_c$ sub-clusters C_{sub}, which is then the basis for primitive fitting. Please note that m_c and n_c are user-controlled parameters.

5.3 Primitive Fitting

For each sub-cluster c_{sub} in C_{sub}, a single primitive is fitted. The primitive type is determined by sampling the user-defined primitive frequency distribution H_p, which assigns a probability to each supported primitive type (sphere, box, cylinder, torus, cone, extruded pentagon and triangle). This results in the primitive set P_{sub}. See Algorithm 1 for all details.
Important to note is that the primitive fitting method does not strive for high matching precision but for fast running times and high geometric variation.

5.4 Per-Cluster CSG Tree Extraction

For each cluster c_i in C, all sub-clusters are merged resulting in a set of primitives P_i and points S_i. The CSG tree extraction process is conducted for each cluster in

Algorithm 1: The primitive fitting algorithm. The method createP$(\cdot)$ generates a primitive instance based on a primitive type (e.g. sphere or box), a center position and a size value. The size value for a sphere would be its radius. For primitives with more than a single size dimension (e.g. boxes), each dimension is set to the size value.

input : Set of sub-clusters C_{sub}, primitive frequency distribution H_p
output: Fitted primitive for each sub-cluster $\in C_{sub}$

$P_{sub} \leftarrow \{\}$
foreach $c_{sub} \in C_{sub}$ **do**
  $d_{min} \leftarrow$ minimum of the largest distances per axes in c_{sub}
  $d_{max} \leftarrow \text{diameter}(c_{sub})$
  $\text{size}_p \leftarrow \text{random}(d_{min}, d_{max} \cdot 0.5)$
  $\text{center}_p \leftarrow \text{center}(c_{sub})$
  $\text{type}_p \sim H_p$
  $P_{sub} \leftarrow P_{sub} \cup \text{createP}(\text{type}_p, \text{center}_p, \text{size}_p)$
return P_{sub}

C and uses a Genetic Algorithm to solve the CSG tree extraction problem.

Initialization. The GA initializes the CSG tree population with randomly generated CSG trees that use primitives from P_i. This is done exactly once while the following steps are executed repeatedly.

Ranking. All CSG trees in a population are ranked using the objective function

$$E(t) = \sigma(t) \cdot \sum_{j=1}^{|S_i|} \begin{cases} f_t(s_{ij}) & \text{l}(s_{ij}) = \text{"on-surf."} \\ |\min(f_t(s_{ij}),0)| & \text{l}(s_{ij}) = \text{"outside"} \\ \max(f_t(s_{ij}),0) & \text{l}(s_{ij}) = \text{"inside"} \end{cases}, \quad (1)$$

where $\sigma(t) = log_2(\text{size}(t))$ is a tree size penalty term, $f_t(\cdot)$ is the signed distance function of tree t and $l(\cdot)$ assigns a label $\in \{\text{"on-surface"}, \text{"outside"}, \text{"inside"}\}$ to each point in S_i. The GA-based CSG tree extraction aims for minimizing Equation 1. See Figure 4 for getting an intuition of the objective function.

Selection. After the whole population was ranked and sorted in ascending order, the best (with respect to their objective function value as defined in Equation 1) n_t CSG trees are selected for the steps "Enhancing", "Crossover" and "Mutation" (collectively referred to as variation steps in the following). Variation steps are applied to each of the n_t selected CSG trees m_t times, resulting in a constant population size of $n_t \cdot m_t$ CSG trees (see Figure 1).

Enhancing. The idea of the enhancement operator is to improve the geometry score before the actual classic variation operators (mutation, crossover) are applied. It can be seen as a local hill-climbing strategy for faster convergence.

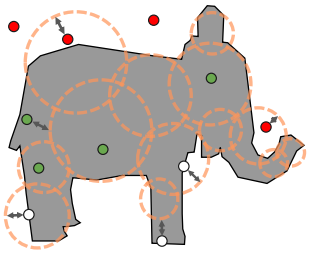


Figure 4: The intuition behind the objective function (CSG tree primitives in orange dotted lines, input mesh in grey). The objective function measures the accumulated absolute distance between the surface induced by the CSG tree t and inside (green), outside (red) and on-surface sampling points (white). Only those distances from incorrectly classified points are added. This is the case for points that are labeled as "outside" but are located inside the CSG tree and vice-versa. Distances to on-surface points are always added (added distances are indicated by grey arrows).

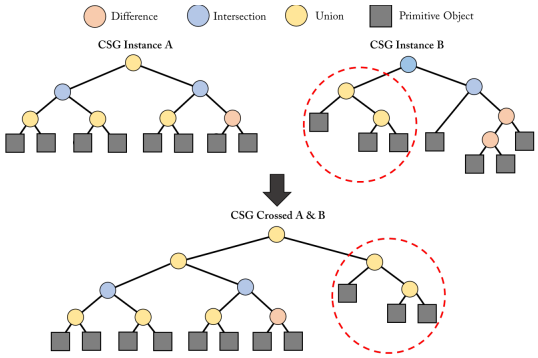
It works as follows: Each CSG tree in the population stores the sample point p_w with the worst objective function value. The enhancement operator modifies the CSG tree in the following way: If $l(p_w) =$ "outside", i.e., the sample point p_w should be outside, but -as it has a bad objective value- is wrongfully placed inside, a randomly generated primitive is cut out by adding it to the CSG tree together with a difference operation. If $l(p_w) =$ "inside", then a randomly generated primitive is added to the CSG tree together with a union operation. In case of $l(p_w) =$ "on surface", a randomly generated primitive is either cut out or added to the solid induced by the CSG tree. In this case, operation choice (cut out or add) is random with both operations having a probability of 0.5.

Crossover & Mutation. The currently used crossover and mutation operators for a particular CSG tree are chosen randomly and applied to the CSG tree m_t times together with the enhancement operator. Each combined enhancing, crossover and mutation operation results in a single new CSG tree. See Figure 5 and Figure 6 for an overview and description of used crossover and mutation operators.

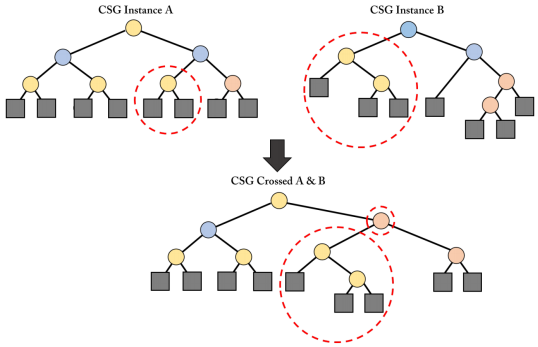
Termination. In order to determine whether or not the GA should terminate, the average score of improvement δ_E for GA iteration k

$$\delta_E^k = \frac{1}{n_b}\left[\min_{0 \le l < k}\left(\sum_{t \in T_l} E(t)\right) - \sum_{t \in T_k} E(t)\right] \quad (2)$$

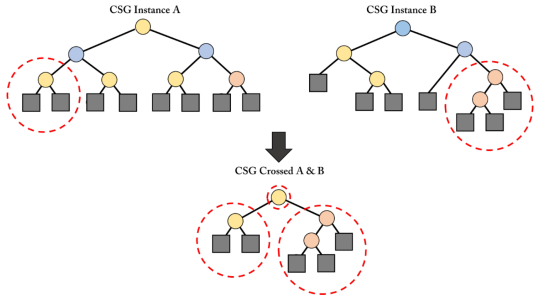
is evaluated, where T_k is the set of the n_b best CSG trees in the population of iteration k and $\min_{0 \le l < k}(\cdot)$ is the best accumulated score reached so far in that particular cluster. δ_E^k is then compared to a pre-defined average score threshold δ_E^{ts} which is multiplied by the best average score reached. This checks whether δ_E^k has improved by a certain amount δ_E^{ts} compared to the current



(a) Crossover 1: Merges the first tree with a randomly chosen sub-tree from the second tree.



(b) Crossover 2: Replaces a proper, randomly chosen sub-tree from one tree with a randomly chosen, not necessarily proper sub-tree (the sub-tree could also be the entire CSG tree), from the second tree and adds a randomly chosen operation (union, intersection or difference) to the parent of the sub-tree of the first tree.



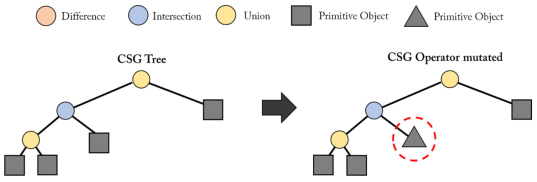
(c) Crossover 3: Selects a sub-tree randomly from both trees and combines them arbitrarily using the union, intersection or difference operator.

Figure 5: All used crossover operators.

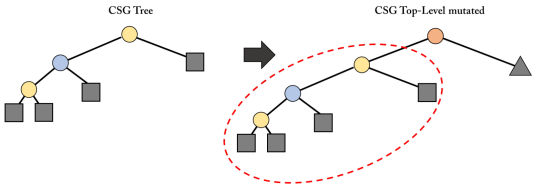
best average score. If this is not the case, an iteration counter is incremented. If the number of iterations surpasses a user-controlled upper bound, the algorithm terminates.

5.5 Merge

The result of the steps described in Section 5.4 is a CSG tree for each cluster in C. In order to combine these per-cluster trees into a single one representing the complete model, we apply a hierarchical merge scheme based on the nearest neighbor information obtained by the clustering mechanism. See Figure 7 for an explanation of



(a) Mutation 1: Replaces a randomly chosen primitive in the tree with a new, randomly selected primitive.



(b) Mutation 2: Adds a new, randomly selected top-level operation to the tree. The old tree is one child, a randomly created primitive the other one.

(c) Mutation 3: Replaces two primitives connected via union with a single randomly created primitive.

Figure 6: All used mutation operators.

the algorithm by example. The proposed merge proce-

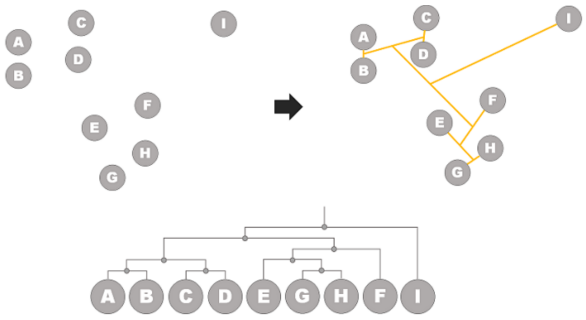


Figure 7: The hierarchical merge process explained by example. Per-cluster trees are combined with close by trees using the union operator. The process is repeated until only a single tree is left.

dure combines per-cluster trees that are close by, which has positive effects on tree editability and tree balance.

5.6 Blending

For certain use cases (e.g. artful model abstraction), additional blending of neighboring per-cluster CSG trees can be applied using the blending operator [Ric73]

$$b(f_{t1}, f_{t2}, x) = f_{t2}(p) \cdot (1-h) + f_{t1}(p) \cdot h - \alpha \cdot h \cdot (1-h), \quad (3)$$

where f_{t1} and f_{t2} are the signed distance functions of the two trees that should be blended together, $h = \min(1, \max(0, 0.5 + \frac{f_{t2}(x) - f_{t1}(x)}{2 \cdot \alpha}))$ and α is a user-defined parameter controlling blending smoothness.

See Figure 15 and 16 for a visualization of different smoothness strengths. Note that a specific α-value is not given since it depends on model size and thus is not generalizable.

6 EVALUATION

The evaluation consists of two parts: The first part describes and explains the performance characteristics of the generator framework and the second part details a possible use case.

Parameter Name	Value
n_t	15
m_t	70
n_b	15
H_p	uniform distribution
δ_E^{ts}	0.2

Table 1: Parameters used throughout the evaluation.

6.1 Generator Framework

The generator framework was evaluated using a machine with an Intel(R) Core(TM) i7 CPU @ 3.06GHz and 12GB of RAM. Experiments were conducted with a varying number of clusters and different sample point cloud sizes with seven 3D models taken from the Google Poly data set [pol] (see Figure 14 for an overview). See Table 1 for a list of parameter values that were used throughout the evaluation and Figure 13 for an exemplary CSG tree result.
For a meaningful quality evaluation, an extra sampling of the triangle mesh's surface with a fixed number of samples is conducted. This point set is then used to evaluate the objective function for a specific CSG tree resulting in its geometry score.

6.1.1 *Number of Clusters*

The impact of the total number of sub-clusters $|C_{sub}| = n_c \cdot m_c$ for a fixed $|S| \approx 2000$ on the running time is shown in Figure 8 ($n_c = 5$, $m_c \in \{1,2,...,7\}$). It is clearly visible that running time and $|C_{sub}|$ have an approximate linear relationship. The number of fitted primitives (which is equal to $|C_{sub}|$) positively affects the quality of the model approximation. As visible in Figure 9, this effect weakens significantly starting from $|C_{sub}| = 15$. This is a good hint for a running time/quality trade-off. Note that geometry scores are not normalized and thus an inter-model comparison is not possible. Figure 15 and 16 show results for models *Elephant* and *Giraffe* for different values of $|C_{sub}|$.

6.1.2 *Point Cloud Size*

We evaluated the impact of the size of the sampling point cloud $|S|$ on the running times and result quality using all seven models and 25 sub-clusters ($n_c = 5$, $m_c = 5$). Results are depicted in Figure 10 and 11. As expected, running times grow linearly with the number of sampling points. The results also show a significant improvement of result quality until a sampling point cloud size of ca. $1000 - 1500$ points. This can be explained by the geometric complexity of the input triangle mesh: There is no positive effect if more samples are used than are required for the representation of the input mesh in all its details.

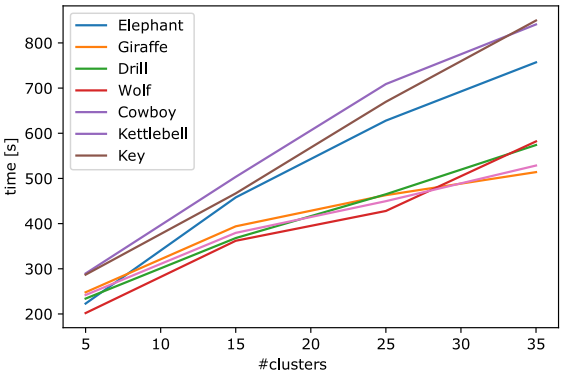


Figure 8: $|C_{sub}|$ and running times.

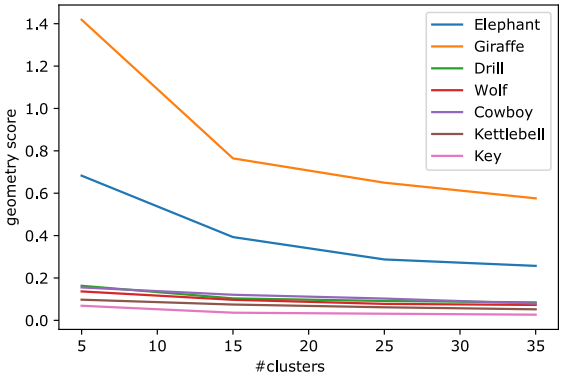


Figure 9: $|C_{sub}|$ and reached geometry score (smaller score means better quality).

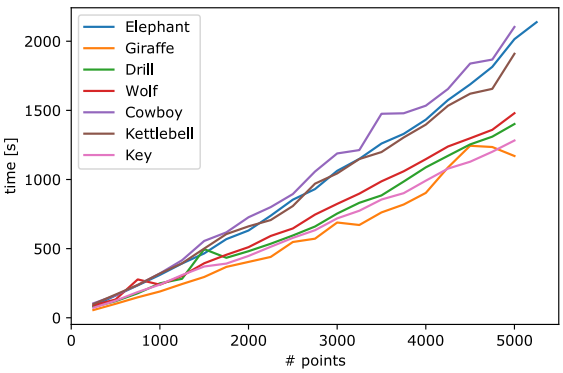


Figure 10: Sample point cloud size and running times.

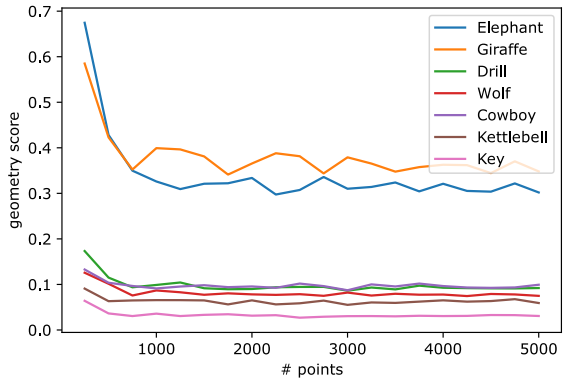


Figure 11: Sample point cloud size on the geometric quality (smaller score means better quality).

6.2 Deep Primitive Segmentation

We tested the usability of our generator framework in an evaluation of three different neural network architectures for point cloud segmentation (PointNet [QSMG16], PointNet++ [QYSG17] and PointCNN [LBSC18]). Our main interest was their performance (Accuracy and Intersection over Union (IoU)) in the task of primitive segmentation (cuboids, spheres, cylinders and cones) as well as their robustness to noise and differences between training and test sets.
The training set consisted of 15.000 point clouds, each representing $20-80$ primitives using 2048 points. In order to evaluate the networks' performance, its robustness and ability to generalize, we used six different test data sets, each containing 500 point clouds:

- **No Rotation**: Generated with the same parameters as the training set.
- **Rotation**: Same as *no rotation* but with additional random rotation.
- **Reduced Points**: Same as *no rotation* but containing only 50% of the points.
- **Noise (low/high)**: Same as *no rotation* but with added low ($\mu = 0, \sigma = 0.5$, *noise low*) and high ($\mu = 0, \sigma = 1.5$, *noise high*) Gaussian noise.
- **More Primitives**: Same as *no rotation* but with an increased minimum primitive count of 40.

The data generation was fully automated and took 5 days on the system described in Section 6.1. Please note that instead of the usual generator output (a CSG tree), a point cloud sampled from it was used here.
Table 2 lists the results. PointNet++ performs best by far on all test data sets which makes it the preferred candidate for further research on enhanced primitive fitting pipelines. Figure 12 shows the result of an exemplary point cloud segmentation for all evaluated networks.

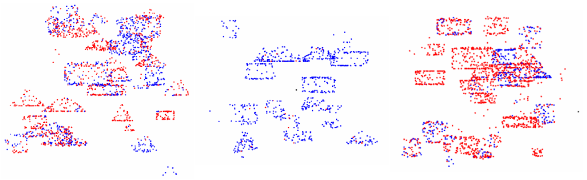


Figure 12: Segmentation results of example point clouds. (left: PointNet, middle: PointNet++, right: PointCNN). Red points indicate classification errors, blue points denote correctly classified points.

Test set	Metric	PointNet	PointNet++	PointCNN
No rotation	Acc	42.91	99.46	36.32
	IoU	18.66	98.73	15.28
Rotation	Acc	39.15	93.41	34.41
	IoU	16.27	80.38	14.55
Red. points	Acc	42.83	99.45	34.25
	IoU	18.59	98.34	13.51
Noise low	Acc	42.88	99.47	35.23
	IoU	18.70	98.65	14.80
Noise high	Acc	42.72	99.30	35.23
	IoU	18.67	97.54	15.06
More prim.	Acc	43.21	99.60	42.16
	IoU	19.32	99.43	19.88

Table 2: Results of PointNet, PointNet++ and PointCNN on the different test sets (Accuracy/IoU).

A possible explanation for the very strong performance of PointNet++ might be its approach of separating the point cloud into local regions first. Using a density adaptive layer, hierarchical features are then extracted in a next step. This seems to be working especially well for the task of classifying single, isolated primitives which are located in regions of otherwise low point density. This is a task in which the other networks show particularly low performance. The approach PointNet++ takes also proves to be robust against noise and it shows good generalization performance when the test set differs from the training set.

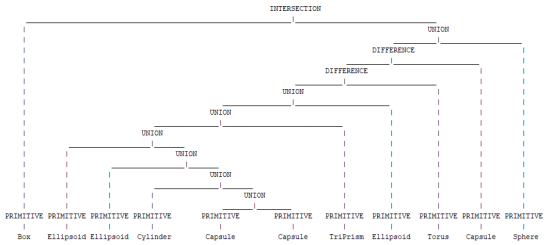


Figure 13: Extracted CSG tree for the giraffe model with 5 clusters (see first column in Figure 16).

7 CONCLUSION & FUTURE WORK

In this paper, we described a flexible and fast generator framework for creating primitive-based abstractions of triangle meshes. The proposed pipeline's output is a CSG tree which is extracted by a Genetic Algorithm. As an intended use case, we presented an evaluation of deep primitive segmentation networks which use training and test sets created by the generator.

Next steps include the integration of PointNet++ as a semantic clustering tool, replacing the currently used k-means approach. In addition, more sophisticated methods for primitive fitting like RANSAC [SWK07] could further improve resulting visual quality.

8 REFERENCES

[BM92] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[CFG+15] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

[FE17] Jonas Freiknecht and Wolfgang Effelsberg. A survey on the procedural generation of virtual worlds. *Multimodal Technologies and Interaction*, 1(4):27, 2017.

[FFPF18] Markus Friedrich, Sebastian Feld, Thomy Phan, and Pierre-Alain Fayolle. Accelerating evolutionary construction tree extraction via graph partitioning. In *Proceedings of WSCG International Conference on Computer Graphics, Visualization and Computer Vision*, 2018.

[HSS17] Karl Haubenwallner, Hans Peter Seidel, and Markus Steinberger. ShapeGenetics: Using Genetic Algorithms for Procedural Modeling. *Computer Graphics Forum*, 36(2):213–223, May 2017.

[KMJ+18] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A big CAD model dataset for geometric deep learning. *CoRR*, abs/1812.06216, 2018.

[LBSC18] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. PointCNN: Convolution on x-transformed points. *CoRR*, abs/1801.07791, 2018.

[Llo82] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[pol] Poly. https://poly.google.com/. Accessed: 2019-02-01.

[QSMG16] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[QYSG17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[Ric73] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973.

[RMGH15] Daniel Ritchie, Ben Mildenhall, Noah D Goodman, and Pat Hanrahan. Controlling procedural modeling programs with stochastically-ordered sequential monte carlo. *ACM Transactions on Graphics (TOG)*, 34(4):105, 2015.

[SGL+18] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. CSGNet: Neural shape parser for constructive solid geometry. 2018.

[SSG+18] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270, 2018.

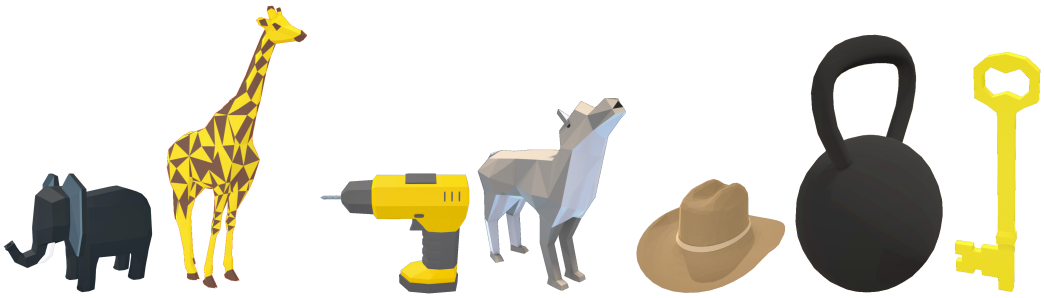
[SWK07] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for point-cloud shape detection. *Computer graphics forum*, 26(2):214–226, 2007.

[TKS+13] Olivier Teboul, Iasonas Kokkinos, Loic Simon, Panagiotis Koutsourakis, and Nikos Paragios. Parsing facades with shape grammars and reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1744–1756, 2013.

[TLL+11] Jerry O Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. Metropolis procedural modeling. *ACM Transactions on Graphics (TOG)*, 30(2):11, 2011.

[WSK+14] Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets for 2.5d object recognition and next-best-view prediction. *CoRR*, abs/1406.5670, 2014.

[ZYY+] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks.



(a) Elephant. (b) Giraffe. (c) Drill. (d) Wolf. (e) Cowboy. (f) Kettlebell. (g) Key.

Figure 14: All used models (©Poly by Google, CC-BY-License).

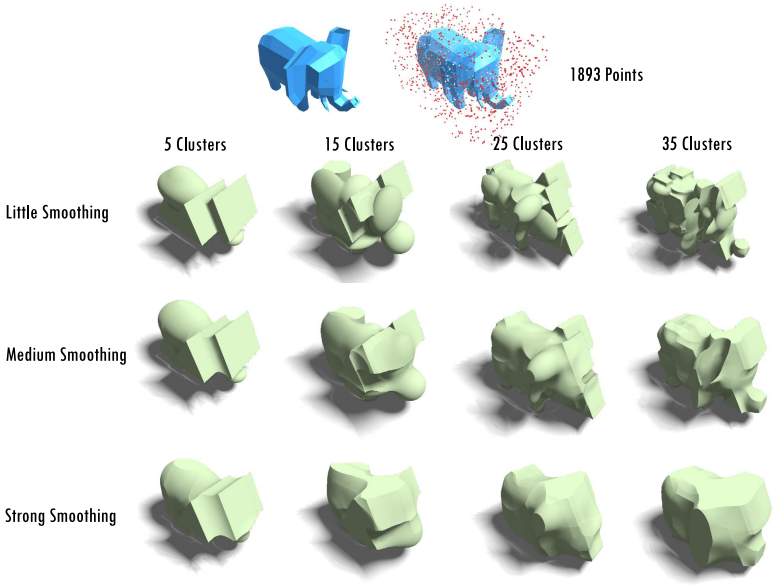


Figure 15: Extracted CSG tree models of the elephant model with different cluster sizes and blending strengths.

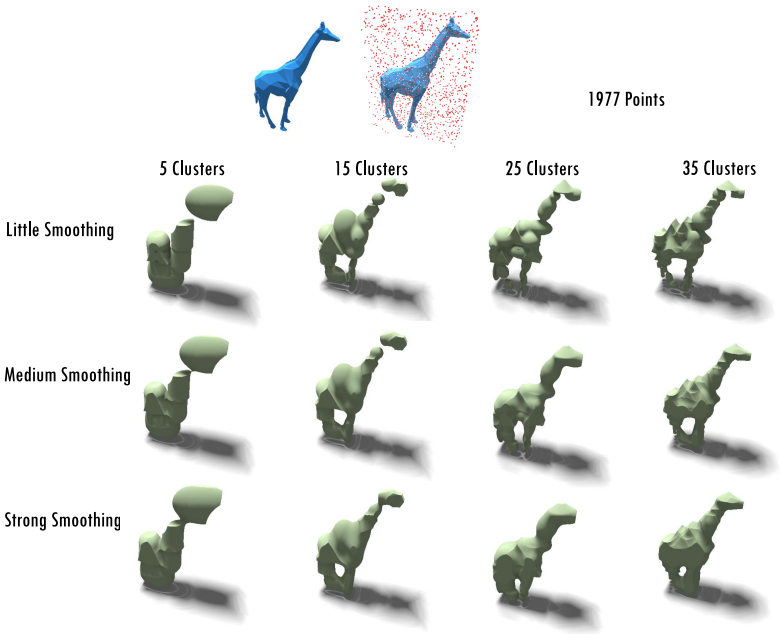


Figure 16: Extracted CSG tree models of the giraffe model with different cluster sizes and blending strengths.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.27, No.1, 2019

Automatic video fire detection approach based on PJF color modeling and spatio-temporal analysis

Zeineb Daoud
REsearch Groups in Intelligent Machines
University of Sfax
National Engineering School of Sfax
Sfax, 3038, Tunisia
zeineb.daoud.tn@ieee.org

Amal Ben Hamida
REsearch Groups in Intelligent Machines
University of Sfax
National Engineering School of Sfax
Sfax, 3038, Tunisia
benhamida.amal@gmail.com

Chokri Ben Amar
REsearch Groups in Intelligent Machines
University of Sfax
National Engineering School of Sfax
Sfax, 3038, Tunisia
chokri.benamar@ieee.org

ABSTRACT

Recently, due to the huge damage caused by fires in many countries in the world, fire detection is getting more and more interest as an increasing important issue. Nowadays, the early fire detection in video surveillance scenes is emerging as an alternative solution to overcome the shortcomings of the current inefficient sensors. In this paper, we propose a new video based-fire detection method exploiting color and motion information of fire. Our approach consists in detecting all moving regions in the scene to select then areas likely to be fire. Further, motion analysis is required to identify the accurate fire regions. The proposed method is evaluated on different video datasets containing diverse fire and non-fire videos. Experimental results demonstrate the effectiveness of our proposed method by achieving high fire detection and low false alarms rates. Moreover, it greatly outperforms the related works with 98.81 % accuracy and only 2 % of false positive rate.

Keywords

Video fire detection, background subtraction, color modeling, PJF color space, spatio-temporal analysis.

1 INTRODUCTION

Natural and human-instigated disasters threaten on the one hand the environment, people's lives and livelihoods on the other. Fire has been one of the most destructive disasters leading to enormous suffering and hazardous effects. The 2018 summer saw an unusually high number of fires in many countries of the world such United States, Sweden, Portugal, etc [Min18]. For instance, successively in 2017 and 2018, California was hit with the most destructive wildfires in its recorded history. According to the report presented in September 2018 by the California Department of Insurance, Carr Fire and Mendocino Complex are the largest ones [Eva18]. Carr Fire has destroyed 1 077 homes, 22 commercial structures, and 500 outbuildings have been devastated with at least hundreds of people were killed. As reported in [Min18], the Directorate General of Civil Security and Crisis Management (DGSCGC) of the French Ministry of the Interior announces that forest fire is covering large areas in Europe. For example, more than 350 000 ha in Portugal in 2017 compared to 24 500 ha in France. Therefore, there are significant social, economic and environmental impacts that the world is facing after fires, in spite the use of currently available fire sensors. These conventional sensors have been widely used for personal security and commercial applications. However, they usually take a long time to respond because carbon particles, smoke or heat require more time to reach the detector. Moreover, these traditional detectors are generally limited to indoor and are not applicable to outdoor in open large spaces such as forests [Çet13]. Most of them cannot provide additional information about fire location, dimension, etc [Çet13]. Face to the weaknesses of these sensors-based fire detection, fast and accurate fire detection methods are needed to protect both people and environment. Video Fire Detection (VFD) can be considered as a suitable solution especially with the propagation of video surveillance systems [ABe14, Ben16]. It has shown better flexibility, effectiveness and reliability [Han17], thanks to the capacity of image processing [Mej08, Gue11, Bou12] and of video processing techniques [Ela06, Kou12] to detect and analyze uncontrolled fire behavior in video surveillance scenes [ABe13, Ben13]. In fact, fire detection errors are reduced, and the response time can be immediate and faster than traditional sensors, as cameras do not need to wait for the smoke or heat diffusion. In addition, VFD systems are able to detect fire in large open scale

areas providing crucial information to understand clearly the development of fire, such as size, growth and propagation [Çet13]. Consequently, VFD methods have received great attention by researchers in these recent years. They focus on developing VFD algorithms, based on exploiting fire characteristics to detect and analyze flames in consecutive video frames. In this context, our work is introduced. It aims to detect fire regions accurately by using color information and motion analysis. The main idea is a combination of background subtraction, to keep at the first stage only moving objects and a color model to identify then the possible fire regions based on PJF color space. Add to that, a motion descriptor, based on spatial and temporal analysis, is applied to filter out fire-colored objects. This paper is organized as follows: Section 2, presents the related works of fire detection. A description of the new fire detection approach is presented in section 3. Section 4 provides experimental results and their discussions for system evaluation. Finally, conclusions are drawn in the last section.

2 RELATED WORKS

There are several methods in the literature developed for fire detection from video sequences. The current video-based fire detection systems use different fire features principally color, shape and motion. Based on these characteristics, the existing VFD systems can be classified into three categories: color-based, shape-based and motion-based.

Generally, almost all current methods employ color and motion information to detect the flame. The first works use purely color-based models. For instance, in [Che04], the RGB channels and the saturation values of the HSV color space are used. A set of rules is developed to determine fire pixels. These rules are based on the assumption that fire pixels have an intensity of red higher than green, and green higher than blue. However, this method does not well perform at distinguishing real fire regions from moving regions having similar fire color. To solve this problem, Çelik et al. in [Çel07] suggest to add the foreground object information to the statistical color model to detect fire pixels in RGB color space. This proposed fire detection method results in very high false alarms rates due to the brightness changes and its sensitivity to the tuning parameters employed in background subtraction. That is why, Çelik et al. propose in [Çel09] a chrominance model based on the YC_bC_r color space, since it can distinguish bright regions more than other color spaces. The authors use the mean intensity of the pixels and the difference between the $C_b$and C_r channels to compose detection rules. Hence, their generic color model improves its robustness in segmenting fire regions. Motivated by the idea that YC_bC_r is better in discriminating the luminance from chrominance, the fire recognition system suggested in [bin15], is based on a set of rules developed to identify the fire pixel values of R, G, B, Y, C_b and C_r components in an image. Results show that combination between color spaces can better detect fire and confirm that YC_bC_r color space is the best as to compare to RGB since it can separate luminance from chrominance more effectively than RGB. However, by applying this method, moving fire-colored regions, in complicated environments such light reflections and brighter or darker environments are also detected. To deal with this issue, a different approach in [Zho15] consists in excluding color and investigating the shape features of flames and interference image. Eight candidate features are selected to establish weak classifier for each shape attributes. Results of this method show the inefficiency of applying just shape to detect fire regions. From these works, the majority of the one-feature based methods achieves seldom satisfactory detection results. It may be concluded that obviously using one feature by itself cannot be reliable to detect fire. In fact, many natural objects sharing the same characteristics with flame are wrongly detected. Besides, changes in the flame appearance and environmental conditions (weather conditions, visibility and time of day) in the scene complicate fire detection. Thereby, a combination of characteristics is needed to detect fire accurately with lower errors rate. Some works such [Yan12] combines color and shape information, based on the assumption that fire randomly changes its shape during its propagation, to be able to distinguish fire from non-fire objects. In the same way, Xi Zhang et al. merge in [XiZ12] color and shape features. Fourier descriptors and edge corrosion model are used to recognize fire regions after its extraction by a mixture of Gaussian distributions in HSV color space. Another approach presented in [Cho10] adds fire's texture to color and shape features. Assuming that fire regions have a significant amount of texture characteristics because of its random nature, the proposed method is useful to differentiate between fire and non-fire regions such as autumn color leaves. Although adding shape or texture information to color improves fire detection, many fire colored objects still detected. It can be observed that using color only or combined with shape is not enough owing to the presence of moving objects having the same color or shape features of fire such as red cars and persons with red clothes. To face this issue, the distinctive disordered movement of fire regions must be taken into account. It is a pertinent way of differentiating moving rigid objects from red colored regions to enhance the detection rate. As example in [Han17], a new method for detecting fire based on motion feature and color information is proposed. Motion detection using Gaussian Mixture Model (GMM) for background subtraction is applied to extract moving objects from the scene. Then, multi-color-based

detection combining the RGB, HIS and YUV color spaces is employed to filter out non-fire regions. Once fire regions are extracted, foreground objects with similar fire colors or caused by color shifting are also selected. To remove these spurious fire-like regions, Kim Heegwang et al. propose in [Kim16] a different approach using color and motion estimation. It consists in extracting candidate regions in the HSI color space to estimate them using optical flow. Then, the motion vectors are quantized into eight directions and fire regions are detected when the amount of the quantized motion vectors exceeds a fixed threshold. The proposed system outperforms the color-based and the shape-based methods in terms of detection performance. As flame detection remains a challenging issue due to the fact that many natural objects have similar characteristics with fire, a new algorithm for video-based flame detection, which employs various spatio-temporal features such as color probability, contour irregularity, spatial energy, flickering and spatio-temporal energy is presented in [Dim12] by Dimitropoulos et al. They suggest adding in [Bar13] a new spatio-temporal consistency to combine it with the features extracted in [Dim12]. Texture feature in [Dim15] is also added to their fire detection system to increase its robustness. Thus, video-based fire detection systems using multi-features of fire are more efficient than systems based on only one feature. Moreover, in [Fog15a], a multi-expert system is developed by combining three descriptors based on color, shape and motion. A set of rules are implemented in the YUV color space to distinguish fire colored objects. Moving regions are detected as blobs via the scale-invariant feature transform (SIFT) descriptor. And, shape variation of the minimum bounding boxes which enclose the detected blobs is also used. Similarly, Li et al. design in [LiS17] a multi-attribute-based fire detection system which combines the color, geometric, and motion attributes. Two novel descriptors to characterize the geometric and motion features of fire are developed, the first one represents the outline of fire using contour moment and line detection and the other captures the instantaneous motion of fire with dense optical flow to well represent the inside and the boundary motions of fire. Despite of the capacity of these methods to detect fire regions, they still have limited application and lack enough robustness. They cannot effectively detect fire because of i) many natural objects sharing the same characteristics with the flame, ii) large variations of flame appearance in videos and iii) environmental changes that complicate fire detection, for example, shadows, clouds, sun shining, car lighting, light reflections or other kinds of lights can behave or flicker like fire [Bar13]. In order to overcome these limitations, a VFD method, which employs color information and motion analysis, is proposed in this paper.

3 THE NEW FIRE DETECTION APPROACH

By reviewing the above works, we aim to develop a video-based fire detection approach to reach high detection rate and low false alarms rate. The VFD method suggested in this work is basically composed of three phases as illustrated in figure 1.

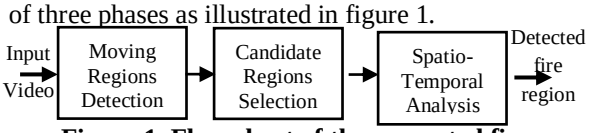


Figure 1. Flow chart of the presented fire detection method.

Moving regions are firstly detected by using a background subtraction algorithm. Then, a novel model in PJF color space is developed to select fire candidate regions. Spatio-temporal analysis is finally applied in order to distinguish fire regions from other fire-colored objects.

3.1 Moving regions detection

Detection of moving regions is a necessary step in VFD. Since fire can be naturally interpreted as a moving object in the video, several methods such as the Gaussian Mixture Model (GMM) in [LiS17] [Han17] and frame differences in [Pel18] [See14] are used to detect moving regions. Nevertheless, the most popular VFD systems are based on background subtraction methods to identify moving objects by separating foreground from background in video sequences. Thanks to their simplicity, real time performing and low computational cost, the Gaussian Mixture Models are widely used to create background models and estimate moving objects. We adopt in our approach the **G**lobal **M**inimum with a **G**uarantee (GMG) algorithm from the GMM family [God12]. It employs probabilistic foreground segmentation algorithm that identifies possible foreground objects to track them later on. In fact, GMG combines statistical background image estimation, per-pixel Bayesian segmentation, morphological filtering operations like closing and opening to remove inherent noise, and an approximate solution to the multi-target tracking problem using a bank of Kalman filters [Kal60] and Gale-Shapley matching [Gal62]. A heuristic confidence model enables selective filtering of tracks based on dynamic data. Figure 2, presented below, illustrates its system block diagram.

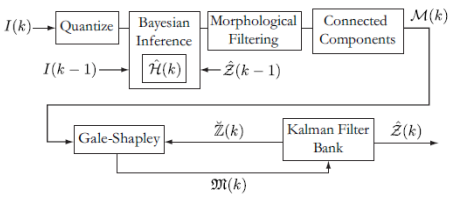


Figure 2. GMG algorithm block diagram [God12].

The GMG algorithm proves its robustness during variable lighting conditions and sudden changes in the

appearance of the background. From the detected moving regions after the GMG application, we aim to select only entities having the same fire's color characteristics. That's why we suggest using the color features of the moving areas to discriminate between fire and non-fire regions.

3.2 Candidate regions selection

Selection of candidate regions is the second stage in our approach for fire detection with the aim of distinguishing the fire regions from the other moving objects. It is based on the most distinctive characteristic of fire, which is the color. Therefore, we propose a new color-based model in the PJF color space [Jac12].

3.2.1 PJF color space

The idea behind the PJF color space was transforming the RGB values into a new color space that follows more closely the concept of L*a*b* by expressing brightness as a single variable. This color space expresses color in firstly a variable that runs from green to red and secondly with a variable from blue to yellow, while remaining vectors are inside the RGB color cube as depicted in figure 4 [Jac12]. That's why PJF reflects better the structure of L*a*b* color space by achieving a low color calibration error.

Given RGB data, the conversion to PJF color space is formulated as follows [Jac12]:

$$P = \sqrt{R^2 + G^2 + B^2} \quad (1)$$

$$J = R - G \quad (2)$$

$$F = R + G - B \quad (3)$$

Where the component P measures the magnitude of brightness, the second component J indicates the relative amounts of red and green and the third one F measures simply the relative amounts of yellow and blue. The new color channels produced by equations (1), (2) and (3) are shown in figure 3 and the PJF color space model is presented in figure 4.

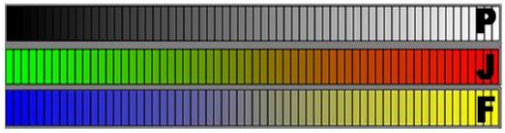


Figure 3. The three-color channels created by the color space transform [Jac12].

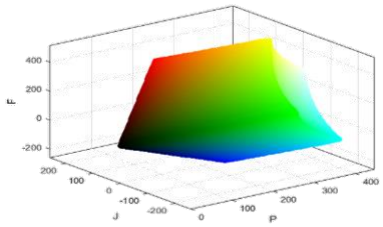


Figure 4. 3D distribution of the PJF color space.

It is assumed that fire's color is generally close to the red color with higher illumination values. Through PJF color space, the representation of fire color information is better than other color spaces since margin's variation of the fire color is presented by its two components J and F [Dus15]. Based on these cited statements, we propose to define rules in PJF color space to select fire regions.

3.2.2 Fire color-based model

A specific color spectrum characterizes fire where its natural colors are often in the red-yellow range. As mentioned above, the J channel in the PJF color space is the difference between red and green, and indicates the relative amounts of these two colors. Likewise, the F channel measures the relative amounts of yellow and blue. For these reasons fire pixels belong clearly in the J-F plane as confirmed in Fig. 4**.** Hence, these two PJF color space components can be used to estimate the range of flame pixels. In RGB image, fire is a red colored light source. Therefore, the red value R for each pixel in fire region must be over a threshold as presented in equation (4) [Che04].

$$R > R_T \quad (4)$$

Where R_T denotes the threshold value for the red channel.

Motivated by these facts and based on the two transformation equations (2) and (3), the proposed rule to select the candidate regions of fire from other moving objects is given below:

$$F \geq 2R_T - J \quad (5)$$

Where R_T is experimentally fixed to 120 [Che04]. At this stage, fire detection leads to high false alarms. The selected candidate regions of fire may still be non-fire because of several moving objects having the same colors as fire such persons wearing red clothes or carrying red artifacts, red vehicles, etc. Consequently, analysis of fire motion are needed to filter the obtained regions.

3.3 Spatio-temporal analysis

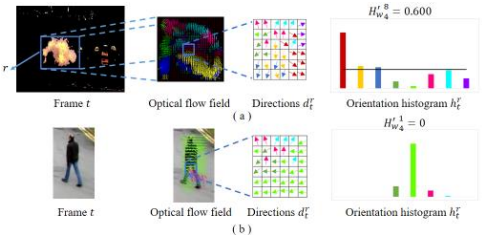


Figure 5. Optical flow fields and orientation histograms for (a) fire region and for (b) pedestrian object.

Motion is an essential characteristic that should be merged with color to detect accurately fire regions. It is clearly that the fire's motion generally differs from the motion of the irrelevant entities since it is characterized by its disordered movement. Thus, motion analysis is needed to effectively discriminate

these areas. It consists mainly of three steps described as follows: (1) computation of motion vectors, (2) spatial analysis and (3) temporal analysis to finally detect fire regions.

3.3.1 Computation of motion vectors

To analyze the motion of the selected regions, we firstly compute the optical flow field. Motion information is obtained by calculating the velocity vectors for each pixel (x, y) in region r at frame t. These motion vectors are got by using Farneback method [Far03]. A two-frame motion estimation algorithm uses firstly polynomial expansion, where a neighborhood of each image pixel is approximated by quadratic polynomials. Afterwards, considering these quadratic polynomials, a new signal is constructed by a global displacement that is calculated by equating the coefficients in the quadratic polynomials' yields. The optical flow field is described by $(Vx_t^r(x,y), Vy_t^r(x,y))$ where $Vx_t^r(x,y)$ and $Vy_t^r(x,y)$ denote the horizontal and vertical velocities respectively of the region r at frame t.

3.3.2 Spatial analysis

By adopting the fact that fire movements are multi-directional, we aim to find fire areas. The motion directions of each pixel of the selected candidate regions are firstly extracted from every frame, by using optical flow field obtained at the previous step. The direction d_t^r of each pixel (x, y) is then computed via the following equation:

$$d_t^r(x,\ y) = \tan^{-1}\left(\frac{Vy_t^r(x,\ y)}{Vx_t^r(x,\ y)}\right) \quad (6)$$

Where r is the selected candidate region in frame t. For a better characterization of the acquired directions, they are quantized to N_d levels as shown in figure 6 where N_d is set to 8.

Figure 6. Quantization of moving directions.

In order to discriminate fire and non-fire regions, the different directions are interpreted by computing the orientation histogram h_t^r for each candidate region r in frame t. The appearance of histograms as represented in the example of Fig. 5(a) confirms that the movement of the fire is disordered and spreads around the fire region, hence the dispersion of its orientation histogram. In contrast, the motion of non-fire objects, like the pedestrian in Fig. 5(b), is distributed in uniform directions, which explains that its orientation histogram is concentrated in few bins. Through the orientation histograms interpretation, we can conclude that a dispersed histogram generated with more bins corresponds to fire region. However, using orientation histograms may be insufficient to detect fire regions because of the similar histogram's appearance of some non-fire objects like autumn color leaves, which have disordered movement. That's why we propose to compute the normalized weighted entropy $H'^{r}_{w_t}$ which is obtained by calculating first of all the weighted entropy $H^{r}_{w_t}$ defined as follows:

$$H^{r}_{w_t} = -W_t^r * \sum_{\theta_j=1}^{N_d} p_t^r(\theta_j) \cdot log_2(p_t^r(\theta_j)) \quad (7)$$

Where r is the region of interest at frame t, p_t^r represents the orientation probability of each quantified direction θ_j, $j = 1,..,N_d$, of the motion vector. W_t^r denotes the weight which aims to maintain a deviation between the entropy of fire regions and that of non-fire in the interest of distinguishing easily between areas. That means, W_t^r enhances the entropy of fire region and reduces the entropy of non-fire region. It is calculated using the orientation histogram by counting the number of the most frequent bins with a greater amount of information as shown in the following equation:

$$W_t^r = Card\ \{h_t^r(\theta_j),\ j=1...N_d \text{ with } h_t^r(\theta_j) > \frac{\sum_{\theta_j=1}^{N_d} h_t^r(\theta_j)}{N_d}\} \quad (8)$$

We normalize then the range of values of $H^{r}_{w_t}$ by using min-max normalization to scale them in [0, 1] to obtain $H'^{r}_{w_t}$.

3.3.3 Temporal analysis

Fire region in certain single frame may not be detected because of the flickering effects and instability of the motion. Flickering is one of the main features of flame and it is very significant to discriminate between fire regions and fire-colored objects. Besides, the motion of non-fire objects, like human and vehicles, is regular, hence it is temporally continuous and uniform. Differently to other entities, fire is always in disordered and discontinuous motion due to the flickering effects. For these reasons, temporal analysis is applied to each candidate region. After calculating normalized weighted entropy $H'^{r}_{w_t}$ in the previous step, the candidate areas must be further verified using movement's temporal variation since motion analysis in a single frame are not enough to identify fire. For each candidate region r, we suggest to calculate, in equation (9), the temporal entropy $H_T(r)$ over the consecutive N frames as follows:

$$H_T(\mathrm{r}) = \frac{\sum_{t=1}^{N} H'^{r}_{w_t}(r)}{N} \quad (9)$$

Where $H'^{r}_{w_t}$ is the normalized weighted entropy of region r at frame t and N represents the number of frames. At this point, a threshold τ will be determined to distinguish fire regions. As shown in equation (10),

if H_T(r) is greater than a given τ, the region r is classified as fire.

$$\text{r} = \begin{cases} fire\ if H_T(r) \geq \tau \\ non\ fire\ if H_T(r) < \tau \end{cases} \quad (10)$$

4 EXPERIMENTAL RESULTS

In this section, we evaluate our approach on a wide dataset. Moreover, we carry out comparison with two recent fire detection methods [LiS17] [Pel18].

4.1 Dataset

In spite of the importance of fire detection researches nowadays, there are no standard benchmark datasets for fire detection up to now. Therefore, to test the presented approach we construct our database from four public datasets available in [Fog15b] [Çet07] [Caz17] [Kmu18]**.** It contains 157 videos varied between 10 and 30fps, in a diverse range of environmental conditions and illumination**:** 137 positive videos characterized by the presence of fire and 20 negative videos, which do not contain fires. Hence, 246 581 frames compose this collection.

4.2 Evaluation metrics

In order to measure the robustness of our new method, it is necessary to compute validation metrics. True positive (TP), false positive (FP), true negative (TN) and false negative (FN) are calculated. TP is the number of frames where fire regions are correctly detected. FP is the number of frames in which non-fire regions are incorrectly detected as fire. TN is the number of frames with non-fire regions that are correctly rejected and FN is the number of frames in which fire regions are wrongly rejected. For negative fire videos, the evaluation is done by using the detection error rate ER calculated in the equation (11) presented below. It represents the rate of recognizing fire in a frame where there is no fire. TNR and FPR metrics are also used to measure respectively the ability of the algorithm to well eliminate non-fire regions and the false alarms rate.

$$ER = \frac{FP}{Total\ Number\ of Frames} \quad (11)$$

Similarly, positive fire videos are assessed with different metrics to measure the effectiveness of detecting fire regions by the proposed method. As a first metric, we use the detection rate DR; it is defined as the rate of detecting fire successfully by the following equation:

$$DR = \frac{TP}{Total\ Number\ of\ Frames} \quad (12)$$

Accuracy is also employed to measure how close our algorithm can detect the correct fire areas. It is calculated via the equation (13) below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (13)$$

Recall and precision are calculated too, as stated in equations (14) and (15). Recall metric, named also sensitivity, expresses the ability of an algorithm to find all the pertinent cases. While, precision measures the proportion of detections that a method indicates as relevant when they are really relevant.

$$Recall = \frac{TP}{TP + FN} \quad (14)$$

$$Precision = \frac{TP}{TP + FP} \quad (15)$$

In order to measure the trade-off between precision and recall, we combine the two metrics using the $F_1\ score$ in the following equation. It is a weighted harmonic mean showing how many frames with fire regions are correctly detected.

$$F_1 score = 2 * \frac{Recall * Precision}{Recall + Precision} \quad (16)$$

4.3 Evaluation of the new fire detection approach

It should be noted that the number of frames N used to calculate the temporal entropy for each candidate area (cited in the section 3.3.3), is experimentally set to 20 according to results presented in Fig. 7. This figure shows the variation of the fire detection accuracy of different fire videos samples to validate the choice of N frames. For example, the first scene (Scene_1) achieves in terms of accuracy 0.84, 0.88, 0.83, and 0.76 with 10, 20, 30, 40 and 50 frames respectively. It is discernable from Fig. 7 that increasing N decreases the accuracy values. The best rates are obtained when N is equal to 20. As a consequence, we adopt $N = 20$ to evaluate our method.

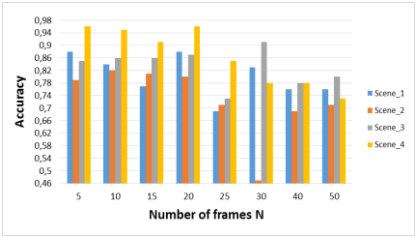


Figure 7. Comparison of fire detection accuracy in video sequences by varying the number of frames *N*.

In this section, we aim at successfully detecting fire regions in several scenes to measure the performance of our video-based fire detection approach. Figure 9 illustrates the output frames of each phase of our approach. On the one hand, Fig. 9(A) shows the results on three different positive fire movies. It can be remarkable that our presented method is able to eliminate non-fire moving objects, such as in video A2. These include fire-like colored objects, such the firefighters wearing jackets and caps with similar color of fire. In the same way, fire-color background areas, which may cause false detection like scene A1

in Fig. 9(A), are rejected too. Moreover, fires in videos, which are filmed in different conditions of environment, weather and illumination (in forest, indoor, outdoor, day and night scenes), are successfully detected. On the other hand, results on negative videos where there is no fire are presented in Fig. 9(B). Spurious fire-like objects such as sun reflections, human with red clothes and bright sparkling lights are correctly rejected. Scene B3 of Fig. 9(B) proves that spatial analysis at Fig. 9(B3.c) (as detailed in the section 3.3.2) are not sufficient to distinguish fire regions from areas sharing the same fire's characteristics. As it is shown in Fig. 9(B3.b) and Fig. 9(B3.c) bright sparkling lights are still detected after applying our color model. But, adding temporal analysis improves differentiation between regions. As depicted in Fig. 9(B3.d), all fire colored areas are rejected. This further confirms the robustness of our approach in recognizing fire regions.

From these results, we can conclude that our proposed method succeeds in distinguishing fire from non-fire regions. This is thanks to the fact that the deviation between the temporal entropy H_T values of fire and that of non-fire regions has become more larger. In fact, they can be easily differentiated. These results are clearly observed in figure 8, which shows the variation of temporal entropy for fire and non-fire areas.

As it is illustrated, values of temporal entropy for fire region (the red curve) is greater than those of non-fire. Through the analysis of the variation of temporal entropy in figure 8, discriminating between moving regions is becoming easier. It is sufficient to determine the threshold τ (cited in the section 3.3.3). Otherwise, H_T of fire area is over than τ. This evidently confirms that fire is always in active motion and its temporal entropy is near to 1. As it is shown in Fig. 5(a), a fire region has a temporal entropy value equal to 0.832.

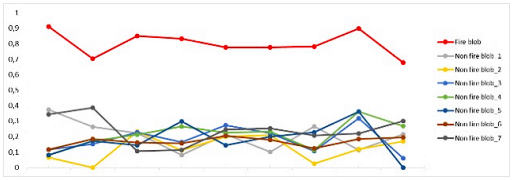


Figure 8. Temporal variation of entropy for fire region and non-fire areas.

However, for areas which may be a moving objects having the same colors as fire, their temporal entropy is close to 0. For example, $H_T$value for a non-fire area is equal to 0.165. As earlier detailed, these values demonstrate effectively that the movement of fire-colored regions is uniform and significantly different from the motion of fire regions. That's why fire areas have a higher value of temporal entropy H_T. In this study, our experiments show that the best discrimination is achieved when the value of the used threshold τ is fixed to 0.5.

The assessment of this approach is given below, and results are shown in Table 1 and Table 2. For almost positive fire movies, our novel method has yielded a true positive (TP) average of 99% for detecting the fire regions correctly, hence the detection rate DR is high. Values of false positive (FP) are null which leads to a null detection error rate ER, this proves how effectively our algorithm can distinguish fire from non-fire regions. Precision and recall values in Table 1 reach over 0.9. Indeed, the obtained values of recall are respectable, they are around of 0.99. High precision relates to the low false positive rate. This proves also that almost all fire regions are detected correctly. Most significantly, the overall performance of the presented approach is high in terms of the weighted harmonic mean $F_1 score$ with 0.99. Add to that, Table 2 shows the achieved results on the accuracy of fire detection in three negative fire scenes.

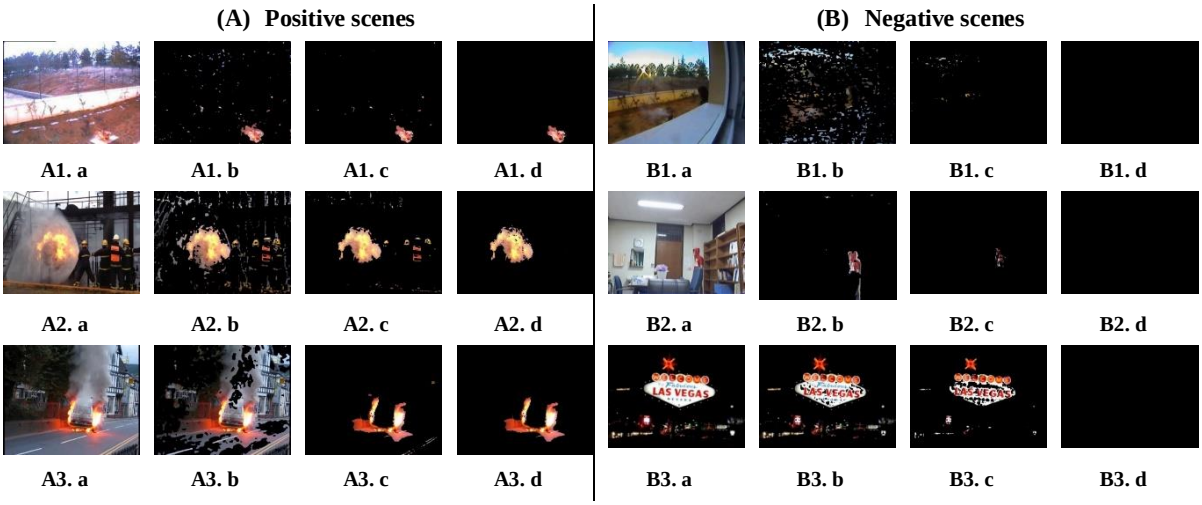


Figure 9. Results of the presented fire detection method on (A) positive fire videos and on (B) negative movies. (a) Original frames, (b) Moving regions detection using the GMG algorithm, (c) Candidate regions selection by applying the PJF color model, (d) Detected fire regions obtained after spatio-temporal analysis.

Videos	TP (%)	FP (%)	DR (%)	ER (%)	Recall	Accuracy	Precision	F_1 score	Sample picture
fireVid_001	98	0	98	0	0.98	0.98	1	0.98	
fire_8	99	0	99	0	0.99	0.99	1	0.99	
flame_1	100	0	100	0	1	1	1	1	
fire_11	99	0	99	0	0.99	0.99	1	0.99	
rescuer_060	96	0	96	0	0.96	0.96	1	0.98	
rescuer_021	100	0	100	0	1	1	1	1	
Average	98.66	0	98.66	0	0.99	0.99	1	0.99	

Table 1. Fire detection results of the presented method applied on positive fire movies

Videos	TN (%)	FN (%)	ER (%)	TNR	FPR	Accuracy	Sample picture
fire_15	100	0	0	1	0	1	
fire_16	100	0	0	1	0	1	
fireworks	94	0	6	1	0.06	0.94	
Average	98	0	2	1	0.02	0.98	

Table 2. False fire detection results of the presented method applied on negative fire movies

Since there is no fire, FN of the proposed algorithm is null. TN average of 98% is obtained resulting firstly in a greater TNR, secondly in a higher accuracy with 0.98 and lastly in a lower error rate with 2%. These values reflect the interesting performance of our contribution that is to say it can distinguish accurately fire regions from non-fire areas having the same colors as fire.

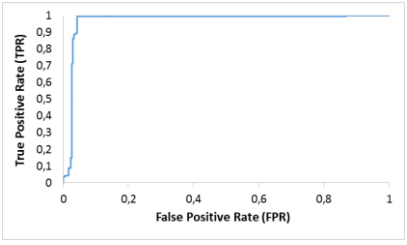


Figure 10. Receiver Operation Characteristics Curve.

As well, the performance is depicted in figure 10 where the Receiver Operating Characteristics (ROC) curve is presented. This curve shows how much the novel approach is able to discriminate between fire and non-fire regions. The area under the ROC curve is a pertinent indicator for performance measuring. The higher is the Area Under the Curve (AUC) the better is the algorithm at distinguishing between fire regions and fire-like colored objects [Faw06]. For this, the AUC computed value is 0.969. Therefore, the ROC curve proves too that our method has a low false alarms rate. From these results, we can conclude that the proposed approach accomplishes a rate of accuracy values over than 0.9 for both positive and negative scenes of the datasets, and only 2% for the false alarms rate.

Method	Accuracy	FPR	FNR
[Pel18]	64.30%	_	_
[LiS17]	92.30%	8.33%	9.09%
Presented	98.81%	2%	1%

Table 3. Comparison of the presented approach with two related works

For better assessment, we compare the presented approach with other fire detection methods based on motion and color features. In [LiS17], a multi-attribute-based fire detection system is developed which combines the color, geometric, and motion attributes. While, [Pel18] is based on color segmentation and moving objects detection by applying the frame differences. The experimental results are illustrated in Table 3.

It should be noted that accuracy, FPR and FNR values presented in Table 3 are the averages of all used datasets. FPR and FNR values obtained with our contribution are better than [LiS17] method with a decrease from 8.33% to 2% and from 9.09% to 1% respectively. These values are very low that is to say that our system can distinguish between fire regions and fire-like colored objects in the same frame, detect very well fire regions and eliminate undesirable areas. From Table 3, it can be found that our results significantly outperform Li's [LiS17] and Peleshko's

methods [Pel18] in accuracy rate with the highest value of 98.81%. By combining the PJF color model, the motion information and the spatio-temporal analysis of the fire's behavior, the obtained accuracy is nearly 6.5% and 34.5% greater than the related works [LiS17] and [Pel18] respectively. As shown in the first row of Table 3, accuracy in [Pel18] is 64.30%, because of using only color segmentation with frame differences. It cannot correctly discriminate fire from non-fire moving objects. Whereas, although Li's method in [LiS17] accomplishes 92.30% of accuracy, it may yield a few failure cases on some sequences (long distance between fire and the camera). In these cases, the geometric and motion attributes cannot work well because the fire areas are too small. For our method, it is able to detect fire regions in these special cases despite the far distance. Compared to the motion attribute in [LiS17], motion analysis in our work can obviously improve the detection accuracy by using spatial and temporal information. Generally, the proposed approach achieves a promising performance in detecting fire.

5 CONCLUSION

In this paper, we present an effective novel approach for fire detection in video sequences. It is based on color and motion features. It consists in detecting moving regions at the first stage using the GMG algorithm for background subtraction. After that, a novel fire color model developed in PJF color space selects candidate regions. Therefore, the detected fire regions are determined by a spatio-temporal analysis of the fire's motion. This method is tested over different datasets. It is shown experimentally that we succeed in detecting fire regions with accurate discrimination between fire and non-fire areas. Through experiments, we demonstrate that the achieved false alarms rate is only 2%. The false negative rate where fire regions are wrongly rejected is reduced too to reach 1%. Experimental results show also that our new method clearly outperforms the related works in terms of fire detection accuracy. In order to face the variable illumination problem, our approach can be further improved in the future. And the forthcoming scope includes the application of deep-learning methods.

6 ACKNOWLEDGMENTS

The research leading to these results has received funding from the Ministry of Higher Education and Scientific Research of Tunisia under the grant agreement number LR11ES48.

7 REFERENCES

[ABe13] A. Ben Hamida, M. Koubaa, H. Nicolas and C. Ben Amar, "Video pre-analyzing and coding in the context of video surveillance applications," *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, San Jose, CA, pp. 1-4, 2013.

[ABe14] A. Ben Hamida, M. Koubaa, C. Ben Amar and H. Nicolas, "Toward scalable application-oriented video surveillance systems," *Science and Information Conference*, London, pp. 384-388, 2014.

[Bar13] Barmpoutis, P., Dimitropoulos, K. & Grammalidis, N. Real time video fire detection using spatio-temporal consistency energy. in *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance* 365–370, IEEE, 2013.

[Ben13] Ben Hamida A, Koubaa M, Nicolas H, Ben Amar C. Spatio-temporal video filtering for video surveillance applications. In: *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*,pp 1–6, 2013.

[Ben16] Ben Hamida, A., Koubaa, M., Nicolas, H., Ben Amar, C. Video surveillance system based on a scalable application-oriented architecture. *Multimedia Tools and Applications*, 75 (24), pp. 17187-17213, 2016.

[bin15] binti Zaidi, N. I., binti Lokman, N. A. A., bin Daud, M. R., Achmad, H. & Chia, K. A. Fire recognition using RGB and YCbCr color space. *ARPN J. Eng. Appl. Sci.* 10, 9786–9790, 2015.

[Bou12] Boughrara, H., Chtourou, M., Amar, C.B. MLP neural network based face recognition system using constructive training algorithm. *Proceedings of 2012 International Conference on Multimedia Computing and Systems, ICMCS 2012*, pp. 233-238, 2012.

[Caz17] Cazzolato, M. T., Avalhais, L. P., Chino, D. Y., Ramos, J. S., de Souza, J. A., Rodrigues-Jr, J. F., & Traina, A. J. FiSmo: A Compilation of Datasets from Emergency Situations for Fire and Smoke Analysis. *Proc. Satell. events*, 2017. Computer vision based fire detection database. [Online]. Available: https://goo.gl/uW7LxW. Accessed August 2018.

[Çel07] Çelik, T., Demirel, H., Ozkaramanli, H. & Uyguroglu, M. Fire detection using statistical color model in video sequences. *J. Vis. Commun. Image Represent.* 18, 176–185, 2007.

[Çel09] Çelik, T. & Demirel, H. Fire detection in video sequences using a generic color model. *Fire Saf. J.* 44, 147–158, 2009.

[Çet07] Çetin, A. E. Computer vision based fire detection software, 2007. Database. [Online]. Available: http://signal.ee.bilkent.edu.tr/VisiFire/. Accessed January 2018.

[Çet13] Çetin, A. E., Dimitropoulos, K., Gouverneur, B., Grammalidis, N., Günay, O., Habiboğlu, Y. H., & Verstockt, S. Video fire detection–review. *Digit. Signal Process.* 23, 1827–1843, 2013.

[Che04] Chen, T.-H., Wu, P.-H. & Chiou, Y.-C. An early fire-detection method based on image processing. in *2004 International Conference on Image Processing, 2004. ICIP'04.* 3, 1707–1710, IEEE, 2004.

[Cho10] Choi, H.-W., Min, I.-K., Oh, E.-S. & Park, D.-H. A study on the algorithm for fire recognition for automatic forest fire detection: The international conference on control, automation and systems 2010 (iccas 2010). in *ICCAS 2010* 2086–2089, IEEE, 2010.

[Dim12] Dimitropoulos, K., Tsalakanidou, F. &

Grammalidis, N. Flame Detection for Video-based Early Fire Warning Systems and 3D Visualization of Fire Propagation. in *13th IASTED International Conference on Computer Graphics and Imaging (CGIM 2012), Crete, Greece*, 2012.

[Dim15] Dimitropoulos, K., Barmpoutis, P. & Grammalidis, N. Spatio-temporal flame modeling and dynamic texture analysis for automatic video-based fire detection. *IEEE Trans. Circuits Syst. Video Technol.* 25, 339–351, 2015.

[DuS15] Du S.-Y.a & c, L. Z.-G. . A comparative study of different color spaces in computer-vision-based flame detection. *Multimed. Tools Appl.* 75, 10291–10310, 2015.

[Ela06] El'Arbi, M., Ben Amar, C., Nicolas, H. Video watermarking based on neural networks. *2006 IEEE International Conference on Multimedia and Expo, ICME 2006* - Proceedings, pp. 1577-1580, 2006.

[Eva18] Evan Mills, Ted Lamm, Sadaf Sukhia, Ethan Elkind, and Aaron Ezroj. Trial by Fire: Managing Climate Risks Facing Insurers in the Golden State. Sacramento, CA: California Department of Insurance, 2018. Available at: https://www.law.berkeley.edu/wp-content/uploads/2018/09/Trial-by-Fire-September-2018.pdf.

[Far03] Farnebäck, G. Two-frame motion estimation based on polynomial expansion. in *Scandinavian conference on Image analysis* 363–370, Springer, 2003.

[Faw06] Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **27,** 861–874, 2006.

[Fog15a] Foggia, P., Saggese, A. & Vento, M. Real-Time Fire Detection for Video-Surveillance Applications Using a Combination of Experts Based on Color , Shape, and Motion. *IEEE Trans. Circuits Syst. Video Technol.* 25, 1545–1556, 2015.

[Fog15b] Foggia, P. Computer vision based fire detection dataset. [Online]. Available: http://mivia.unisa.it/datasets/video-analysis-datasets/fire-detection-dataset/. Accessed December 2017.

[Gal62] Gale, D and Shapley, L. D. College admissions and the stability of marriage. *Am. Math. Mon.* 69, 9–15, 1962.

[God12] Godbehere, A. B., Matsukawa, A. & Goldberg, K. Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. in *2012 American Control Conference (ACC)* 4305–4312, IEEE, 2012.

[Gue11] Guedri, B., Zaied, M., Ben Amar, C. Indexing and images retrieval by content. *Proceedings of the 2011 International Conference on High Performance Computing and Simulation, HPCS 2011*, pp. 369-375, 2011.

[Han17] Han, X.-F. *et al.* Video fire detection based on gaussian mixture model and multi-color features. *Signal, Image Video Process.* 11, 1419–1425, 2017.

[Jac12] Jackman, P., Sun, D.-W. & ElMasry, G. Robust colour calibration of an imaging system using a colour space transform and advanced regression modelling. *Meat Sci.* 91, 402–407, 2012.

[Kal60] Kalman, R. E. A new approach to linear filtering and prediction problems. *J. basic Eng.* 82, 35–45, 1960.

[Kim16] Kim, H., Park, J., Park, H. & Paik, J. Fire flame detection based on color model and motion estimation. in *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)* 1–2, IEEE, 2016.

[Kmu18] KMU Fire and smoke database. [Online]. Available: https://cvpr.kmu.ac.kr/dataset/dataset.htm. Accessed October 2018.

[Kou12] Koubaa, M., Elarbi, M., Ben Amar, C., Nicolas, H. Collusion, MPEG4 compression and frame dropping resistant video watermarking. *Multimedia Tools and Applications*, 56 (2), pp. 281-301, 2012.

[LiS17] Li, S., Liu, W., Ma, H. & Fu, H. Multi-attribute based fire detection in diverse surveillance videos. in *International Conference on Multimedia Modeling* 238–250, Springer, 2017.

[Mej08] Mejdoub, M., Fonteles, L., BenAmar, C., Antonini, M. Fast indexing method for image retrieval using tree-structured lattices. *2008 International Workshop on Content-Based Multimedia Indexing, CBMI, Conference Proceedings*, pp. 365-372, 2008.

[Mia13] Miao, L. & Wang, A. Video flame detection algorithm based on region growing. in *2013 6th International Congress on Image and Signal Processing (CISP)* 2, 1014–1018, IEEE, 2013.

[Min18] Ministry of the interior of France. Dossier de presse relatif à la campagne 2018 de lutte contre les feux de forêts. 2018. Available at: https://www.interieur.gouv.fr/Media/MI/Files/Actualites/dossier-de-presse-relatif-a-la-campagne-2018-de-lutte-contre-les-feux-de-forets. Accessed: November 2018.

[Pel18] Peleshko, D., Maksymiv, O., Rak, T., Voloshyn, O. & Morklyanyk, B. Core generator of hypotheses for real-time flame detecting. in *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* 455–458, IEEE, 2018.

[See14] Seebamrungsat, J., Praising, S. & Riyamongkol, P. Fire detection in the buildings using image processing. in *2014 Third ICT International Student Project Conference (ICT-ISPC)* 95–98, IEEE, 2014.

[Tru12] Truong, T. X. & Kim, J.-M. Fire flame detection in video sequences using multi-stage pattern recognition techniques. *Eng. Appl. Artif. Intell.* 25, 1365–1372, 2012.

[XiZ12] Xi, Z., Fang, X., Zhen, S. & Zhibin, M. Video flame detection algorithm based On multi-feature fusion technique. in *2012 24th Chinese Control and Decision Conference (CCDC)* 4291–4294, IEEE, 2012.

[Yan12] Yan, Y. Y., Gao, S. B., Wang, H. Y. & Guo, Z. B. Contour extraction of flame for fire detection. in *Advanced Materials Research* 383, 1106–1110, Trans Tech Publ, 2012.

[Zho15] Zhou, Q., Yang, X. & Bu, L. Analysis of shape features of flame and interference image in video fire detection. in *2015 Chinese Automation Congress (CAC)* 633–637, IEEE, 2015.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.27, No.1, 2019

An automatic stopping criterion for nonlinear anisotropic diffusion

César Bustacara-Medina
cbustaca@javeriana.edu.co
Pontificia Universidad Javeriana
Bogotá D.C., Colombia

Leonardo Flórez-Valencia
florez-l@javeriana.edu.co
Pontificia Universidad Javeriana
Bogotá D.C., Colombia

Abstract

Nonlinear anisotropic diffusion (NAD) filtering is a procedure based on nonlinear evolution PDEs which seeks to improve images qualitatively by removing noise while preserving details and even enhancing edges. However, well-known implementations are sensitive to parameters which are necessarily tuned to sharpen a narrow range of edge slopes; otherwise, edges are either blurred or staircased [Kee2002]. One important parameter is the iterations number, for that reason, in this paper a stopping criterion to halt the diffusion process is proposed. To meet this goal, two stopping criteria were compared. The first is the stopping criterion proposed by Joao et. al. [Joã2016], which is based on the Mean Squared Error (MSE). The second is our proposed method based on the CIRR contrast measure. To this end, a comparative analysis of five diffusion methods is performed. Four of them are nonlinear anisotropic diffusion methods and the fifth is the Perona-Malik method. According to the tests performed, the number of iterations required by the smoothing algorithms using the proposed stopping criterion is lower.

Keyword: Smoothing, Stopping criteria, Nonlinear anisotropic diffusion, Edge detection

1 INTRODUCTION

Medical images typically suffer from one or more of the following imperfections, low resolution (in the spatial and spectral domains), high level of noise, low contrast, geometric deformations and/or presence of imaging artifacts. These imperfections can be inherent to the imaging modality (e.g., X-rays offer low contrast for soft tissues, ultrasound produces very noisy images, and metallic implants will cause imaging artifacts in CT) or the result of a deliberate trade-off during acquisition. For example, finer spatial sampling may be obtained through a longer acquisition time. However that would also increase the probability of patient movement and thus blurring. To remove noise while preserving details and even enhancing edges techniques based on Partial Differential Equations (PDEs) have been used. The idea of using the PDE diffusion equations in image denoising and restoration arose from the use of the Gaussian filter in multiscale image analysis.

Convolving an image with a two or three dimensional Gaussian filter is equivalent to the solution of the diffusion equation in two or three dimensions [Bar2014]. Nowadays, PDEs have been successfully applied to many problems in image processing and computer vision [Ter1994, Cas1998, Sap2006, Aub2006, Cao2003], e.g., denoising [Per1990a], enhancement [Rud1989], inpainting [Ber2000], segmentation [Li2005], stereo and optical flow computation [Sap2006].

Nonlinear anisotropic diffusion is a variant of the heat equation, generalized in two regards: nonlinearity and anisotropy. Nonlinearity in diffusion means that diffusion tensors are automatically generated from the processed image. Anisotropy means that the smoothing induced by the PDE can be favored in some directions and prevented in others. This is specified by local eigenvectors and eigenvalues of the diffusion tensor field [Wei1996]. Diffusion coefficients are thus location and direction dependent, generalizing the approach of Perona and Malik [Per1990, Per1990a] which is only location dependent.

NAD is a powerful image processing technique, which allows to simultaneously remove the noise and enhance sharp features in two or three dimensional images. Anisotropic diffusion is understood here in the sense of Weickert [Wei1998], meaning that diffusion tensors are anisotropic and reflect the local orientation of image features. Weickert [Wei1999] proposed two nonlinear anisotropic diffusion algorithms. The first one is called Edge Enhancing Diffusion (EED), which al-

lows smoothing while preserving the edges. The second one is called Coherence Enhancing Diffusion (CED), which allows smoothing based on the structures (flow-like) present in the images. Based on the filters defined by Weicket, several methods have been proposed. For example, Bazan et. al. [Baz2007, Baz2009] proposed a new approach based on nonlinear anisotropic diffusion and bilateral filtering for electron tomography of mitochondria. Dong et. al. [Don2015] introduced a source term in the CED filter to restore the initial image and contrast lost by pure diffusion filters. Prasath [Pra2016] proposed an adaptive coherence enhancement diffusion filter (CED) combining anisotropic diffusion and diffusion functions derived from the structural tensor. Mirebeau et. al. [Mir2014] proposed two variants to the Weickert's algorithms. The first is associated with the EED algorithm, which is called Conservative variant of EED (cEED). The second is called Conservative variant of CED (cCED). The main distinction lies in the definition of the diffusion parameters of the diffusion tensor.

This paper is organized as follows: In section 2, nonlinear anisotropic diffusion filters are explained. In section 3, the most relevant methods to stop the diffusion propagation are described. In section 4, the proposed method for stopping the propagation is briefly explained. In section 5, tests performed in 2D and 3D images are presented, and the paper finishes with some conclusions.

2 NONLINEAR ANISOTROPIC DIFFUSION FILTERS

The idea of nonlinear anisotropic diffusion was pioneered by Nitzbeg et. al. [Nit1992] and Cottet et al. [Cot1993]. Later on, Weickert [Wei1999] put forward a formal method for enhancing the elongated structure, referred to as coherence-enhanced diffusion (CED).

NAD filtering is a procedure based on nonlinear evolution PDEs which seeks to improve images qualitatively by removing noise while preserving details and even enhancing edges. In the anisotropic case not only the amount of diffusion is adapted locally to the data but also the direction of smoothing. It allows for example to smooth along image edges while inhibiting smoothing across edges. This can be achieved by replacing the scalar-valued diffusivity function by a matrix-valued diffusion tensor [Bro2006].

The eigenvectors of the diffusion tensor define the principal directions of smoothing and the corresponding eigenvalues define the amount of smoothing. Weickert based the diffusion tensor on the structure tensor [Wei1996, Wei1997, Wei1998], which describes structures in the image using first order derivative information [Bus2016].

In general, any nonlinear anisotropic diffusion can be described by the equation

$$\frac{\partial u}{\partial t} = \operatorname{div}\left(D\left(\nabla u\right)\nabla u\right) \tag{1}$$

where u is the initial smoothed image that is initialized with the input image f (that is $u\left(\mathbf{x},0\right)=f\left(\mathbf{x}\right)$), and D represents a matrix-valued diffusion tensor that describes the smoothing directions and the corresponding diffusivities [Erd2012]. In this case, the diffusion tensor D is a function of $\mathbf{x}$, i.e., depends on the space. Additionally, D is a positive definite symmetric matrix [Wei1998, Wei2002]. The idea is to adaptively choose the diffusion coefficient D such that intra-regions become smooth while edges of inter-regions are preserved [Cha2010]. As D must be a nonnegative function of gradient magnitude so that small variations in intensity such as noise or shading can be well smoothed, and edges with large intensity transition are retained. It is generally given by an exponential function or an inverse quadratic function, and determined by the gradient magnitude with respect to a predetermined edge strength threshold [Cha2010].

Thus the given image u is usually convolved with a Gaussian kernel G_σ with a relatively small standard deviation σ as a presmoothing step. Cottet and Germain [Cot1993] and Weickert [Wei1996] devise a diffusivity matrix of the form:

$$D_\sigma = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ v_3^T \end{bmatrix} \tag{2}$$

where the vectors v_i are the eigenvectors of the structure tensor and the parameters λ_i are functions of the eigenvalues of the structure tensor. The images's structure tensor is defined as [Wei1997]:

$$J_\rho\left(\nabla u_\sigma\right) = G_\rho * \left(\nabla u_\sigma \cdot \nabla u_\sigma^T\right) \tag{3}$$

where G_ρ is the Gaussian kernel with standard deviation ρ (integration scale), over which the orientation information is averaged, and ∇u_σ is the gradient of the image u at scale σ. Principle axis transformation gives the eigenvectors and eigenvalues of $J_\rho\left(\nabla u_\sigma\right)$ [Men2009].

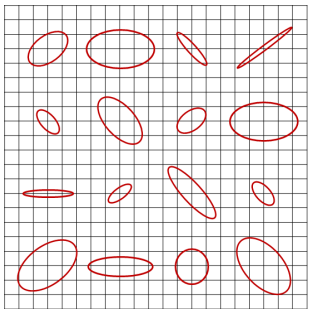


Figure 1: Nonlinear anisotropic diffusion (Adapted from [Li2005a])

Figure 1 shows a nonlinear anisotropic diffusion process illustrated by ovals of different sizes and different orientations. The ratio of two dimensions of the ovals can be arbitrarily different. The orientations of the ovals can also be random. This is the diffusion filter with the ultimate freedom in terms of the changes of filter strength location-wise or direction-wise [Li2005a].

Two specializations of nonlinear anisotropic diffusion were introduced by Weickert, edge-enhancing diffusion (EED) and coherence-enhancing diffusion (CED) [Wei1998]. Both were initially defined in two dimensions. EED was designed to smooth noise while enhancing edges and CED was designed to enhance line-like textures. CED is essentially one dimensional diffusion [Wei1999], since there is either diffusion in one direction or almost not diffusion at all. In addition, Mirebeau et. al. [Mir2014] proposed a conservative variant of both the EED and CED method. These variants are called cEED and cCED respectively.

3 DIFFUSION STOPPING CRITERIA

Filtering process involves the solution of the anisotropic diffusion equations as a time-marching problem, a possible approach is to halt the filtering when a certain set of metrics falls below a predefined threshold [Joã2016]. In addition, the definition of the number of iterations (diffusion time t) based on the metrics selected to stop the diffusion process is crucial to obtain a good image reconstruction [Baz2007]. For example, if t is too small, the reconstructed signal is very noisy; if t is too large it is smooth and discontinuities are lost. In conclusion, automatically stopping the diffusion process is a challenging task. Normally, the stopping criterion depends on the image characteristics and on the parameters of the diffusion equation.

Several authors have addressed this issue in the past in an attempt to devise an optimal stopping criterion [Baz2007, Ily2010]. A brief review of previous works on the stopping criteria is presented below.

- Sporring and Weickert [Spo1999]

  This is focused on the maximal entropy change by scaling to estimate the size of image structures. They argued that the minimal change by scale indicates especially stable scales with respect to evolution time, and conjectured that these scales could be good candidates for stopping times in nonlinear diffusion processes. In addition, this is based on the signal to noise ratio (SNR) and the relative variance at time t and the initial image [Ily2010]. The authors pointed out that the monotonically decreasing 'relative variance', $0 \leq var(u)/var(u_0) \leq 1$, could be used to measure the distance of u from the initial state u_0 and, by prescribing an appropriate value for the relative variance, it can constitute a good criterion for stopping the nonlinear diffusion [Baz2007].

- Capuzzo and Ferretti [Cap2001]

  They determine the optimal time by finding the minimum of a performance index which balances the computing and stopping costs. This is then applied to the regularized Perona-Malik equation. Their method requires a constant that is found by experimentation using a typical image with similar details and discontinuities as the image to be processed. This is a rather vague requirement and they demonstrate that one only needs some approximation to the constant [Ily2010].

- Mrázek and Navara [Mra2003]

  They choose the stopping criteria so that the correlation of the signal $u(T)$ and noise $u(0) - u(T)$ in the filtered image is minimized. This method is applicable to any images where the noise to be removed is uncorrelated with the signal, under the assumptions that the filter used is suitable for the given type of data, and that neither the additive noise nor the filtering procedure alter the average gray value; no other knowledge (e.g. the noise variance, training data etc.) is needed [Mra2003]. This method is applied to several nonlinear filters both isotropic and anisotropic [Ily2010]. In addition, this requires no prior estimation of the noise statistics [Tsi2013].

  Proposed method is called decorrelation criterion. This selects the time T as the time that minimizes the correlation

$$T \equiv \arg\min_t \frac{\operatorname{cov}\left(u\left(0\right)-u\left(t\right),u\left(t\right)\right)}{\sqrt{\operatorname{var}\left(u\left(0\right)-u\left(t\right)\right)\cdot\operatorname{var}\left(u\left(t\right)\right)}} \quad (4)$$

- Gilboa, Sochen and Zeevi [Gil2004]

  Stopping criterion is based on obtaining of minimal SNR, i.e. one stops the process when filtering more signal than noise. This is done by estimating the covariance of the image and the noise. This method requires an estimate of the standard deviation of the noise σ_{n0} of the input noisy image $u\left(0\right)$, which is considered to be a priori known [Tsi2013]. They also compare the advantages and disadvantages of the approaches that use the covariance [Ily2010].

  The condition for selecting the value of parameter T is

  $$T = \arg\min_t \frac{\partial_t \operatorname{cov}\left(\bar{N},u\left(0\right)-u\left(t\right)\right)}{\partial_t \operatorname{var}\left(u\left(0\right)-u\left(t\right)\right)} \quad (5)$$

  The variance of noise $\bar{N}$ of the original image is considered a priori known.

- Bazán and Blomgren [Baz2007]

  This stopping criterion is inspired by observation of the behavior of the correlation between the noise-free image and the filtered image, $corr\left(f,u\right)$, and the correlation between the noisy image and the filtered image, $corr\left(u_0,u\right)$. Although the former measure is only available in experimental settings it helps validate the usefulness of the latter.

  The nonlinear diffusion process starts from the observed (noisy) image, $u_0(x)$, and creates a set of filtered images, $u\left(x,t\right)$, by gradually removing noise and details from scale to scale until, as $t\to\infty$, the image converges to a constant value. During this process the correlation between the noise-free image and the filtered image increases as the filtered image moves closer to the noise-free image. This behavior continues until it reaches a peak from where the measure decreases as the filtered image moves slowly towards a constant value. During the same process the correlation between the noisy image and the filtered image decreases gradually from a value of 1.0 (perfect correlation), to a constant value, as the filtered image becomes smoother [Baz2009].

  By comparing both measures, they observed that as $corr\left(f,u\right)$ reaches its maximum (the best possible reconstructed image), the curvature of $corr\left(u_0,u\right)$ changes sign. They suggested that a good stopping point of the diffusion process is where the second derivative of $corr\left(u_0,u\right)$ reaches a maximum [Baz2009].

- Tsiotsios and Petrou [Tsi2013]

  The method examines directly the quality of the edges in every iteration. It evaluates, in every iteration, the quality of a percentage of the true edges of the image, taking into consideration the contrast and the noise brightness fluctuations around them, and leads to a judicious choice of the stopping time T that corresponds to the maximum overall quality of the edges [Tsi2013]. This method requires an estimate of the standard deviation of the noise σ_{n0} of the input noisy image $u\left(0\right)$, which is considered to be a priori known [Tsi2013]. The proposed method has five steps that finally compute the stop time T as

  $$T = \arg\max_t \frac{1}{N}\sum_{i=1}^{N} Q_i\left(t\right) \quad (6)$$

  where N is the number of edges and $Q\left(t\right)$ reflects the quality of the edges within the image, in every iteration.

- Joao, Gambaruto, Tiago and Sequeira [Joã2016]

  The relative residual error of Mean Square Error (MSE) measure is the metric chosen for this purpose, specifically

  $$\frac{\left|MSE_{t+1}-MSE_t\right|}{\left|MSE_{t+1}\right|} < \varepsilon_1 \quad (7)$$

  where $\varepsilon_1 = 10^{-2}$. The choice of ε_1 is influenced by the need for a small value to identify a convergence of solution, and large enough to make the iterative procedure less computationally demanding.

  In addition, they propose to use Structural Similarity Index Metric (SSIM) in combination with above criterion, using a threshold value of $SSIM < \varepsilon_2$ and $\varepsilon_2 = 0.7$. The choice of ε_2 is influenced by the importance of allowing the image to evolve and deviate from the original, and yet not to allow too large a distortion that will make the image unrecognizable compared to the original.

  In conclusion, the optimal number of iterations is obtained when $\left|MSE_{t+1}-MSE_t\right| / \left|MSE_{t+1}\right| < 10^{-2}$ and $SSIM_\beta\left(t+1\right) < 0.7$.

  This, depending on the size of each image and respective data set, can be rather computationally expensive; therefore, a parallel implementation was used, which proved to be effective.

4 STOPPING CRITERION PROPOSED

Sporring and Weickert pointed out that the monotonically decreasing '*relative variance*', $0 \leq var(u)/var(u_0) \leq 1$, could be used to measure the distance of u from the initial state u_0 and, by prescribing an appropriate value for the relative variance, it can constitute a good criterion for stopping the nonlinear diffusion.

The Contrast Improvement Ratio Revisited measure (CIRR) [Bus2019] is an increasing function that reaches its steady state when $t \to \infty$. The residual error of the CIRR measure is a decreasing function of values between one and zero. Then, applying the same idea presented by Sporring and Weickert, it can be indicated that the residual error of the CIRR measure can be used as a stopping criterion to halt diffusion processes.

The residual error of CIRR measure is computed as:

$$CIRR\ index_t = \frac{|CIRR_t - CIRR_{t-1}|}{|CIRR_t|} \tag{8}$$

where t is the diffusion time. Diffusion process is iterated while $CIRR\ index_t$ is greater that a specific constant ε defined by the user. In general, the stopping criteria is defined as:

$$\frac{|CIRR_t - CIRR_{t-1}|}{|CIRR_t|} > \varepsilon \tag{9}$$

The choice of ε is influenced by the need for a small value to identify a convergence of solution, and large enough to make the iterative procedure less computationally demanding. According to João et. al. [Joã2016], who defined a stopping criterion for anisotropic diffusion, ε value can be 10^{-2}.

5 EXPERIMENTAL RESULTS

Nonlinear anisotropic diffusion algorithms require several parameters such as diffusion time, lambda, noise scale and feature scale. Therefore, to select the most appropriate diffusion algorithm to preserve the edge information is a complex task. For this reason, the experiments were divided into two parts. The first part is related to the automatic definition of the diffusion time for both two-dimensional and three-dimensional images using the stopping criterion presented by Joao et. al. [Joã2016] and the proposed criterion. The second part is associated with the selection of the algorithm that generates better results with respect to the image quality measures as Mean Square Error (MSE), Peak Signal-Noise Ratio (PSNR), and Contrast Improvement Ratio Revisited (CIRR). As a qualitative measure of smoothing, the Canny edge detection filter is used [Can1986, Afr2017]. The filter is applied to the resulting images by using each of the stopping criteria.

5.1 2D Case

In the first part, the stopping criterion proposed by Joao et. al. [Joã2016] is based on the MSE quality measure, this criterion is called JGTS. The proposed criterion is based on the CIRR measure and it is called BF. These two stopping criteria are evaluated.

5.1.1 Original Images

Five images were selected that are used traditionally in image processing. The images are the baboon, barbara, boat, cameraman and lena. Each of them has different characteristics that allow evaluating the quality of the smoothing obtained for each of them according to each stopping criterion.

Initially smoothing is calculated using each of the selected diffusion algorithms (Isotropic, CED, cCED, EED, cEED). The number of iterations (diffusion time) applied is initially set to 10. In each iteration the MSE, PSNR and CIRR quality measures are calculated. The results for the lena image are presented in Tables 1, 2 and 3 respectively.

As you can see in Tables 1, 2, and 3 respectively, the values for the CED, cCED and EED algorithms are quite similar for the MSE measure. In the case of PSNR, the values are very similar in all cases. For the CIRR measure, the values obtained by using CED, cCED and cEED are similar. However, in the latter case, the cEED algorithm is reduced compared to CED and cCED, as the number of iterations increases.

Iter	Isotropic	CED	cCED	EED	cEED
1	2,870	3,731	3,171	7,513	4,652
2	4,127	8,146	7,044	12,618	7,082
3	5,055	12,017	10,623	16,728	8,915
4	5,875	15,537	13,935	20,255	10,498
5	6,650	18,676	16,870	23,364	11,952
6	7,407	21,500	19,628	26,176	13,322
7	8,160	24,122	22,142	28,750	14,650
8	8,916	26,491	24,437	31,136	15,942
9	9,671	28,692	26,626	33,370	17,203
10	10,427	30,790	28,647	35,482	18,442

Table 1: MSE measure - lena.

Iter	Isotropic	CED	cCED	EED	cEED
1	43,205	42,065	42,771	39,025	41,107
2	41,627	38,674	39,305	36,774	39,282
3	40,747	36,986	37,521	35,549	38,282
4	40,093	35,870	36,342	34,718	37,572
5	39,555	35,070	35,512	34,098	37,009
6	39,087	34,459	34,855	33,604	36,538
7	38,666	33,959	34,331	33,197	36,125
8	38,282	33,552	33,903	32,851	35,758
9	37,929	33,206	33,530	32,550	35,427
10	37,602	32,899	33,213	32,283	35,125

Table 2: PSNR measure - lena.

Iter	Isotropic	CED	cCED	EED	cEED
1	0,00016	0,00031	0,00026	0,00055	0,00030
2	0,00027	0,00072	0,00061	0,00106	0,00052
3	0,00036	0,00110	0,00096	0,00149	0,00070
4	0,00044	0,00145	0,00129	0,00186	0,00086
5	0,00051	0,00177	0,00158	0,00218	0,00101
6	0,00058	0,00205	0,00185	0,00246	0,00114
7	0,00065	0,00230	0,00210	0,00272	0,00127
8	0,00071	0,00253	0,00232	0,00296	0,00140
9	0,00078	0,00274	0,00253	0,00317	0,00153
10	0,00084	0,00294	0,00273	0,00337	0,00165

Table 3: CIRR measure - lena.

Figures 2, 3, and 4 show the behavior of the three measures for each of the smoothing algorithms used. The best results are obtained using the Isotropic and cEED algorithms for all cases.

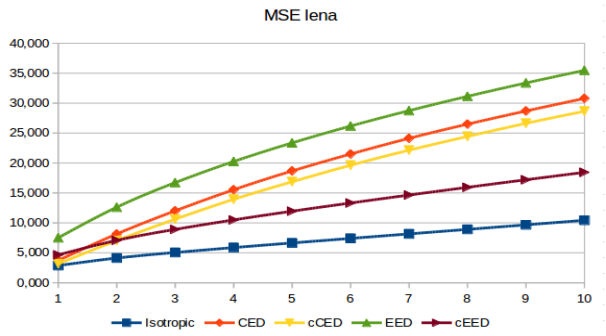


Figure 2: MSE measure - lena.

Table 4 shows the total number of iterations defined by the stopping criteria JGTS and BF for the five reference images. As you can see, the results are quite similar, they differ in one or two iterations. It can also be observed that the isotropic and cEED methods require a greater number of iterations. This is directly related to the MSE, PSNR and CIRR quality measurements obtained.

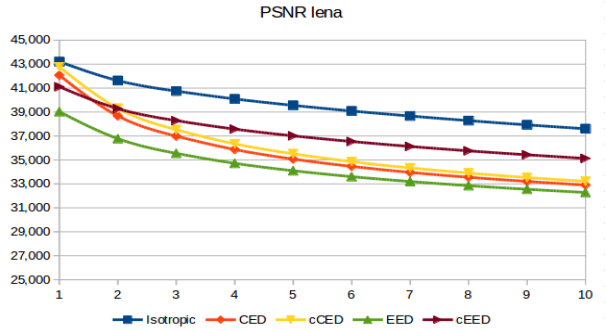


Figure 3: PSNR measure - lena.

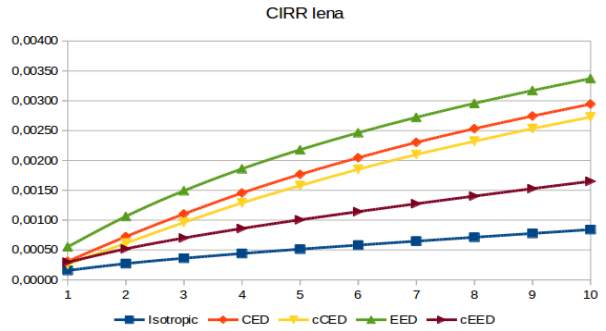


Figure 4: CIRR measure - lena.

Image	Index	Iso	CED	cCED	EED	cEED
baboon	JGTS	10	6	6	5	8
	BF	10	6	7	6	9
barbara	JGTS	10	6	7	6	9
	BF	10	6	6	6	9
boat	JGTS	8	7	7	7	7
	BF	10	7	8	7	9
cameraman	JGTS	8	7	7	6	7
	BF	9	6	7	6	8
lena	JGTS	6	7	7	6	6
	BF	7	7	7	6	7

Table 4: Number of iterations using JGTS and BF stopping criteria.

In the particular case of lena image, Tables 5 and 6 show the values for the MSE and CIRR measurements respectively. It can be indicated that the isotropic method and cEED are those that present a variation. For these

two cases, the number of iterations required by the proposed method (BF) is greater than for the JGTS method.

Iter	Isotropic	CED	cCED	EED	cEED
1	1,0000	1,0000	1,0000	1,0000	1,0000
2	0,3047	0,5420	0,5498	0,4046	0,3432
3	0,1835	0,3221	0,3369	0,2457	0,2057
4	0,1396	0,2266	0,2377	0,1741	0,1507
5	0,1165	0,1681	0,1740	0,1331	0,1217
6	**0,1022**	0,1314	0,1405	**0,1074**	**0,1029**
7	0,0923	**0,1087**	**0,1136**	0,0895	0,0907
8	0,0848	0,0894	0,0939	0,0767	0,0810

Table 5: MSE index - lena.

Iter	Isotropic	CED	cCED	EED	cEED
1	1,0000	1,0000	1,0000	1,0000	1,0000
2	0,4206	0,5728	0,5798	0,4801	0,4308
3	0,2502	0,3440	0,3606	0,2880	0,2599
4	0,1793	0,2412	0,2541	0,1976	0,1854
5	0,1411	0,1765	0,1837	0,1465	0,1447
6	0,1172	0,1363	0,1474	**0,1156**	0,1194
7	**0,1017**	**0,1115**	**0,1176**	0,0946	**0,1037**
8	0,0913	0,0907	0,0961	0,0793	0,0915

Table 6: CIRR index - lena.

Figures 5 and 6 reveal a variation in the behavior of the MSE index between iterations 2 and 3 for the isotropic and cEED methods. This may be the reason why the BF method is more uniform in the number of iterations required to stop the diffusion process.

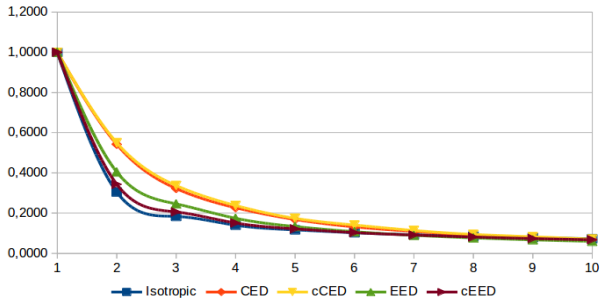


Figure 5: MSE index behaviour for lena image.

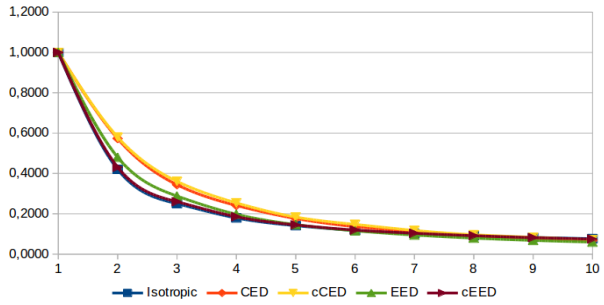


Figure 6: CIRR index behaviour for lena image.

Figure 7: Smoothed image of lena using JGTS and BF stopping criteria.

Table 7 shows the images obtained by applying the selected smoothing methods (Isotropic, CED, cED, EED and cED). Based on the results for the MSE and CIRR measures, the results are the same or similar. Subjectively, it is very difficult to notice the differences.

After exploring the lena image content, the eye region was identified to zoom in and see in a greater level of detail the effect of the smoothing algorithms. Table 8 presents the original image and the images obtained when applying the three algorithms that show a better behaviour with respect to the selected quality measures. As can be seen, the original image differentiates a semi-circular region in the centre of the eye. This region is maintained when applying isotropic smoothing and cEED, however, when cCED smoothing is applied, that region becomes blurred. The latter behaviour is maintained when applying the CED and EED algorithms (see Table 8).

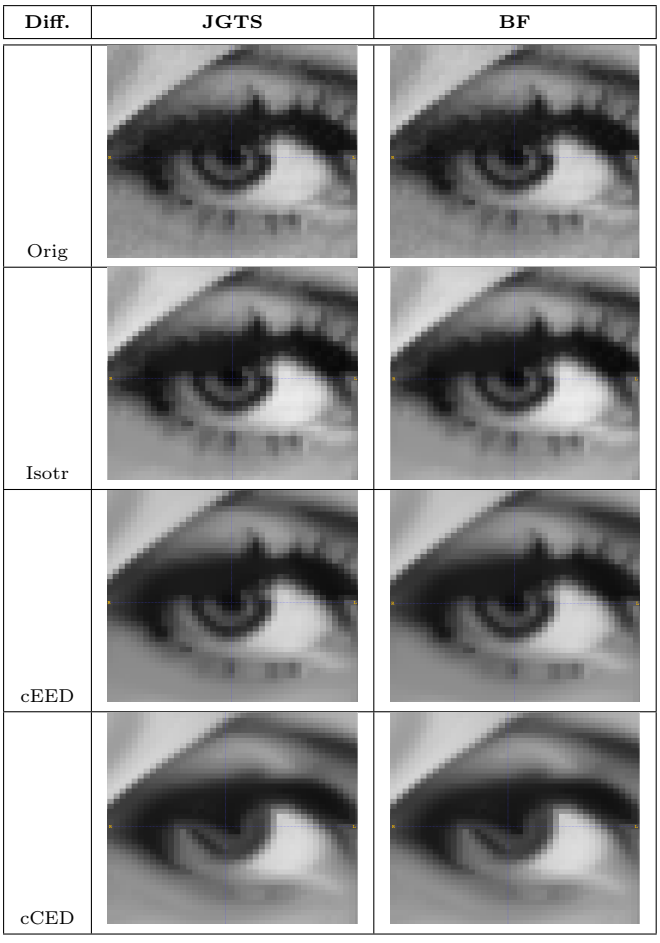


Figure 8: Zoom in of eye region of smoothed image of lena.

In addition, it is observed that the isotropic algorithm presents a lower smoothing in some regions compared to the cEED and cCED algorithms, for example, in the upper left region of the images, it is seen that the isotropic algorithm presents a more stepped variation than the results of the cEED and cCED algorithms.

To visually identify the impact of the smoothing algorithms, row 266 of the lena image was selected. Figure 9 shows the behaviour of the original image and the images obtained from applying each smoothing algorithms. As can be seen, the isotropic diffusion algorithm generates a profile very close to the original image and therefore the image quality measures are better. The cEED algorithm maintains the intensity in the areas where edges are present and in regions with low-intensity variation it makes good smoothing, for example, in the interval [386, 398]. The other smoothing algorithms generate a loss of information at the edges and attenuate their intensity, causing some of them to be eliminated, for example in the intervals [260, 272] and [320, 335].

Based on the elements mentioned above, the initial alternative to smooth the images by preserving the edge information corresponds to the nonlinear anisotropic diffusion algorithm cEED.

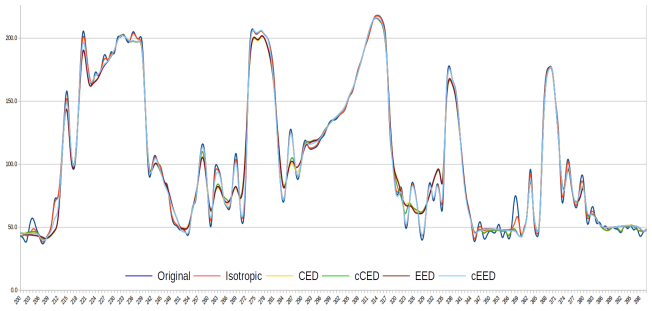


Figure 9: Profile behaviour of smoothing algorithms using lena image.

5.2 3D Case

For the tests with 3D images were selected ten CT images of head-neck. It is proceeded in a similar way to the 2D case, ie, the original images are used first to evaluate the quality measures and to apply the smoothing algorithms in order to identify which stopping criterion performs better. Contrast-enhanced images are then used to identify if there is any change in the behaviour of the smoothing algorithms and in the stopping criteria.

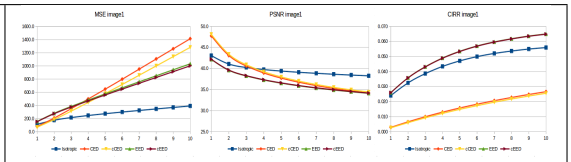
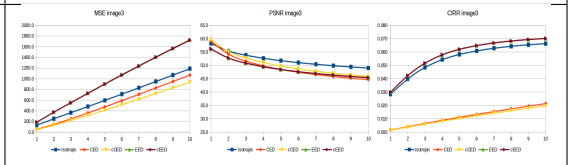
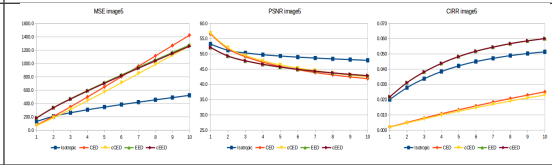
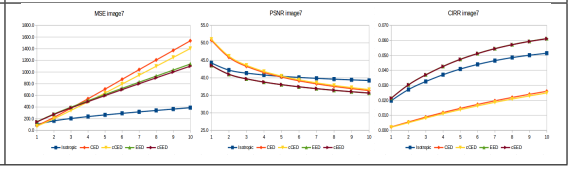
Img	MSE and CIRR behaviour (3D images)
im1	
im3	
im5	
im7	

Table 7: MSE, PSNR and CIRR of 3D smoothed images.

Table 7 shows the behaviour of the MSE, PSNR and CIRR quality measures for images 1, 3, 5 and 7. The algorithms to be selected are those that generate the lowest value for the MSE measure and a higher value for PSNR and CIRR measures. However, it should be considered that as the image is smoothed the value of the PSNR measure decreases, for that reason, the EED and cEED algorithms present a lower value than the other algorithms.

The number of iterations defined by each of the stopping criteria for the ten test images is presented in Table 8. As can be seen, isotropic diffusion is similar using the two stopping criteria except for *image3* and *image4*. In the case of the EED and cEED diffusion, the number of iterations defined by BF criterion is half the number of iterations required by the JGTS criterion. JGTS stopping criterion generates a greater number of iterations required to stop the diffusion in all 3D images with respect to BF criterion. In addition, the number of iterations defined by the BF criterion is the same for the isotropic, EED and cEED diffusion algorithms.

Table 9 shows the images obtained from the smoothing process using stopping criteria JGTS and BF respectively. Visually the differences between the images are not perceptible. To see the impact of the number of iterations in the diffusion algorithms, the edges of the *image1* for the images generated by each of them were calculated. The algorithm proposed by Canny was used for this purpose. As the largest variation in the number of iterations was presented in the EED and cEED diffusion algorithms, it is expected that there is a significant variation in the edges.

Img	Index	Isotr	CED	cCED	EED	cEED
im1	JGTS	5	10	10	8	8
	BF	4	7	7	4	4
im2	JGTS	5	10	10	8	8
	BF	4	7	7	4	4
im3	JGTS	10	10	10	9	9
	BF	4	9	9	4	4
im4	JGTS	10	10	10	10	10
	BF	4	9	9	4	4
im5	JGTS	5	10	10	8	8
	BF	4	8	8	4	4
im6	JGTS	5	10	10	8	8
	BF	4	8	8	4	4
im7	JGTS	5	10	10	9	9
	BF	4	8	8	4	4
im8	JGTS	5	10	10	9	9
	BF	4	8	8	4	4
im9	JGTS	6	10	10	9	9
	BF	4	6	7	4	4
im10	JGTS	6	10	10	9	9
	BF	4	6	7	4	4

Table 8: Number of iterations using JGTS and BF stopping criteria.

The results of the obtained edges are presented in Table 10. In the rows is found each of the diffusion algorithms. In the second column the images obtained by using the JGTS stopping criterion for each diffusion algorithm. In the third column the images using the BF criterion. As can be seen, there is no variation in the edge detection in the images obtained using the EED and cEED diffusion by applying the two stopping criteria. In addition, isotropic diffusion presents results similar to EED and cEED diffusion. On the other hand, the diffusion CED and cCED allow detecting a greater number of edges with respect to the other three algorithms, independent of the stopping criterion used. This is due to the fact that the CED and cCED algorithms apply less smoothing in the internal regions of the image structures.

In conclusion, the edges detected in the images obtained using each one of the diffusion algorithms are equal independent of the stopping criterion used. Therefore, it is considered that the BF stopping criterion is more efficient than the JGTS criterion because it allows stopping the diffusion in a smaller number of iterations.

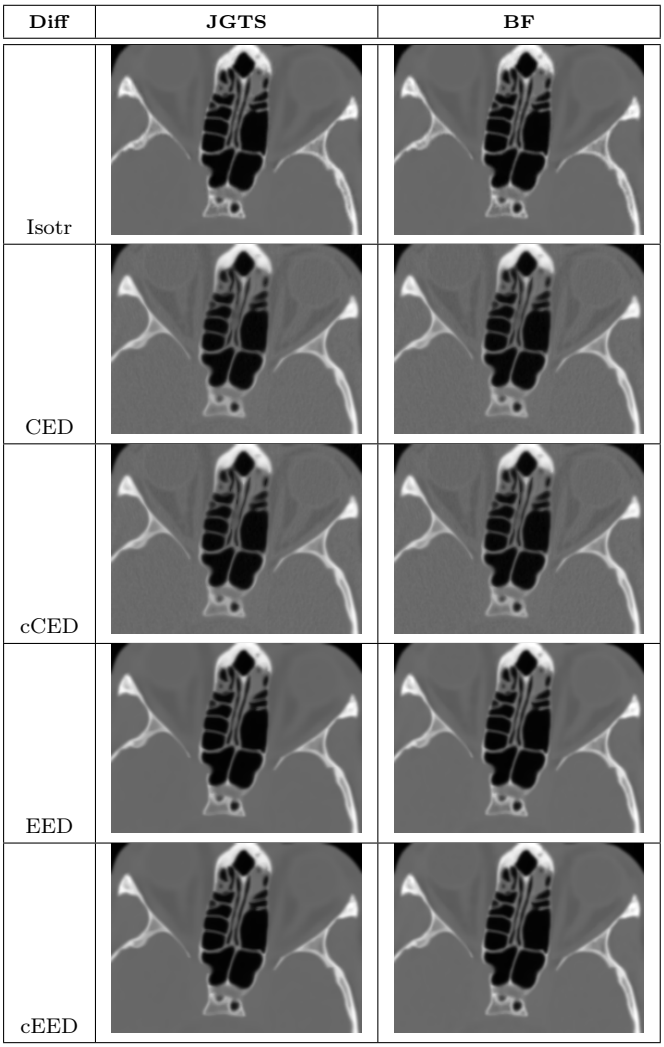


Table 9: 3D images smoothed using MSE and CIRR stopping criteria.

6 CONCLUSIONS

The selected nonlinear diffusion algorithms allowed to define that the edge information is preserved in a better way using the cEED algorithm. Isotropic algorithm also preserves the edges but in the internal regions of the structures does not perform a good smoothing. The CED and cCED algorithms do not properly preserve the edges and generate edges continuity incorrectly.

The proposed BF stopping criterion requires a lower number of iterations in 3D images. This is because the CIRR measure has an asymptotic behavior, while the MSE measure has a more linear behavior. This result allows to increase in automatic way the efficiency of the smoothing algorithms based on nonlinear anisotropic diffusion.

The stopping criterion BF is independent of the smoothing algorithm and it is not necessary to include it in the partial differential equation (PDE).

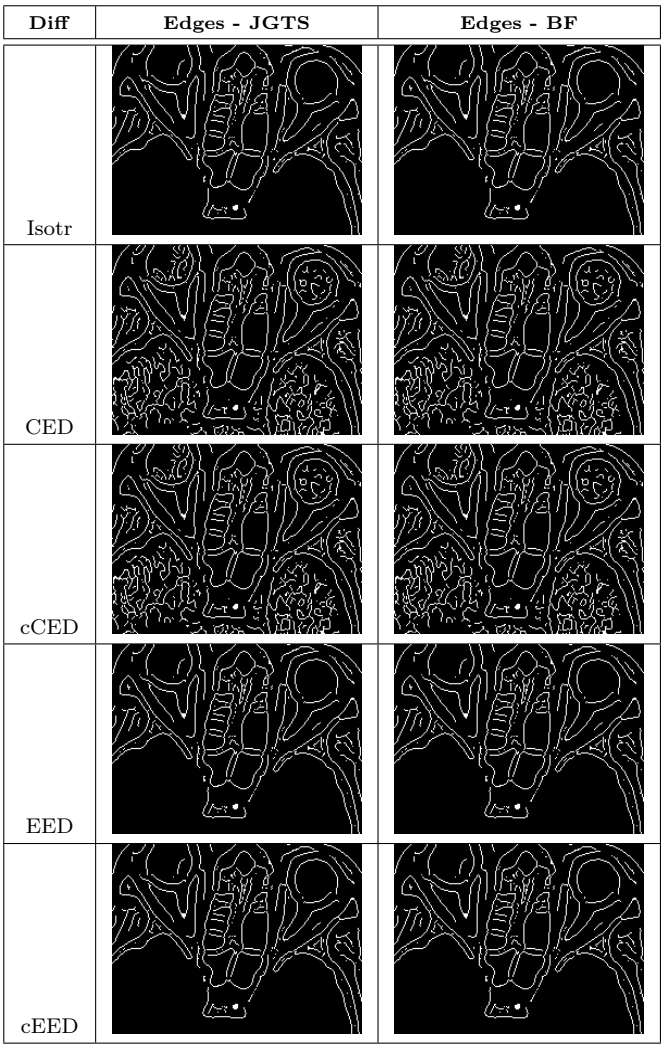


Table 10: Edges of 3D images smoothed.

References

[Afr2017] N. Afrin, W. Lai and N. Mohammed, "Performance Analysis of Corner Detection Algorithms Based on Edge Detectors," in WSCG2017 Proceedings, 2017, pp. 21-28

[Aub2006] G. Aubert and P. Kornprobst, Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations, Second. Springer Science+Business Media, LLC, 2006.

[Bar2014] T. Barbu, "Robust anisotropic diffusion scheme for image noise removal," Procedia Comput. Sci., vol. 35, no. C, pp. 522–530, 2014.

[Baz2007] C. Bazán and P. Blomgren, "Image Smoothing and Edge Detection by Nonlinear Diffusion and Bilateral Filter," Res. Rep. CSRCR, vol. 21, 2007.

[Baz2009] C. Bazán, M. Miller, and P. Blomgren, "Structure enhancement diffusion and contour extraction for electron tomography of mitochondria," J. Struct. Biol., vol. 166, no. 2, pp. 144–155, 2009.

[Ber2000] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," Proc. Conf. Comput. Graph. Interact. Tech., pp. 417–424, 2000.

[Bro2006] T. Brox, J. Weickert, B. Burgeth, and P. Mrázek, "Nonlinear structure tensors," Image Vis. Comput., vol. 24, no. 1, pp. 41–55, 2006.

[Bus2019] C. Bustacara-Medina, L. Flórez-Valencia, An automatic stopping criterion for contrast enhancement using multi-scale top-hat transformation, To appear in Sensing and Imaging, 2019.

[Bus2016] C. Bustacara-Medina, M. Gomez-Mora and L. Flórez-Valencia, "Anisotropic Diffusion for Smoothing: A Comparative Study," Computer Vision and Graphics: International Conference, ICCVG 2016, Warsaw, Poland, September 19-21 Proceedings, pp. 109-120, 2016.

[Can1986] J. Canny, "A Computational Approach to Edge Detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-8, no. 6, pp. 679–698, 1986.

[Cao2003] F. Cao, Geometric Curve Evolution and Image Processing. Berlin Heidelberg: Springer-Verlag, 2003.

[Cap2001] I. Capuzzo Dolcetta and R. Ferretti, "Optimal stopping time formulation of adaptive image filtering," Appl. Math. Optim., vol. 43, no. 3, pp. 245–258, 2001.

[Cas1998] V. Caselles, J.-M. Morel, G. Sapiro, and A. Tannenbaum, "Introduction to the Special Issue on Partial Differential Equations and Geometry-Driven Diffusion in Image Processing and Analysis," IEEE Trans. Image Process., vol. 7, no. 3, pp. 269–273, 1998.

[Cha2010] S. M. Chao and D. M. Tsai, "Anisotropic diffusion with generalized diffusion coefficient function for defect detection in low-contrast surface images," Pattern Recognit., vol. 43, no. 5, pp. 1917–1931, 2010.

[Cot1993] G. H. Cottet and L. Germain, "Image-Processing Through Reaction Combined With Nonlinear Diffusion," Math. Comput., vol. 61, no. 204, pp. 659–673, 1993.

[Don2015] G. Dong, Z. Guo, Z. Zhou, D. Zhang, and B. Wo, "Coherence-enhancing diffusion with the source term," Appl. Math. Model., vol. 39, no. 19, pp. 6060–6072, 2015.

[Erd2012] E. Erdem, "Linear Diffusion," Ankara, 2012.

[Gil2004] G. Gilboa, N. Sochen, and Y. Y. Zeevi, "Image enhancement and denoising by complex diffusion processes," IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 8, pp. 1020–1036, 2004.

[Ily2010] A. Ilyevsky and E. Turkel, "Stopping criteria for anisotropic PDEs in image processing," J. Sci. Comput., vol. 45, no. 1–3, pp. 333–347, 2010.

[Joã2016] A. João, A. M. Gambaruto, J. Tiago, and A. Sequeira, "Computational advances applied to medical image processing: an update," Open Access Bioinformatics, vol. 8, pp. 1–15, 2016.

[Kee2002] S. L. Keeling and R. Stollberger, "Nonlinear anisotropic diffusion filtering for multiscale edge enhancement," Inverse Probl., vol. 18, no. 1, pp. 175–190, 2002.

[Li2005] C. Li, C. Xu, C. Gui, and M. Fox, "Level Set Evolution without Re-Initialization: A New Variational Formulation," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, vol. 1, pp. 430–436.

[Li2005a] X. Li, "The anatomy of anisotropic diffusion filters," Can. Soc. Explor. Geophys., vol. 30, no. 5, pp. 54–60, 2005.

[Men2009] A. M. Mendrik, E. J. Vonken, A. Rutten, M. A. Viergever, and B. Van Ginneken, "Noise reduction in computed tomography scans using 3-D anisotropic hybrid diffusion with continuous switch," IEEE Trans. Med. Imaging, vol. 28, no. 10, pp. 1585–1594, 2009.

[Mir2014] J.-M. Mirebeau, J. Fehrenbach, L. Risser, and S. Tobji, "Anisotropic Diffusion in ITK," Insight J., pp. 1–9, 2014.

[Mra2003] P. Mrázek and M. Navara, "Selection of optimal stopping time for nonlinear diffusion," Int. J. Comput. Vis., vol. 52, no. 2–3, pp. 189–203, 2003.

[Nit1992] M. Nitzberg and T. Shiota, "Nonlinear Image Filtering with Edge and Corner Enhancement," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 8. pp. 826–833, 1992.

[Per1990] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," IEEE Trans. Pattern Anal. Mach. Intell, vol. 12, no. 7, pp. 629–639, 1990.

[Per1990a] P. Perona and J. Malik, "Detecting and localizing edges composed of steps, peaks and roofs," in Proceedings Third International Conference on Computer Vision, 1990, pp. 52–57.

[Pra2016] S. Prasath, "Adaptive coherence-enhancing diffusion flow for color images," Inform., vol. 40, no. 3, pp. 337–342, 2016.

[Rud1989] L. Rudin and S. Osher, "Feature-Oriented Image Enhancement with Shock filters, I," 1989.

[Sap2006] G. Sapiro, Geometric Partial Differential Equations and Image Analysis. Cambridge University Press, 2006.

[Spo1999] J. Sporring and J. Weickert, "Information Measures in Scale Spaces," IEEE Trans. Inf. Theory, vol. 45, no. 3, pp. 1051–1058, 1999.

[Ter1994] B. M. ter Haar Romeny, Geometry-Driven Diffusion in Computer Vision, vol. 1. Springer-Science and Business Media, 1994.

[Tsi2013] C. Tsiotsios and M. Petrou, "On the choice of the parameters for anisotropic diffusion in image processing," Pattern Recognit., vol. 46, no. 5, pp. 1369–1381, 2013.

[Wei1996] J. Weickert, "Theoretical Foundations of Anisotropic Diffusion in Image Processing," Comput. Suppl., vol. 11, pp. 221–236, 1996.

[Wei1997] J. Weickert, S. Ishikawa, and A. Imiya, "Scale-Space has been Discovered in Japan," 1997.

[Wei1998] J. Weickert, Anisotropic diffusion in image processing, vol. 256, no. 3. B.G. Teubner Stuttgart, 1998.

[Wei1999] J. Weickert, "Coherence-Enhancing Diffusion Filtering," Int. J. Comput. Vis., vol. 31, no. 2, pp. 1–23, 1999.

[Wei2002] J. Weickert and H. Scharr, "A Scheme for Coherence-Enhancing Diffusion Filtering with Optimized Rotation Invariance," J. Vis. Commun. Image Represent., vol. 13, no. 1–2, pp. 103–118, 2002.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.27, No.1, 2019

Floor Map Visualizations of Medical Volume Data

Nico Merten Sylvia Saalfeld Bernhard Preim
1st Affiliation: Department of Simulation and Graphics, Otto-von-Guericke University
2nd Affiliation: Research Campus *STIMULATE*
Universitätsplatz 2
D-39106, Magdeburg, Germany
[nmerten, sylvia, preim]@isg.cs.uni-magdeburg.de

ABSTRACT

Typically, volumetric medical image data is examined by assessing each slice of an image stack individually. However, this enables observers to assess in-plane spatial relationships between anatomical structures only and requires them to keep track of relationships along the third anatomical plane mentally. Therefore, visualization techniques are researched to support this task by depicting spatial information along the third plane, but they can introduce a high degree of abstraction. To overcome this, we present a novel approach that transforms image stacks with labeled anatomical structures into maps with a three-dimensional layout, namely floor maps. Since this approach increases the visual complexity under certain conditions, some clinical application scenarios, e. g. diagnosis and therapy planning, probably will not benefit. Thus, the approach is mainly aimed to support student training and the generation of clinical reports. We also discuss how to enhance the slice-based exploration of medical image stacks via floor maps and present the results of an informal evaluation with three trained anatomists.

Keywords

Floor Maps, Exploration Support for Medical Volume Data, Abstraction

1 INTRODUCTION & MOTIVATION

Over the last decades, much research was carried out to improve medical image scanners, such as *Computed Tomography* (CT) or Magnetic Resonance Imaging (MRI) scanners, e. g. with respect to spatial resolution. Thus, less anatomical information is compiled and mapped into individual *Volume Elements* (voxels). On the one hand, this improved the versatility and fidelity of the image data. On the other hand, this results in more visual information that has to be assessed. Generally, this is achieved by axial slicing, e. g. via computer mouse scrolling. The main disadvantage is that users have to understand three-dimensional spatial relationships between anatomical structures although only in-plane relationships are depicted. Especially for medical students in training this can become mentally exhausting, because they are not yet accustomed to these tasks.

The main contribution of this work is a processing pipeline that can transform labeled, medical image stacks into interactively explorable floor maps. These maps are a well-known concept to provide spatial information in large, multi-story buildings, such as malls, hospitals, or federal buildings [11]. These maps aim to use the natural exploration-supporting properties of maps while introducing only a moderate degree of abstraction. Many people are used to maps, e. g. from various handheld devices. Additionally, maps support our visuo-spatial working memory, which is a mental resource that we require for orientation and navigation tasks in spatial environments. Furthermore, since CT or MRI scans are multiple, stacked images, their data layout is three-dimensional and the visual layout of maps can be three-dimensional, too. Finally, maps and medical image visualization techniques purposefully abstract and simplify geometric details. This can be beneficial for anatomical education and report generation, since, in the beginning, understanding spatial relationships between anatomical structures is more important than learning geometric details, and the generation of easy-to-understand and interpret documentations of findings is an important task in every clinical workflow. Therefore, we hypothesize that floor maps are suitable to offer exploration support for medical image data.

2 RELATED WORK

One downside of the slice-based exploration and assessment of image stacks is that only the in-plane spatial relationships between structures are depicted. Thus, physicians have to keep track of spatial relationships along the third anatomical plane mentally. To support this task, 2D and 3D visualization techniques were developed that,

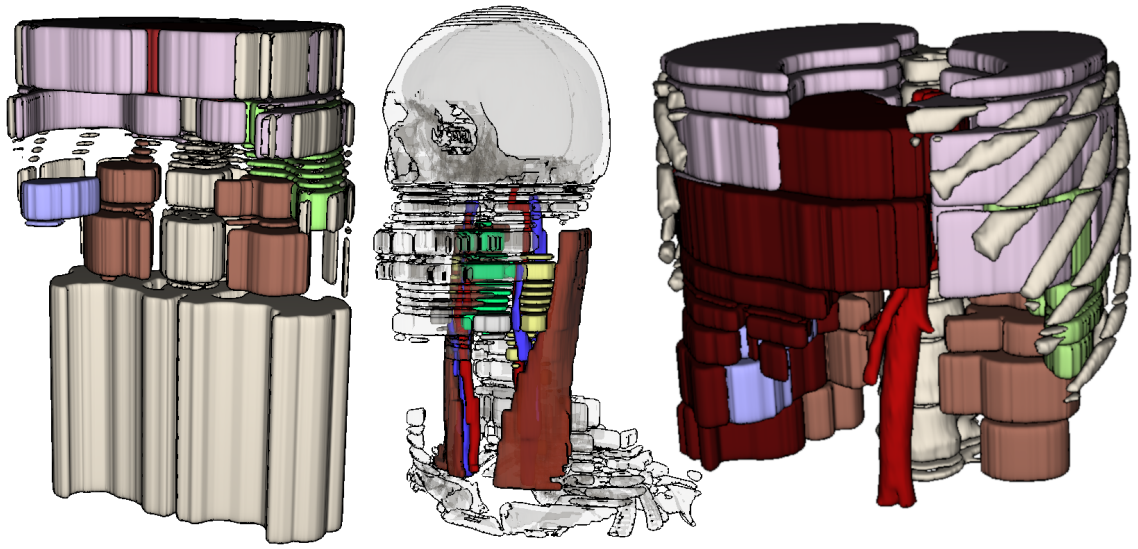


Figure 1: Floor map visualizations depicting three image data sets (cf. Tab. 1). **Left (DS1)**: A standard floor map is depicted. **Middle (DS2)**: Various structures, such as the two cervical muscles (brown), the arteries (red), and veins (blue), were re-merged. **Right (DS3)**: The aorta and lateral ribs were re-merged while also preserving their shape.

in general, depict additional spatial information of the explored and assessed image stack along the third plane.

Exploration Support. Tietjen et al. [17] presented a 2D technique called *lift charts*: While an image stack is explored in a 2D view, the spatial extent of labeled structures along the third plane is depicted in an additional 2D chart. The result is an adapted bar chart with multiple vertical bars that can be interpreted as lifts that make it possible to distinct vertical sections. Generally,

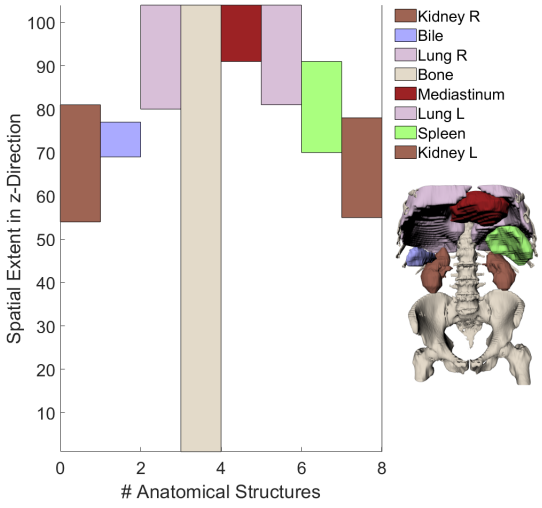


Figure 2: A *lift chart* to support the slice-based exploration of medical volume data as introduced by Tietjen et al. [17]. All labeled anatomical structures are represented via bars and their vertical positions and extents are defined by the z_{min} and z_{max} coordinates of their respective structure's Axis-Aligned Bounding Box. The respective data set is presented on the right.

lifts are defined along the z-axis, because tomographic scans are acquired and assessed in axial slices. However, lift charts can be applied for arbitrary projection planes. Furthermore, a unique color-coding is assigned to each structure type, such as bones and lymph nodes. The lift chart in Figure 2 was generated using *Data Set 1* (see Tab. 1). The respective floor map can be seen in the leftmost subfigure of Figure 1.

Later, Balabanian et al. [3] used lift charts for their hierarchical graph network, which enabled them to use lift charts for anatomical substructures, too. Thus, there now exist multiple charts for one data set: Depending on the currently observed hierarchy level, a chart presents spatial relationships between structures and substructures at different scales. Diepenbrock et al. [7] extended lift charts to present information from scans of different modalities: Rather than using morphological scans only, their lift charts also present information from *functional MRI* (fMRI) scans like a graph plot.

Although lift charts are a straightforward visualization technique to enhance the slice-based exploration of image stacks, their overall degree of abstraction is very high. Following the theory of Viola and Isenberg [19], lift charts would rank high on the *geometric* and *photometric abstraction* axes, because only two values and one color are presented per structure. Estimations of the individual degrees of abstraction for lift charts and floor maps are depicted in Figure 3. When considering real-world anatomy as the starting point and biochemical processes at the end point of the *scale* axis, lift charts and floor maps are located at a very low point, because they are used for high-level morphological information. Moreover, since only one scan is used, there is no temporal information; thus, both techniques introduce no

temporal abstraction. Techniques with a low *photometric abstraction* use complex light propagation models, whereas more photometrically abstract techniques use simpler shading or stylizations, such as flat shading or line drawings. When techniques introduce a low degree of *geometric abstraction*, many shape details are preserved, whereas objects will get very simplified when a high degree of geometric abstraction is used. The degrees of photometric and geometric abstraction are very high for lift charts, since only their vertical extent is preserved and presented in a simple color-coding. Thus, we hypothesized that floor maps are more appropriate to offer exploration support for medical volume data, because they preserve more geometric details, such as the axial shape of structures, and present them using ambient and diffuse shading, which further emphasize valleys between floors and rooms.

Mindek et al. [14] presented a different type of exploration support by virtually altering the slicing speed during 2D exploration via non-linear interaction. They define *representative slices* if chosen structures undergo large morphological changes between adjacent slices, for example, if the cross-section areas of blood vessels change or if they branch. During 2D exploration, only these representative slices are presented to the user, which results in a *slower* exploration in regions with large changes, because there are more representative slices. Similarly, in image regions with many morphological changes our method generates more floors and rooms. Consequently, both methods generate only few representative slices or large floors with few rooms in image regions with only small or no changes.

Digital 3D Maps. Research for digital maps focuses on

- pathway planning in emergency situations [6],
- generation of 3D models from 2D drawings [20],
- level-of-detail techniques for indoor maps [13], or
- the ontological description of buildings [12].

Floor maps help users to familiarize with complex spatial layouts, such as offices, high-rise buildings, and malls [11]. To exploit this potential and to prevent visual clutter, special care is necessary. For example, color scales with only few colors that can be easily distinguished have to be defined and occlusion problems have to be controlled, e. g. by generating good default views on building models. Our approach enables users to interactively adapt the color-coding and opacity of structures.

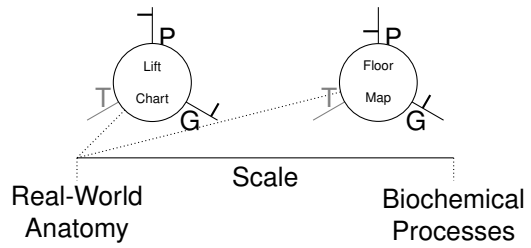


Figure 3: Ranking the lift chart and floor map visualization techniques using the theory of Viola and Isenberg [19]. The radiant axes show the **P**hotometric, **G**eometric, and **T**emporal abstraction categories. Axes ticks show the estimated degrees of abstraction in the respective category. The T category is not applicable.

3 FLOOR MAP GENERATION

In this section the data sets and processing pipeline to create interactive 3D floor map visualizations from medical image volumes are described. Figure 4 depicts the proposed pipeline to transform segmentation masks into an interactively explorable floor map visualization. After presenting the conceptual design, each step of the pipeline is described in Section 3.3.

3.1 Image Data and Implementation

To develop the proposed method, three CT scans were used (see Tab. 1). The processing pipeline, which transforms label images into floors and rooms, was implemented in *MeVisLab 2.8.2* [15] (see Fig. and Sec. 4). Subsequently, users can use a *Graphical User Interface* (GUI) (see Fig. 9) to simultaneously explore the unaltered image stacks and floor maps.

3.2 Conceptual Design

When applying the floor map concept on medical volume data, we discussed how human anatomy can be transformed into floors and rooms. Generally, larger structures, such as organs, have a deformed, spherical shape with soft edges. There are some exceptions, such as extremity bones or intestines, which, while also having a round, have a rather elongated shape. Smaller structures, such as blood vessels, are tubular with circular or ellipsoid shaped cross-sections.

In contrast, buildings are man-made structures with considerable regularity. For most buildings, the ground plan is extruded vertically and divided horizontally into equally high floors. Thus, they often have a cuboid surface. However, especially for smaller buildings, such as homes, the inner floor layout can be very individual with varying room sizes, whereas for taller buildings, the floor layout can be very repetitive. Additionally, pathways, such as corridors, staircases and elevators, are straight, horizontally or vertically oriented tubes.

In clinical practice, physicians typically assess image stacks via axial slicing. Therefore, the proposed approach divides image stacks along the z-axis to create floors, and to make them clearly visually distinguishable, small gaps are inserted between adjacent floors. However, although the gap size is adjustable, each division results in a geometric distortion. Therefore, the number of divisions should be minimal.

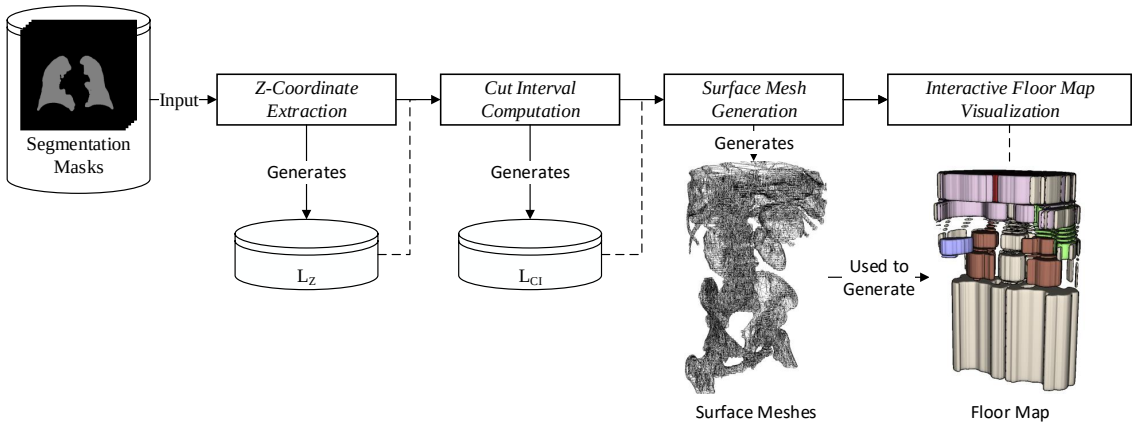


Figure 4: Pipeline to generate floor map visualizations from medical image stacks. L_Z is a compiled list of unique z_{min} and z_{max} coordinates of all segmentation masks. L_{CI} is a list of cut intervals that are used to divide the segmentation masks into floors.

To create building-like maps, rooms have to be created. Anatomical structures can have complex spatial arrangements, e. g. the heart is encompassed by the lungs, but they are clearly separated. This feature is considered in the processing pipeline by creating floors and rooms that are free from overlaps (cf. Figures 6 and 7). However, to enable an easy visual grouping of rooms that represent similar anatomical structures, they are color-coded identically. For example, in Figure 4 the kidneys and lungs are colored brown and pink, respectively.

3.3 Image Stack Processing

Z-Coordinate Extraction. In the first step of the processing pipeline, the z_{min} and z_{max} coordinates of all labeled structures' axis-aligned bounding boxes are extracted and compiled into a list named L_Z. Before the coordinates are compiled, certain structures can be marked as being *too small* for processing, if their spatial extent in z-direction ($z_{max} - z_{min}$) is below a user-defined threshold. Thus, they will not be divided into floors and their shape will be preserved. However, due to gaps between floors, they have to be moved to their correct vertical position for the final visualization. In Figure 5, lymph nodes are depicted that were marked *too small*. This *protects* very small structures from any geometric abstraction and distortion, which would be larger for them than for larger anatomical structures.

Cut Interval Computation. The compiled z-coordinates are then used to compute *cut intervals* to define floors. Similar to the approach of Mindek et al. [14], floors are defined when the composition of segmentation masks between adjacent slices changes. This results in unique floors and visually guides users to regions with high anatomical variability. Therefore, first, L_Z is sorted in ascending order and double coordinates are removed since the method only needs knowledge about slices in which the composition changes, but not about which segmentation masks are the reason. For each remaining entry $e_k \in L_Z$ only one of the following statements (S1-S3) is correct:

$$\begin{aligned} &\text{S1} && e_k = z_{min_{S_i}} \\ &\text{S2} && e_k = z_{max_{S_i}} \\ &\text{S3} && e_k = z_{min_{S_i}} = z_{max_{S_j}} \end{aligned}$$

DS	DS size (voxels)	# S	Site of Scan
DS 1	512 × 512 × 105	8	lower thorax to pelvis
DS 2	513 × 513 × 108	22	head and neck area
DS 3	512 × 512 × 99	9	lower thorax and upper abdomen

Table 1: Details of the used CT data sets (DS) with their respective number of voxels, the labeled structure count (# S), and the anatomical site of the scan (cf. Fig. 1).

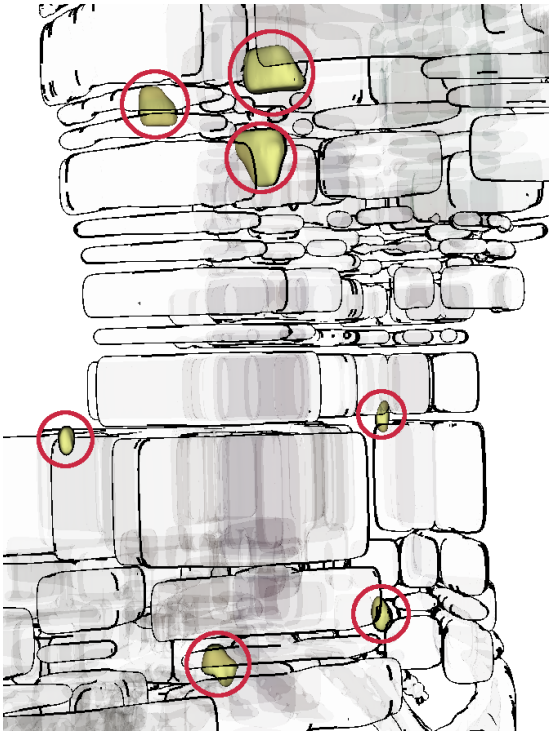


Figure 5: Lymph nodes that are flagged too small, which protects them from geometric abstraction and distortion.

For each $e_k \in L_Z$ there exist structure segmentations S that either start in slice e_k (S1), end in e_k (S2), or at least one starts while at least another ends (S3). Subsequently, all $e_k \in L_Z$ have to be flagged accordingly. Using an extended interval notation and L_Z of Data Set 1, the following line enables a quick understanding on how cut intervals are defined using our approach:

$$\underset{0}{[}\ldots\underset{54}{[}\ \underset{55}{[}\ldots\underset{69}{[}\ \underset{70}{[}\ldots\underset{77}{]}\ \underset{78}{]}\ldots\underset{80}{[}\ \underset{81}{|}\ldots\underset{91}{|}\ldots\underset{104}{]}$$

For each entry e_k, the color-coding and token show which statement is true: A red left square bracket is used for S1, a blue right square bracket is used for S2, and a violet bar is used for S3.

Subsequently, all entries are processed pairwise. Depending on which statements are true for *this* and the *next* entries e_k, the respective cut interval will be defined differently. There exist nine combinations, because, e. g., e_k can fulfill S1 while e_{k+1} can fulfill S1, S2, or S3. For example: For the slices 0 and 54 the composition of segmentation masks does not change until slice 54. Thus, the first floor will be defined from slice 0 to 53 and slice 54 will be processed in the *next* step. For 70 and 77, slice 77 can be included in *this* processing step and has to be skipped in the *next* step. Applying the method shown in Algorithm 1 to Data Set 1, the resulting list of cut intervals L_{CI} will be:

$$\begin{aligned} L_{CI} = \{&[0,53],[54,54],[55,68],[69,69],\\ &[70,77],[78,78],[79,79],[80,80],\\ &[81,81],[82,90],[91,91],[92,104]\} \end{aligned}$$

The algorithm produces the smallest number of floors with no double slices in adjacent intervals. Thus, visual distortion is minimal and the linking between original images and floor map is unique for each slice, which is important for the simultaneous exploration later. Finally, to create visual gaps between floors, empty slices are created between L_{CI} intervals before mesh creation. We found that one or two slices are sufficient, because larger gaps increase the need that users have to navigate through the final visualization manually. This would be unfavorable, because medical experts are not as well used to 3D interactions as computer graphic experts.

Vertex Mesh Generation. To create meshes, the *Neighboring Cells Algorithm* is used [2]. However, before that, rooms have to be created inside the previously defined floors: For each floor and all segmentation masks therein, this is achieved via projection and extrusion (see Fig. 6): First, all masks are projected along the z-axis. Secondly, this shape is extruded along the floor's height. This results in rooms that are defined by the maximum axial shape of their respective structures.

This approach preserves some geometric features of the anatomical structures, which, in combination with their vertical position, supports individual recognizability. However, it also creates overlapping artifacts between rooms. This can be seen in Figure 6. Although there exists no overlap between the segmentation masks S_1 and S_2, extruding their maximum axial shape results in the overlapping rooms R_1 and R_2.

```
Input: L_Z

for all e ∈ L_Z
if L_Z(e) == [
|   if L_Z(e+1) − L_Z(e) == 1
|   |   if L_Z(e+1) == [ or L_Z(e+1) == |
|   |   |   L_CI(e) = [L_Z(e), L_Z(e)]
|   |   elif L_Z(e+1) == ]
|   |   |   L_CI(e) = [L_Z(e), L_Z(e+1)]
|   |   endif
|   else
|   |   if L_Z(e+1) == [ or L_Z(e+1) == |
|   |   |   L_CI(e) = [L_Z(e), L_Z(e+1)−1]
|   |   elif L_Z(e+1) == ]
|   |   |   L_CI(e) = [L_Z(e), L_Z(e+1)]
|   |   endif
|   endif
endif

if L_Z(e) == |
|   L_CI(e) = [L_Z(e), L_Z(e)]
|   e = e+1
|   if L_Z(e+1) − L_Z(e) == 1
|   |   continue
|   else
|   |   if L_Z(e+1) == [ or L_Z(e+1) == |
|   |   |   L_CI(e) = [L_Z(e)+1, L_Z(e+1)−1]
|   |   elif L_Z(e+1) == ]
|   |   |   L_CI(e) = [L_Z(e)+1, L_Z(e+1)]
|   |   endif
|   endif
endif

if L_Z(e) == ]
|   if L_Z(e+1) − L_Z(e) == 1
|   and L_Z(e) != L_CI(e−1)[1]
|   |   L_CI(e) = [L_Z(e), L_Z(e)]
|   |   continue
|   elif L_Z(e) − L_Z(e−1) == 1
|   and L_Z(e+1) − L_Z(e) > 1
|   |   L_CI(e) = [L_Z(e), L_Z(e)]
|   |   e = e+1
|   endif
|   if L_Z(e+1) == [ or L_Z(e+1) == |
|   |   L_CI(e) = [L_Z(e)+1, L_Z(e+1)−1]
|   elif L_Z(e+1) == ]
|   |   L_CI(e) = [L_Z(e)+1, L_Z(e+1)]
|   endif
endif
endfor

Output: L_CI
```

Algorithm 1: Cut Interval Generation.

This issue is resolved by performing pairwise overlapping tests between all rooms. If overlaps exist, the overlapping volume is assigned to the smaller room. This decision was taken, because, if rooms overlap, smaller rooms are already at a higher risk to be overlooked. Thus, their visibility is increased. In Figure 6, this would be R_2 and the red-colored mediastinum, respectively.

After overlapping artifacts are resolved, processed rooms are very close to each other. Therefore, to increase visual separability of adjacent rooms, surface meshes are smoothed, e. g. via Laplace smoothing. The result is depicted in Figure 7: After overlap removal, with respect to the used color-coding, the mediastinum and lungs can be clearly distinguished. However, the visual transition is abrupt, which can become a problem if rooms with similar colors are too close to each other. Applying mesh smoothing creates valleys between adjacent rooms and in combination with diffuse shading, the visual separability is further increased.

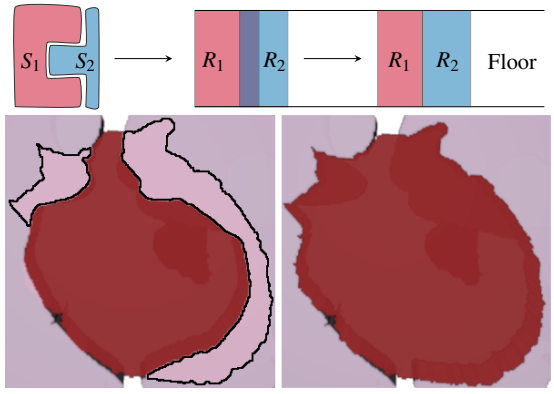


Figure 6: **Top**: To create rooms, the maximum axial shape of each segmentation mask S_i is extruded along a floor. This can result in overlap artifacts between rooms R_i. This problem is addressed by assigning overlapping volume to the smaller room. **Bottom**: The mediastinum (red) and the lungs (rose) overlap laterally. The boundaries of the overlapping volumes are emphasized for better visibility. Since the mediastinum is smaller than both lungs, the overlapping volumes are assigned to it.

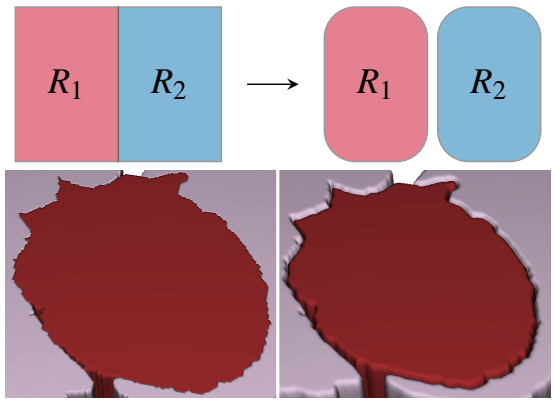


Figure 7: After extrusion and overlap removal, adjacent rooms are contiguous. Therefore, Laplacian smoothing is applied, which results in volume shrinkage. This creates gaps between adjacent meshes and *valleys* around the top and bottom edges. In combination with diffuse shading, rooms become more visually separated.

Before mesh generation, users have two options to alter the transformation of individual segmentation masks into floors and rooms. First, the shape of structures can be preserved. This means that while the cut intervals in L_{CI} are still used to create individual floors, not their maximum axial shape but the unaltered segmentation mask is used to create rooms. Secondly, masks can be protected from being divided into floors. To do this, a morphological dilatation operator in z-direction is used to re-merge vertical gaps that are a result of empty slices that are created between adjacent cut intervals. Both options can be combined, which is depicted in Figure 8: Here, the *musculi sternocleidomastoideus* are protected from divisions and geometric abstraction into rooms.

Another option that was included is that all structures in the lowest or highest floor can have their shape preserved. This can be seen in Figure 1: In the leftmost subfigure, the lowest and highest floors were processed normally. Due to their large vertical extent, they introduce a large

Figure 8: Users can preserve the shape of segmented structures and re-merge gaps. Here, both options were combined for the *musculi sternocleidomastoideus*.

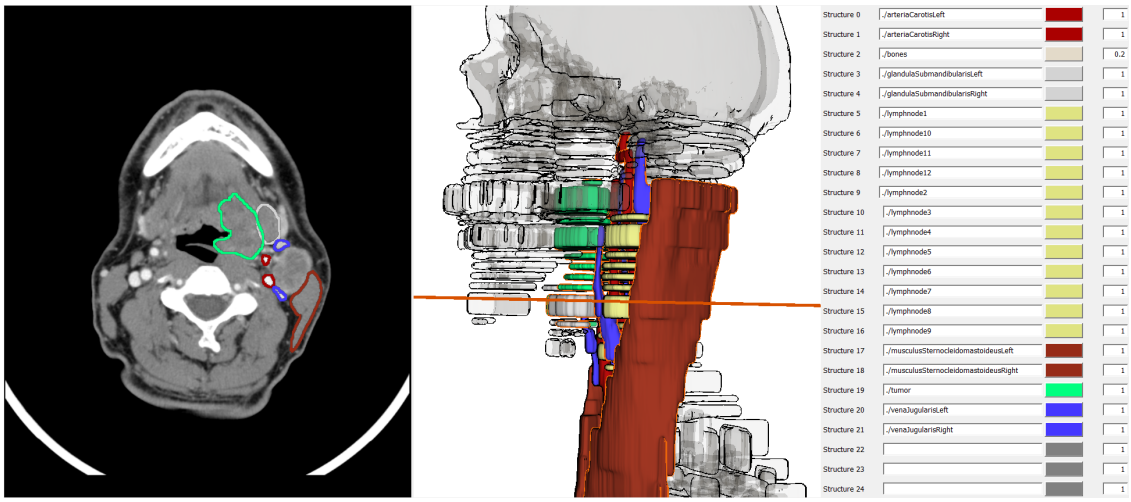


Figure 9: The GUI of the combined 2D view and floor map visualization. A tumor (green), and the left *glandula submandibularis* (white), carotis (red), and jugular vein (blue) are highlighted in 2D and 3D. The orange-colored frame corresponds with the currently displayed slice. On the right, the names, colors, and opacities are shown for all structures. Users can interactively adjust these options to change the floor map visualization.

geometric distortion. In contrast, for the middle subfigure, the shapes of the upper skull and collarbones were preserved, which can increase the recognizability of the depicted anatomical site considerably.

Interactive Floor Map Visualization. Figure 9 depicts the GUI that enables an interactive and simultaneous exploration of the original images and floor map. In the 2D view on the left, users can slice up and down through the original image stack, e. g. via mouse wheel scrolling. In the floor map, an orange-colored frame moves up and down accordingly that shows which floor the currently displayed slice belongs to. Note that this frame has to *jump* over gaps in the floor map, because the original image stack does not have empty slices that are virtually inserted to create individual floors. Moreover, users can select room meshes and the boundaries of the respective segmentation masks are highlighted in the 2D images. In Figure 9, various structures, such as a tumor (green), arteries (red), and veins (blue) were selected. Additionally, a geometry-based contour shading is used to highlight the 3D contours of selected rooms. To control occlusion problems, the opacities of all rooms can be adjusted on the right using order-independent transparencies [4].

Category	Data Set (# S/ # F) DS 1 (9/ 15)	DS 2 (22/ 28)	DS 3 (9/ 15)
Z-Coordinate Extraction	0.94	2.51	1.52
Cut Interval Computation	0.01	0.05	0.01
Vertex Mesh Generation			
Room Extrusion	12.66	32.94	10.93
Overlap Removal	21.68	47.64	14.34
Mesh Generation	29.65	64.01	30.92
Mesh Smoothing	2.43	3.08	4.41
Total Processing Time	67.37	150.23	62.13

Table 2: Computation times in seconds for each processing step from Figure 4. For each data set, the number of structures and the number of floors are given with # S and # F. No structure was marked *too small*, no shape was preserved, and no rooms were re-merged.

4 RESULTS

The main result of this work is a pipeline that transforms labeled, medical image stacks into interactively explorable floor maps (see Fig. 4). Algorithm 1 is the core of the proposed method and it produces the smallest possible number of floors. Because double coordinates are removed during cut interval computation, there exist no ambiguities when the original images and the floor map are explored. In Table 2, the computation times of all processing steps are shown. They were acquired using an i5-2500 processor with 3.30 GHz.

4.1 Evaluation

To evaluate the proposed approach, three trained anatomists were interviewed informally and each of them filled out a questionnaire. Two interviewees are physicians (I1 and I2) and one is a biologist (I3). Similar to Figure 9, they were given a software prototype to simultaneously explore CT image stacks of three data sets (cf. Tab. 1 and Fig. 9) with either lift charts or floor maps. The interviewees were given a five-point Likert scale ($--$, $-$, o, $+$, $++$) to answer questions. Their answers are compiled in Table 3 and are presented using a diverging red-white-blue color-scale. The questionnaires were divided into three parts, which will be explained in the following paragraphs and a discussion will be provided in the next section.

In the first part, the interviewees were asked to provide information about their clinical and technical experience. Their answers are shown in the upper part of Table 3:

Q 1 How do you rate your anatomical knowledge?

Q 2 How familiar are you with medical visualization techniques in general?

Q 3 How familiar are you with exploration-assisting visualization techniques for medical images?

Q 4 How do you rate your spatial thinking capabilities?

For the second part, the interviewees were given instructions on how to interact with the elements of the software prototype. First, they were asked to explore the CT data sets with lift charts (cf. Fig. 2). The currently depicted slice was represented as a horizontal line in the lift chart. Subsequently, the interviewees explored the data sets with floor maps. After each exploration, they were asked to rate the exploration-assisting capabilities of each visualization technique (VT):

Q 5 How easy is it for you to get a first overview of the data set using VT?

Q 6 How easy is it for you to assess spatial relationships between anatomical structures using VT?

Q 7 How much does VT support you to find anatomical abnormalities, e. g. a tumor?

Q 8 How easy is it for you to use VT for orientation?

Q 9 How fast do you recognize segmented structures using VT?

Q 10 Overall: How much do you like using VT?

For the last part, they were asked how feasible they think each VT is for various clinical application areas.

5 DISCUSSION

Method Discussion. The presented pipeline requires labeled images and does not include segmentation algorithms. Depending on the data set (cf. Tab. 1), the available structure segmentations were obtained using different segmentation algorithms. For example, the liver in DS 3 was segmented using *HepaVision2* from Bourquain et al. [5], which uses a semi-automatic livewire approach. This results in a binary mask that can immediately be processed with our method. In contrast, the lymph nodes and blood vessels in DS 2 were segmented using the model-based methods of Dornheim et al. [8, 9] to support the treatment planning for neck dissection surgeries. These methods produce polygonal representations of segmented structures, which requires a conversion into binary masks. This is not ideal, since this conversion degrades the quality of segmentation results to some extent. However, this is not an issue, because the presented method was not developed with this use case in mind.

Although healthy anatomical structures are clearly separated from each other, obtaining segmentation masks involves some degree of uncertainty. This uncertainty is usually increased around neoplasias, e. g. tumors or metastases and, thus, segmentations can overlap. This can be seen in Figure 9, where the tumor (green) and glandula (gray) segmentations overlap. Our approach addresses these cases by assigning overlapping volume to the smaller structure. However, there exist cases where structures are embedded in each other, e. g. tumors or metastases in organs. As long as the neoplastic structures are smaller in volume than the surrounding organ tissue, overlapping volume will be assigned to the neoplasm. Thus, tumors and metastases will not become occult in the final floor map. In this scenario, the overlapping and neoplasm room volumes are identical. However, if the embedded, cancerous tissue is larger than the remaining healthy tissue, the overlapping volume would be wrongfully labeled *healthy* and, thus, the cancerous tissue will be completely omitted. Such problematic cases could be addressed by enabling users to label embedded structures *favored*. As a result, overlapping volume would always be assigned to the favored structure, although it has the larger volume. Moreover, the proposed approach does not distinguish between elongated and spherical structures. Therefore, blood vessels can be heavily distorted if their shape is not preserved (cf. Figure 8).

Result Performance Discussion. The room extrusion, overlap removal, and mesh generation steps require the most processing time. The performance of each step could be improved via parallel processing of structures.

	Interviewees		
	I 1	I 2	I 3
Gender	m	m	m
Age	26	27	43
Experience with Human Anatomy	2.5	3	12
Clinical Experience	4.5	0	0
Active Anatomy Teaching	0	3	12
Q 1			
Q 2			
Q 3			
Q 4			

	Lift Chart			Floor Map		
	I 1	I 2	I 3	I 1	I 2	I 3
Q 5						
Q 6						
Q 7						
Q 8						
Q 9						
Q 10						
Diagnosis						
Therapy Planing						
Physician-Patient Consultation						
Interdisciplinary Communication						
Student Training						
Communication of Findings						

Legend −− − o + ++

Table 3: Evaluation results of the informal interview and questionnaires. The upper part shows how the interviewees rate their anatomical and technical knowledge. Numbers mean *in years*. The lower parts show their assessments about the exploration-supporting capability and clinical versatility of each visualization technique.

In case of the overlap removal step, a well-elaborated and implemented process and resource management would be required, because, using the presented approach, extruded structures are processed pairwise. Regarding the mesh generation, the measured times include the transformation of all structures into meshes. Thus, for example, the average processing time per structure for Data Set 2 was 2.91 seconds. To increase the performance of this step, the cell extent of the Neighboring Cells Algorithm can be increased [2], which can decrease the mesh quality considerably. However, the processing into surface meshes that are used for the floor map visualization has only to be done once.

Evaluation Results Discussion. The interviewees were encouraged to think aloud while exploring the original images with lift charts and floor maps. This revealed two limitations of our approach: First, although Algorithm 1 produces the smallest possible number of floors, the interviewees still reported that the final presentation would include too much visual clutter (Quote: "*There is too much going on.*"). This is depicted in Figure 9 that shows a heavily dissected mandible. From a cognition perspective, this can be explained with the work of Alvarez and Cavanagh [1]: For orientation and navigation tasks, we process and store visuo-spatial stimuli and construct a so-called mental map and they are our mental representation of our spatial environment. However, the required resource to do so, namely the *Visuo-Spatial Working Memory*, is limited and heavily depends on the visual complexity and number of displayed objects. Whereas an anatomical structure is represented by one *bar* in a lift chart, it can be represented by multiple rooms in a floor map. Therefore, although the geometric appearance of rooms can be considered simple, displaying them all appears to be overwhelming. This limitation could be addressed by including knowledge about the in-plane anatomical variability so that adjacent rooms in regions with a low variability get merged.

Secondly, Algorithm 1 divides image stacks into floors when the composition of segmentation masks changes between adjacent slices. This was done to guide the attention of users towards regions with a large anatomical variability. Although this approach is used in other visualization domains [14], the interviewees remarked that they are not used to this type of orientation and that the floor map "*requires a lot of reading*". They use certain anatomical landmarks for orientation, e. g. the vertebrae for the upper body. In regard of the lift charts of Data Set 1 and 3, the interviewees noted that horizontal divisions in the bar that represents the spine and textual descriptions, namely C1-C7 for cervical, T1-T12 for thoracic, and L1-L5 for lumbar vertebrae, would be beneficial to further increase exploration support. This is related to the method of Balabanian et al. [3], who extended lift charts to be applicable for hierarchical relationships between anatomical structures. Therefore, the proposed floor division approach should also be reviewed with respect to anatomical landmarks.

In addition, the evaluation showed the clinical feasibility of the proposed approach (cf. Tab. 3). For diagnosis and therapy planning, e. g. in cases of cancer or to select positions for multiple radiation sources for radiation therapies, distances between anatomical structures are an important decision criterion. Although our approach introduces less geometric abstraction than other methods, by using the maximum axial shape of anatomical structures to create rooms in-plane distances become heavily distorted. For communication tasks, i. e. with patients, students, and colleagues, our approach was rated just as good as lift charts. However, when the aforementioned limitations are addressed, the overall feasibility of the presented approach should improve.

6 CONCLUSION & FUTURE WORK

In this paper, a novel visualization approach was presented that transforms labeled medical image stacks into a three-dimensional map layout, namely floor maps. Furthermore, it was discussed how the resulting visualizations can be combined with the conventional slice-based exploration of CT image stacks. The proposed approach was evaluated by interviewing three anatomy experts, which revealed two shortcomings: First, the main goal was to offer exploration support for medical image stacks via maps. However, although the presented method guarantees to minimize the number of produced floors and rooms, it was reported that anatomical structures are still represented by too many visual entities [1]. This shortcoming could be addressed by generating *good* default views and limiting the number of presented rooms, for example, by *pulling* out corresponding floors like drawers during a slice-based exploration [11, 18]. The exploration-supporting facilities of floor maps should not be hindered, because humans are still able to construct mental maps from piecewise, sequentially presented maps for orientation and navigation tasks [21]. Secondly, a landmark-based approach to construct floors and rooms appears to be beneficial. However, using vertebrae as landmarks only works for the upper body. Therefore, the method has to be adjusted with respect to the anatomical area that was scanned.

Currently, the approach is limited to visualize one data set at a time. A potentially useful extension is the integration of multiple data sets of different modalities. Inspired by Ropinski et al. [16], a combination of CT and *Positron Emission Tomography* (PET) scans could be interesting: While morphological information from CT scans would still be used to create floors and rooms that give a spatial context, physiological information from PET scans could be used to emulate certain functionalities inside rooms, for example light sources. Rooms that represent segmented structures with an increased PET

activity, e. g. tumors or metastases, could be presented in a *lights on* mode, while all other rooms are presented normally in a *lights out* mode. We argue that this feature would strongly guide the attention of users towards interesting (malignant) structures.

Finally, we think about applying varying degrees of geometric abstraction to different types of anatomy, i. e. organs and vessel trees [19]. For organs, instead of extruding their maximum axial shape, their axis or object-aligned bounding boxes could be used in floor map layouts. Methods, such as the Douglas-Peucker algorithm [10], could be used to simplify vessel trees, since they are geometrically similar to rivers and estuaries.

ACKNOWLEDGMENTS

This work is partly funded by the Federal Ministry of Education and Research within the Forschungscampus *STIMULATE* (13GW0095A). We want to thank Wolfgang D'Hanis, Sven Schumann, and Leonardo Nardi for being our interviewees and providing us with feedback.

REFERENCES

[1] G. Alvarez and P. Cavanagh. The Capacity of Visual Short-Term Memory is Set Both by Visual Information Load and by Number of Objects. *Psychol Sci*, 15(2):106–111, 2004.

[2] R. Bade, O. Konrad, and B. Preim. Reducing Artifacts in Surface Meshes Extracted from Binary Volumes. *Journal of WCSG*, 15:67–74, 2007.

[3] J.-P. Balabanian, I. Viola, and E. Gröller. Interactive Illustrative Visualization of Hierarchical Volume Data. In *Proc. of Graphics Interface*, pages 137–144, 2010.

[4] P. Barta, B. Kovács, L. Szecsi, and L. Szirmay-Kalos. Order Independent Transparency with Per-Pixel Linked Lists. In *Proc. of CESCG*, pages 51–57, 2011.

[5] H. Bourquain, A. Schenk, F. Link, B. Preim, G. Prause, and H.-O. Peitgen. HepaVision2 – A Software Assistant for Preoperative Planning in Living-Related Liver Transplantation and Oncologic Liver Surgery. In *Proc. of CARS*, pages 341–346, 2002.

[6] L.-C. Chen, C.-H. Wu, T.-S. Shen, and C.-C. Chou. The Application of Geometric Network Models and Building Information Models in Geospatial Environments for Fire-Fighting Simulations. *Comput Environ Urban*, 45:1–12, 2014.

[7] S. Diepenbrock, J.-S. Praßni, F. Lindemann, H.-W. Bothe, and T. Ropinski. Interactive Visualization Techniques for Neurosurgery Planning. In *Proc. of. Eurographics*, pages 13–16, 2011.

[8] J. Dornheim, D. Lehmann, L. Dornheim, B. Preim, and G. Strauss. Reconstruction of Blood Vessels from Neck CT Datasets using Stable 3D Mass-Spring Models. In *Proc. of Eurographics Workshop on VCBM*, pages 77–82, 2008.

[9] J. Dornheim, H. Seim, B. Preim, I. Hertel, and G. Strauss. Segmentation of Neck Lymph Nodes in CT Datasets with Stable 3D Mass-Spring Models. In *Proc. of MICCAI*, volume 4191 of *LNCS*, pages 904–911, 2006.

[10] D. H. Douglas and T. K. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Cartographica*, 10(2):112–122, 1973.

[11] M. Gai and G. Wang. Indoor3D: A WebGL Based Open Source Framework for 3D Indoor Maps Visualization. In *Proc. of Web3D*, pages 181–187, 2015.

[12] M. Goetz and A. Zipf. Extending OpenStreetMap to Indoor Environments: Bringing Volunteered Geographic Information to the Next Level. In *Proc. of UDMS*, volume 2011, pages 47–58, 2011.

[13] B. Hagedorn, M. Trapp, T. Glander, and J. Döllner. Towards an Indoor Level-of-Detail Model for Route Visualization. In *Proc. of MDM*, pages 692–697, 2009.

[14] P. Mindek, G. Mistelbauer, M. E. Gröller, and S. Bruckner. Data-Sensitive Visual Navigation. *Comput Graph*, 67:77–85, 2017.

[15] F. Ritter, T. Boskamp, A. Homeyer, H. Laue, M. Schwier, F. Link, and H.-O. Peitgen. Medical Image Analysis. *IEEE Pulse*, 2(6):60–70, 2011.

[16] T. Ropinski, S. Hermann, R. Reich, M. Schäfers, and K. Hinrichs. Multimodal Vessel Visualization of Mouse Aorta PET/CT Scans. *IEEE T Vis Comput Gr*, 15(6):1515–1522, 2009.

[17] C. Tietjen, B. Meyer, S. Schlechtweg, B. Preim, I. Hertel, and G. Strauss. Enhancing Slice-based Visuaizations of Medical Volume Data. In *Proc. of Eurographics/ IEEE-VGTC Symposium on Visualization*, pages 123–130, 2006.

[18] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint Selection Using Viewpoint Entropy. In *Proc. of VMV*, pages 273–280, 2001.

[19] I. Viola and T. Isenberg. Pondering the Concept of Abstraction in (Illustrative) Visualization. *IEEE T Vis Comput Gr*, 24(9):2573–2588, 2018.

[20] X. Yin, P. Wonka, and A. Razdan. Generating 3D Building Models from Architectural Drawings: A Survey. *IEEE Comput Graph*, 29(1):20–30, 2009.

[21] H. D. Zimmer. The Construction of Mental Maps Based on a Fragmentary View of Physical Maps. *J Educ Psychol*, 96(3):603–610, 2004.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.27, No.1, 2019

Multi-Resolution Rendering for Computationally Expensive Lighting Effects

Simon Besenthal
Ulm University
simon.besenthal@uni-ulm.de

Sebastian Maisch
Ulm University
sebastian.maisch@uni-ulm.de

Timo Ropinski
Ulm University
timo.ropinski@uni-ulm.de

Abstract

Many lighting methods used in computer graphics such as indirect illumination can have very high computational costs and need to be approximated for real-time applications. These costs can be reduced by means of upsampling techniques which tend to introduce artifacts and affect the visual quality of the rendered image. This paper suggests a versatile approach for accelerating the rendering of screen space methods while maintaining the visual quality. This is achieved by exploiting the low frequency nature of many of these illumination methods and the geometrical continuity of the scene. First the screen space is dynamically divided into separate sub-images, then the illumination is rendered for each sub-image in an adequate resolution and finally the sub-images are put together in order to compose the final image. Therefore we identify edges in the scene and generate masks precisely specifying which part of the image is included in which sub-image. The masks therefore determine which part of the image is rendered in which resolution. A step wise upsampling and merging process then allows optically soft transitions between the different resolution levels. For this paper, the introduced multi-resolution rendering method was implemented and tested on three commonly used lighting methods. These are screen space ambient occlusion, soft shadow mapping and screen space global illumination.

Keywords

Real-Time Rendering, Multi-resolution

1 INTRODUCTION

As a subarea of computer science, real-time computer graphics has developed continuously since the middle of the last century and is of great importance today. With a variety of applications, including medicine or computer-aided design (CAD), real-time computer graphics is nowadays indispensable in many areas of life and is thus a relevant factor in research as well as in business. To render a realistic image many optical and physical phenomena such as camera lenses, light transport, or micro-surface structure must be taken into account. All of these phenomena need to be calculated at pixel level but might rely on information of the surrounding scene to create the effect. Therefore, the number of pixels to be rendered, especially with more complex illumination, is crucial to the necessary computing power and thus to the performance of an application. While the increase in computing power of modern graphics hardware allows for more complicated algorithms, the demand for photo-realistic global illumination effects and high output resolutions in real-time graphics can not be met by current hardware sufficiently.

In order to reduce the computational effort upsampling is often used. This technique renders individual effects, or sometimes the full image, in a lower resolution. Subsequently, the generated images are scaled back up to the full resolution by interpolation. Ultimately, fewer pixels must be calculated and stored, which reduces the computational effort and also the required storage space. Upsampling is particularly common in soft, continuous post-processing effects such as bloom filters or blur, in which quality losses are virtually invisible, depending on the scaling factor. If, on the other hand, you render effects with more concrete structures such as shadows or reflections in a lower resolution and then scale them up, hard edges are displayed washed out and aliasing becomes visible. In addition, there is a risk of under-sampling, which can cause visual artifacts affecting the image quality, especially in animated scenes or during camera movements. Rendering such effects or the entire image by upsampling is therefore usually not always useful, however, two interesting observations can be made: Although such effects may generally have more concrete structures such as hard edges, these high-frequency details are firstly not necessarily evenly distributed in the image space, and secondly, they are often only marginally present in relation to the total area. For example, considering naive shadow mapping with a single light source, depending on the complexity of the scene, a rendered image may contain large areas

that are either completely shaded or fully illuminated. Nevertheless, the necessary operations to determine the brightness of these areas are performed for each individual pixel. For naive shadow mapping, this is certainly not important, but if one considers computationally more complex effects such as ambient occlusion or indirect illumination, the performance could be drastically increased by an intelligent subsampling of certain image areas.

The technique developed in this work exploits the often existing optical continuity of a scene in order to realize computationally intensive lighting effects more efficiently. For this purpose, the image space is first divided into multiple disjoint partial images, so that areas which contain edges or are in their immediate vicinity are separated from areas without edges or with a greater distance to them. Each partial image can be rendered individually with the illumination effects to be realized in suitable resolutions. In principle, a higher resolution is required to correctly create the effect in areas with a higher detail density. However, areas that do not include edges and thus have a lower density of detail can be rendered in lower resolution. The partial images are then reassembled to the original image. In the best case, this image should not differ visually from a full-resolution rendered image. Of particular importance for visual quality and performance is the way in which the individual steps of the technology work. For each different step approaches are presented and explained in this paper.

2 RELATED WORK

In this section we present and explain the techniques and approaches relevant to this work. They follow similar conceptual principles and can be considered as a starting point for the technique developed here. We also highlight the differences to these approaches.

2.1 Upsampling

Upsampling is a technique commonly used in low-frequency visual effects in real-time computer graphics. Examples of effects that are often realized are Bloom or Glare filters [1] and Depth of Field [2]. The blur for the respective effect is not rendered in the full resolution of the application, but in an often much lower resolution. Subsequently, the result is scaled back to the full screen size by means of bilinear interpolation. This can greatly increase the performance at the same optical quality.

2.2 Adaptive Multi-Resolution

There are several approaches that split the computation of illumination effects into multiple resolutions to separate the rendering of low frequency and higher frequency components of these effects. Examples are implementations for indirect light transport [3] and Screen Space Ambient Occlusion [4], which achieve better performance with optically good results. In both approaches, multiple mipmap stages of the G-buffer are used to render the lighting effect to be realized in various resolutions. Subsequently, an upsampling is performed by means of bilateral filters and the different levels are combined. The multi-resolution rendering technique developed in this work makes use of the fundamental principle of separating high and low-frequency components of the illumination, but divides the image into several partial images on the basis of these different proportions. An area of the image is not rendered in all resolutions, but in the best case only in one. This makes it possible to drastically reduce the calculations for higher-frequency components in the image areas in which ultimately no high-frequency components occur exactly.

Nichols and Wyman [5] describe a real-time technique for rendering indirect illumination using multi-resolution splatting. They use min-max mipmaps to find the discontinuities in the geometry. Using these discontinuities, the image space is hierarchically divided into smaller squares, so that areas with higher-frequency components obtain a finer resolution. After the image is completely split into such 'splats' of an appropriate size, the indirect illumination is rendered in all resolutions and the layers are then combined by upsampling to produce the final image. Our technique differs from the algorithm presented by Nichols and Wyman among other things in the method used to decide which resolution to render in. We can apply more flexible filters depending on the situation, while their approach using min-max mipmaps can only find geometric discontinuities. We also use a different approach to combine the final images that prevents visible artifacts. Finally, our technique is not only specialized for indirect illumination using Reflective Shadow Maps, but can also be applied and optimized for various lighting effects due to its high flexibility.

Iain Cantlay [6] describes a technique for rendering lower resolution particles offscreen and combining the result with high resolution renderings of other geometry. In contrast to our approach, this technique can only be applied, if distinct parts of the geometry (in this case particles) are to be rendered in a fixed lower resolution while our technique is more flexible working on pixels.

Guennebaud et al. [7] use variable resolutions for soft shadow mapping in screen space. Again our approach is more flexible and can be applied to a multitude of screen space effects.

2.3 Variable Rate Shading

He et al. [8] propose an extension of the graphics pipeline to natively support adaptive sampling techniques. Nvidia's Maxwell and Pascal architectures have already implemented graphics hardware technologies that could speed up the rendering of an image through the use of different resolutions. Multi-Resolution Shading [9] and Lens Matched Shading [10] can be applied in virtual reality applications to adapt the

resolution of individual image areas to the optical properties of the physical lens that is part of the display. For more general uses Variable Rate Shading [11] (VRS) was introduced as part of the Nvidia Turing architecture. With this technique, the image can be divided into much finer regions, which can be rendered independently in appropriate resolutions. The regions are made up of squares with a edge length of sixteen pixels. Possible applications include 'Content Adaptive Shading' (as for example presented by Vaidyanathan et al. [12]), 'Motion Adaptive Shading' (as for example presented by Vaidyanathan et al. [13]), and 'Foveated Rendering' (as presented by Guenter et al. [14]). In this case, the sampling rate of the image areas is selected adequately depending on the detail density, movement, or focus of the viewer.

The multi-resolution rendering technique developed in this work allows for an even finer and more flexible division of the image, since image areas do not necessarily have to consist of square tiles, but can have any desired shape. This means that a possibly even lower part of the image must be rendered in full resolution, and the performance can be further increased. Apart from that, in contrast to VRS, our technique allows for any number of levels and even lower sampling rates. Our technique is also not dependent on current graphics hardware and can be implemented for widely available systems. In our implementation we focus on the density of details in a scene (Content Adaptive Shading) to decide for the resolution to render in but we can extend our technique by using different edge detection filters or even masks that describe the geometry of lenses in virtual reality.

2.4 Global Illumination Effects

For the exemplary implementation of our technique we use three illumination effects commonly used in modern computer graphics.

Screen Space Ambient Occlusion (SSAO) is a real-time approximation of the occlusion of ambient light by local geometry. The technique was first presented by Mittring [15] and further developed and improved (e.g. by Bavoil et al. [16]).

Shadow Mapping is an algorithm presented by Williams [17] that allows for a fast calculation of shadow rays using a depth buffer. Artifacts introduced by the resolution of the depth buffer can be reduced by percentage closer filtering, introduced by Reeves et al. [18] that also softens the shadows edges. A plausible penumbra can also be realized as described by Fernando [19]. The shadow map is not only sampled at a single position but at multiple neighboring locations.

Screen Space Global Illumination as, for example, described by Ritschel et al. [20] generalizes SSAO to not only dim ambient illumination but also add indirect illumination from other surfaces visible on the screen. The light transport between chosen samples close to a pixel is calculated inducing information from the G-Buffer.

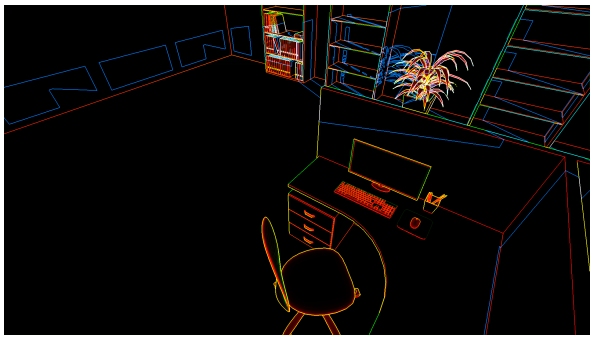


Figure 1: A possible edge image for the multi-resolution rendering technique, the edges are colored for better visualization: the red edges were determined by the differentiation of the normals, the green ones by the depth values and the blue ones by the shadows, normal edges and depth edges are often determined at the same point in the image space (yellow edges).

3 MULTI-RESOLUTION RENDERING

Our presented multi-resolution rendering technique can be subdivided into three basic steps. In the first step, we create a mask in screen space, based on which the image to be rendered is divided into disjoint or complementary sub-images. In the second step, the lighting method to be implemented is rendered for each sub-image in its adequate resolution. Finally the sub-images are combined to create the result image. The conceptual approaches of these steps will be described in more detail below. A visual overview of the algorithms workflow will be given in the supplementary material.

3.1 Mask Creation

The masks are used to divide an image into individual sub-images. While masks can be acquired in multiple ways and even combined using the minimum or maximum (depending on the application) an obvious choice is to use them to separate the higher-frequency image parts from the low-frequency ones. It is often sufficient to use the geometry edges of the scene in screen space to achieve this. These can be found through the information available in the G-Buffer by numerically differentiating depth values and normals for each pixel. For the normal, the first derivative in each of the two dimensions is sufficient, whereas for the depth values, the second derivative gives more reliable results. The discontinuities found reproduce the geometric edges of the scene and can be used to split the image. For screen space ambient occlusion and screen space global illumination, the geometric edges are already sufficient but depending on the illumination effect to be realized, additional information may be required. In case of soft shadow mapping for example, the shadow edges of the scene are needed above all. To this purpose, when creating the mask using the previously created shadow map, a fast shadow calculation (one sample per pixel) can be implemented. We differentiate these values to find dis-

i	SSAO		SSM		SSGI	
	σ_i^2	w_i	σ_i^2	w_i	σ_i^2	w_i
1	0.924	100	0.924	1000	0.924	1000
2	1.848	50	1.848	1000	–	–
3	3.696	20	3.696	1000	0.924	100
4	0	1	0	1	0	1

Table 1: Variances (σ_i^2) and weights (w_i) for each sub-image (i) of all techniques we used. The variances are used to blur the mask, while the weights are used to combine the final image. For SSGI we did not use the second sub-image at all.

continuities in the shading. To avoid artifacts at the geometry edges, we also take them into account for the mask when rendering the soft shadows. Fig. 1 shows an edge image of a scene in which normals, depths, and shadows are differentiated. As an alternative to the edge images we use, min-max mipmaps can also be used to decompose the image as explained by Nichols and Wyman [5].

After we created the final high-resolution mask we downsample it to the resolutions we want our final sub-images to be. We use blur filters with different variances (σ^2) on the downsampled images to determine the areas near the edges. The blurs variance gives the developer control over the size of the area around the edges and determines which areas around the edges are rendered in which resolution. The variances we use can be found in Tab. 1.

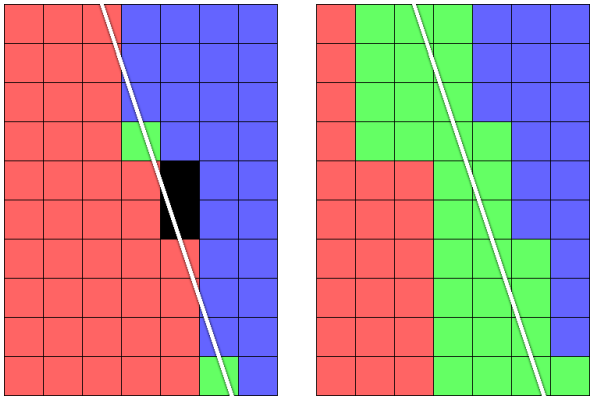


Figure 2: Without accounting for overlap (left), „dead pixels“ (black) occur at the edges of the sub-images (red and blue), which are not contained in any of the sub-images and thus are not rendered. When ensuring an overlap (right), the intersection of the sub-images (green) prevents this circumstance.

A simple way to separate the image into sub-images is to divide them into complementary tiles. An advantage of this method is the disjoint decomposition, whereby no area of the image has to be rendered multiple times. A drawback, however, is that the granularity of the decomposition of the image is limited by the lowest resolution of a sub-image. When naively using the granularity that is determined directly by the resolution of each sub-image, we obtained undefined spaces in the final image between two masked areas. To avoid these we make sure areas of different resolutions have an overlap as shown in Fig. 2. Therefore, we do not separate the image into almost disjoint areas, but always completely include the higher resolution levels in the underlying ones. This means, in particular, that the lowest resolution sub-image always renders the effect to be realized for the entire image. Losses in performance due to the multiple rendering of some image areas are extremely small, because the additional computational effort arises mainly in the lower resolutions. If the blur is optimally selected for the creation of the masks, this approach lets us keep the areas of the higher resolution levels extremely small, resulting in an overall good performance. In addition, this decomposition approach later allows for a very simple re-composition of the final image, because the masks together with fixed weights can serve as an alpha channel for blending the sub-images (see Section 3.3). Fig. 3 shows a possible decomposition of an example scene in screen space.

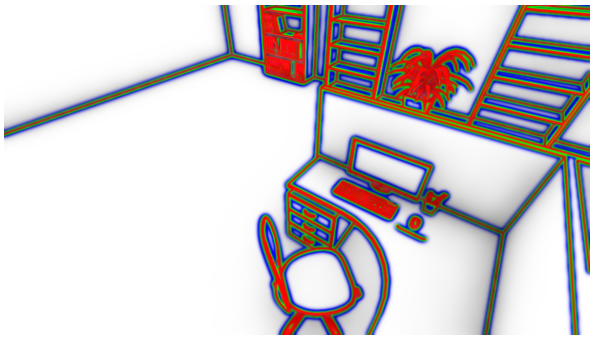


Figure 3: Visualization of the decomposition of an image into four sub-images by means of inclusive areas: The sub-image of the full resolution contains all the red areas, the sub-image of the half resolution all red and green areas, the sub-image of the quarter resolution all red, green and blue areas. The fourth sub-image renders the entire image space at an eighth of the resolution.

3.2 Rendering the Sub-images

Throughout the rendering process we generate all sub-images independently of each other in the chosen resolution. Shape and resolution of the sub-image are defined by the masks determined in step one. Accordingly, an image area of a sub-image is only rendered if and only if the corresponding mask in this image area permits it. Fig. 4 shows an example of rendering four sub-images.

3.3 Blending the Sub-Images

As the final step of the technique we blend the individually rendered sub-images in order to generate the final image. All sub-images are upsampled to the full

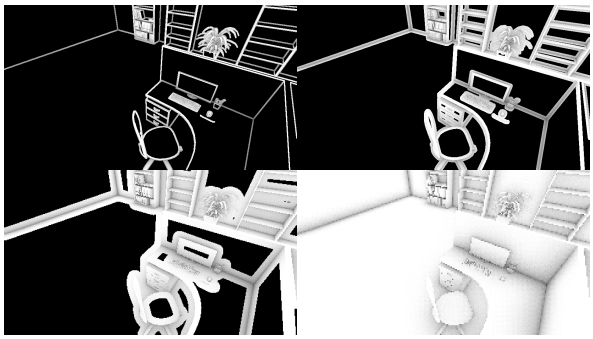


Figure 4: Screen space ambient occlusion rendered in four sub-images, no lighting is calculated for the black areas. The individual sub-images render SSAO in full (top left), half (top right), quarter (bottom left) and eighth resolution (bottom right).

resolution and combined. Using a simple bilinear interpolation would lead to artifacts, as pixels containing visual information can be interpolated with those that contain no information.

A simple solution for this problem would be bilateral interpolation as described by Tomasi and Manduchi [21]. When using this, the sub-images are gradually scaled and merged without scattering missing information of a resolution level into the relevant pixels of the image. To this purpose, a sub-image is always combined with the sub-images already blended in one step. This upsampling technique is also used by Nichols and Wyman [5].

In our case we can use the decomposition masks to calculate the final blending weights. Each sub-image, starting at the lowest resolution, is blended with the next higher resolution sub-image based on the alpha value of each mask. The softness of the transitions between the resolution levels can be determined flexibly using weights. These weights are multiplied with the alpha mask and define the final alpha value for blending.

4 IMPLEMENTATION

In our implementation we applied our multi-resolution rendering technique to three illumination effects commonly found in modern real-time computer graphics. These effects are SSAO, soft shadow mapping (SSM) and screen space global illumination (SSGI). In this section, we describe the implementation of our technique and specific adjustments for the illumination effects used. Our implementation relies solely on the OpenGL 3.3 core profile and can as such run on widely available hardware. According to our experiences during the development stage, a decomposition in four sub-images appears as the best compromise between image quality and speed. The width of the sub-images is successively halved, starting at full resolution width, and are set to full, half, quarter, and eighth. For SSGI we found that not using the halved sub-image did not result in worse image quality. This contributed to a further performance enhancement.

4.1 Rendering of the Sub-Images

To render the sub-images, we use the previously generated masks to create a stencil buffer for each resolution determining the areas. We check if the mask is greater than zero and set the stencil value to one or zero accordingly. We thought about using different thresholds for creating the stencil masks but for our purposes just using zero provided the best results. For each resolution level used, we subsequently render each sub-image using the stencil buffer to eliminate regions that we do not want to render.

For SSAO, depending on the number of samples used, we blur the resulting sub-images in order to reduce the occurring variance of the effect, especially in the lower resolutions. However, we needed to ensure not to transport missing pixel information into the defined areas of the respective sub-image. We achieved this, with a bilateral blur filter.

4.2 Blending of the Sub-Images

Subsequently, the rendered sub-images are blended to compose the final image. We use bilinear interpolation to scale the sub-images to full size and then combine them sequentially, starting at the lowest resolution level. We carry out the final blending between two sub-images by using the values of our masks (a_i) multiplied by a weight (w_i) as a linear interpolation parameter. The weights of our example cases can be found in Tab. 1. We calculate the following for each pixel of the final image. We define c_i as that pixels color value in the i-th sub-image, where c_1 is the full resolution image. The composed image including the i-th sub-image as its highest resolution is called c'_i. The fourth sub-image has the lowest resolution, covers the entire image space and is defined for each pixel. We use its value as the initial value $c'_4 = c_4$. All other c'_i are calculated successively using the alpha values a_i from the corresponding masks and the weights w_i by:

$$c'_i = c_i \cdot \min(a_i w_i, 1) + c'_{i-1} \cdot \big(1 - \min(a_i w_i, 1)\big) \quad (1)$$

The last computed value c'_1 describes the pixel value of the final composite image.

5 EVALUATION

For a basic evaluation we applied our multi-resolution rendering technique to the three illumination effects mentioned (SSAO, SSM and SSGI). We used three test scenes “Office” (20,189 triangles), “Hall” (183,333 triangles), and “Breakfast Room” (a slightly modified version of the one provided by Morgan McGuire [22] with 269,565 triangles) with eight camera configurations for

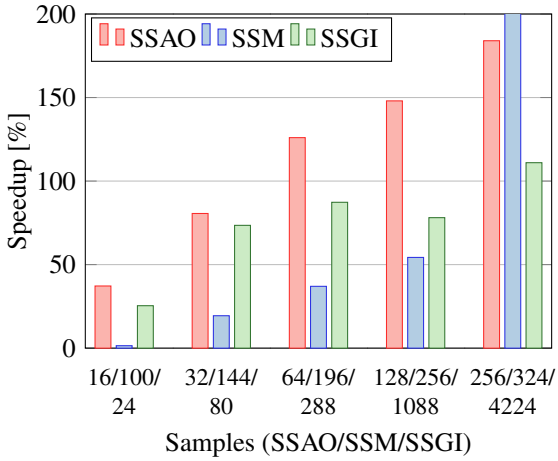


Figure 5: Average speedup in percent by using our multi resolution technique in 4K (3840x2160 Pixels). We show the speedup for our three tested techniques using different numbers of samples for each of them.

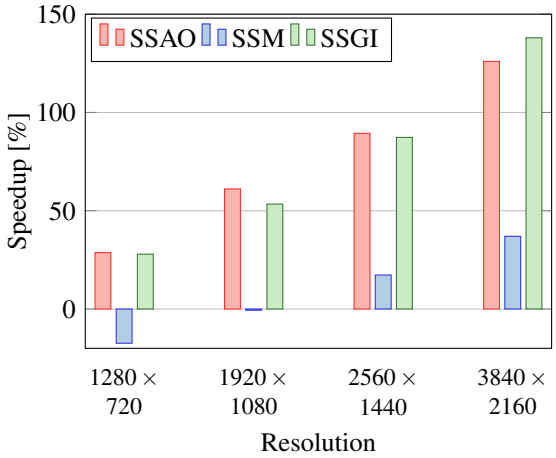


Figure 6: Average speedup in percent of our multi resolution technique at different resolutions. We used fixed numbers of samples for all techniques: 64 samples for SSAO, 196 samples for SSM, and 228 samples for SSGI.

speed and visual comparison. For Soft Shadow Mapping and Screen Space Global Illumination, a modified version of the second scene with 255,432 triangles was used, because it works better with the given directional light sources. For each perspective, the rendering speed was measured using our technique and compared to the speed measured for naive rendering in full resolution. In addition, comparison images of the test scenes are shown and their differences measured and visualized. All tests were performed on a Nvidia Geforce GTX 1080.

5.1 Rendering Speed

For testing the speedup of our technique we used 3840×2160 as a base resolution. We tested each technique with a different number of samples. The average results for 24 different configurations (scene and camera) are listed in Fig. 5. Despite the additional rendering steps needed, our technique outperforms naive rendering in all cases. For a higher number of samples our technique will perform better, since more processing on the GPU can be skipped due to lower resolution rendering.

We also tested our technique for lower resolutions. The Results were not as good as the ones reported for 4K. Nevertheless with the exception of SSM with 196 Samples we achieved clear positive speedups for all illumination techniques even in 720p. Starting from 1440p, all illumination techniques provided positive speedups. Our results for SSM can be explained by the fact that the technique is relatively simple while the mask generation still produces observable overhead. Compared to this overhead, the reduction in GPU computations is relatively low. For lower resolutions the overhead of generating the mask to divide the image and the cost of the additional rendering passes for multiple resolutions dominate over the positive effect of our technique. Fig. 6 shows these results.

5.2 Visual Comparison

While our technique tries to prevent producing images that differ from renderings created with naive full resolution rendering, we could not prevent all visual artifacts. As can be seen in Fig. 7 to 9 these errors occur at the borders of our masks and are mostly due to the Gaussian blur we need to apply to the images to reduce discontinuities at these edges. The blur kernel is very narrow so it is hard to detect the errors when just com-

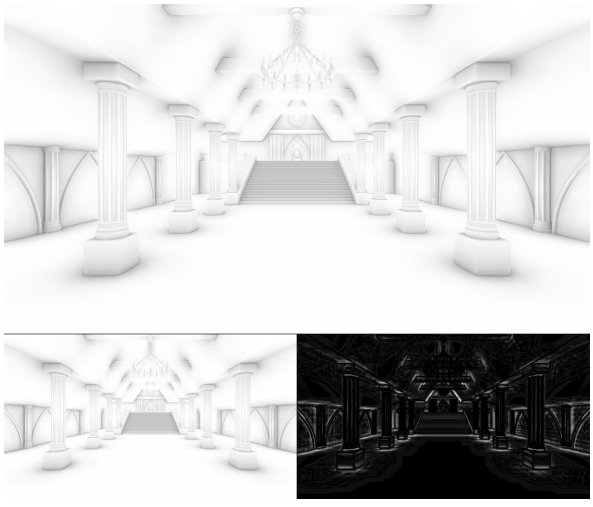


Figure 7: The “Hall” dataset using SSAO and 64 samples. The top image shows our multi resolution technique while in the lower left corner the reference image is shown. In the lower right corner is an enhanced difference image between those two.

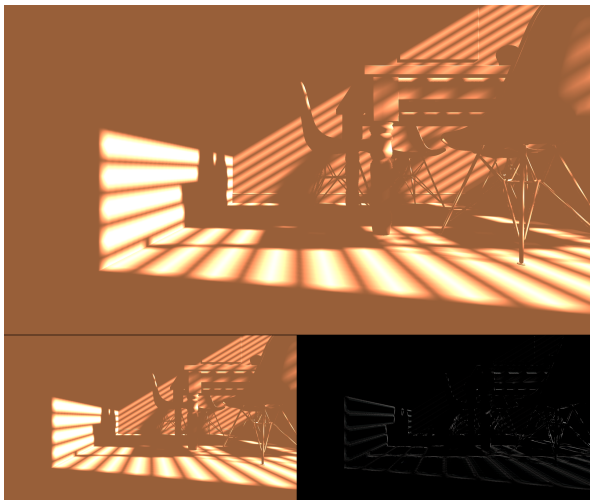


Figure 8: The "Breakfast Room" dataset using SSM and 196 samples. The top image shows our multi resolution technique while in the lower corner the reference image is shown. In the lower right corner is an enhanced difference image between those two.

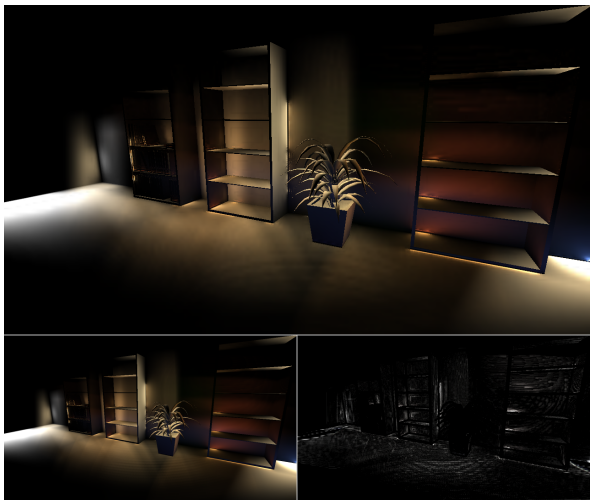


Figure 9: The "Office" scene using SSGI and 4224 samples. The top image shows our multi resolution technique while in the lower corner the reference image is shown. In the lower right corner is an enhanced difference image between those two. The image only shows the SSGI effect without direct illumination to better show the differences caused by our technique.

paring the images directly but is visible in the difference images provided.

Fig. 7 shows the results for SSAO using 64 samples. We chose this number of samples as we think it is a reasonable choice for real applications and a good compromise between speed and image quality. As this image is very bright the differences in the difference image are also more prominent as with the other technique.

Fig. 8 shows the results for SSM using 196 samples. For this lighting effect we can use masks that do not depend directly on the screen space geometry for our technique. The occurring errors are relatively low compared to the other techniques due to the parts of the scene in shadow that are lit with a constant ambient illumination.

Results of the SSGI technique we implemented are shown in Fig. 9. For a visually plausible global illumination effect in screen space we needed a lot of samples so we chose to present the results for 4224 samples. While our results are still convincing some small artifacts can be seen in the corners of the right rack. While these present visible differences to the original image the effects are very minor.

Besides the visual results we provide an overview over all errors in the graphs in Fig. 10. These numbers do not only include the presented images but include images from all three scenes with eight camera configurations each. These numbers support our claim that the errors introduced by our technique are very low.

5.3 Discussion

We presented the performance and visual quality of our method and have two general findings. As a general rule, it was observed that illumination techniques that are more computationally demanding can benefit more from our technique than less demanding ones. This is because of a constant overhead due to mask generation and multiple rendering passes. This overhead becomes dominant for techniques that are less computationally demanding. The second finding is the fact that our technique excels especially in higher resolutions for the same reason.

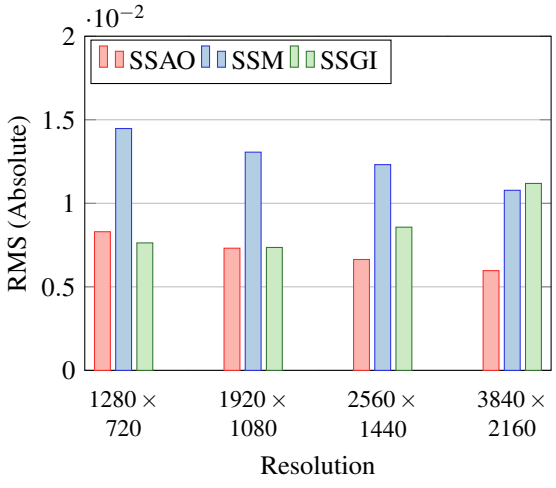


Figure 10: The absolute root mean squared (RMS) errors between result images of our multi resolution technique and images naively rendered with high resolution. We used 64 samples for the SSAO images, 196 samples for SSM and 288 samples for the SSGI images. Values in the compared images ranged from 0 to 1 so the resulting errors can be considered low.

A minor finding is that masks which are more complicated to generate than by simply using the G-Buffer also cause a greater overhead. This makes the use of these masks only feasible for the highest resolutions or techniques that are more computationally demanding than the soft shadow mapping presented here.

6 CONCLUSION & FUTURE WORK

We presented a technique for multi resolution rendering that can be implemented on widely available graphics hardware. Our technique can improve the rendering speed of screen space algorithms drastically (especially for high resolutions) as we have shown for three cases. While the technique presented here is only used for 'Content Adaptive Shading' we can trivially extend it to 'Foveated Rendering' by modulating the mask we use by an importance mask provided by eye trackers. Including 'Motion Adaptive Shading' is also possible by using information of pixel motion in the mask generation process.

To further improve our technique we think that the mask generation process should be modified. For determining the geometry edges, we use normals and depth values from the G-Buffer in screen space. In practice however, non-smooth, modified normals are often used to calculate the illumination. For smooth shading, pixel normals are calculated by the linear interpolation of vertex normals, but in real applications bumpmaps or normal maps are used to modify the normals. In this case, the edge filter could potentially find many more edges, which can result in dramatically increased computational effort and significantly lower efficiency. Possible solutions to these problems would be the exclusive use of unmodified normals or an alternative determination of the edges using the pixel locations in world space. Another problem may arise with certain effects, including, for example, reflections or caustics, since their edges can not be calculated with the information contained in the G-buffer. Also in this case, image areas with higher-frequency components could be rendered in too low a resolution. For such lighting effects, further development of the progressive decomposition of the image would certainly be beneficial. To prevent sub-sampling for some effects, sub-images could also be realized by just using a lower number of samples in full resolution instead of rendering the effect in a lower resolution.

Another interesting application for the multi resolution rendering technique would be using ray tracing for physically correct illumination. In particular, diffuse indirect illumination can only be achieved by relatively high computational effort and can barely be realized in real-time on current graphics hardware. Using the multi-resolution approach, the performance could be increased drastically.

REFERENCES

[1] G. James, *GPU Gems*. Pearson Higher Education, 2004, ch. Real-Time Glow.

[2] T. Scheuermann *et al.*, "Advanced Depth of Field," *GDC 2004*, vol. 8, 2004.

[3] C. Soler, O. Hoel, and F. Rochet, "A Deferred Shading Pipeline for Real-Time Indirect Illumination," in *ACM SIGGRAPH 2010 Talks*. ACM, 2010, p. 18.

[4] T.-D. Hoang and K.-L. Low, "Multi-Resolution Screen-Space Ambient Occlusion," in *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*. ACM, 2010, pp. 101–102.

[5] G. Nichols and C. Wyman, "Interactive Indirect Illumination Using Adaptive Multiresolution Splatting," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 5, pp. 729–741, 2010.

[6] I. Cantlay, *GPU Gems 3*. Addison-Wesley Professional, 2007, ch. High-Speed, Off-Screen Particles.

[7] G. Guennebaud, L. Barthe, and M. Paulin, "High-Quality Adaptive Soft Shadow Mapping," in *Computer graphics forum*, vol. 26, no. 3. Wiley Online Library, 2007, pp. 525–533.

[8] Y. He, Y. Gu, and K. Fatahalian, "Extending the graphics pipeline with adaptive, multi-rate shading," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 142:1–142:12, Jul. 2014.

[9] "VRWorks – Multi-Res Shading," https://developer.nvidia.com/vrworks/graphics/multiresshading, accessed: 2019-02-05.

[10] "VRWorks – Lens Matched Shading," https://developer.nvidia.com/vrworks/graphics/lensmatchedshading, accessed: 2019-02-05.

[11] "VRWorks – Variable Rate Shading (VRS)," https://developer.nvidia.com/vrworks/graphics/variablerateshading, accessed: 2019-02-05.

[12] K. Vaidyanathan, M. Salvi, R. Toth, T. Foley, T. Akenine-Möller, J. Nilsson, J. Munkberg, J. Hasselgren, M. Sugihara, P. Clarberg, T. Janczak, and A. Lefohn, "Coarse Pixel Shading," in *Proceedings of High Performance Graphics*, ser. HPG '14. Goslar Germany, Germany: Eurographics Association, 2014, pp. 9–18.

[13] K. Vaidyanathan, R. Toth, M. Salvi, S. Boulos, and A. Lefohn, "Adaptive Image Space Shading for Motion and Defocus Blur," in *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*, ser. EGGH-HPG'12. Goslar Germany, Germany: Eurographics Association, 2012, pp. 13–21.

[14] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder, "Foveated 3D Graphics," *ACM TOG*, vol. 31, no. 6, pp. 164:1–164:10, Nov. 2012.

[15] M. Mittring, "Finding Next Gen: Cryengine 2," in *ACM SIGGRAPH 2007 courses*. ACM, 2007, pp. 97–121.

[16] L. Bavoil, M. Sainz, and R. Dimitrov, "Image-Space Horizon-Based Ambient Occlusion," in *ACM SIGGRAPH 2008 talks*. ACM, 2008, p. 22.

[17] L. Williams, "Casting Curved Shadows on Curved Surfaces," in *ACM Siggraph Computer Graphics*, vol. 12, no. 3. ACM, 1978, pp. 270–274.

[18] W. T. Reeves, D. H. Salesin, and R. L. Cook, "Rendering Antialiased Shadows With Depth Maps," in *ACM Siggraph Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 283–291.

[19] R. Fernando, "Percentage-Closer Soft Shadows," in *ACM SIGGRAPH 2005 Sketches*. ACM, 2005, p. 35.

[20] T. Ritschel, T. Grosch, and H.-P. Seidel, "Approximating Dynamic Global Illumination in Image Space," in *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. ACM, 2009, pp. 75–82.

[21] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," in *Sixth International Conference on Computer Vision, 1998*. IEEE, 1998, pp. 839–846.

[22] M. McGuire, "Computer Graphics Archive," https://casual-effects.com/data, July 2017.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.27, No.1, 2019

A Lightweight Approach to 3D Measurement of Chronic Wounds

Tim Shirley, Dmitri Presnov, Andreas Kolb
Computer Graphics Group,University of Siegen, Hoelderlinstrasse 3, 57076 Siegen
tim.shirley@student.uni-siegen.de, {dmitri.presnov,andreas.kolb}@ uni-siegen.de

ABSTRACT

This paper presents a light-weight process for 3D reconstruction and measurement of chronic wounds using a commonly available smartphone as an image capturing device. The first stage of our measurement pipeline comprises the creation of a dense 3D point cloud using structure-from-motion (SfM). Furthermore, the wound area is segmented from the surrounding skin using dynamic thresholding in CIELAB color space and a surface is estimated to simulate the missing skin in the wound area. Together with a mesh reconstruction of the wound, the skin surface and the segmented wound is used to calculate the wound dimensions, i.e., its length, surface area and volume. We evaluate the presented pipeline using three wound phantoms, representing different stages in healing, and compare the subsequently scanned and measured wound dimensions with manually measured ones.

Keywords

Scene Reconstruction, Wound Measurement, Structure-from-Motion, Object Segmentation.

1 INTRODUCTION

Chronic wounds are a major and growing health issue worldwide. Besides increasing mortality and treatment expenses, they cause substantial pain and distress due to, e.g., significantly reduced mobility, lower self-esteem and social isolation [1]. As 1-2% of the population [1, 2] are affected by chronic wounds, they are also considered a "silent epidemic". Since chronic wounds mainly affect elderly patients, the demographic shift within Western societies causes their increasing dissemination [3].

The most common type of chronic wounds are venous and arterial ulcers, which primarily affect elderly patients. Diabetic ulcers, one of the characteristics of the diabetic foot syndrome and a frequent long-term result of diabetes mellitus, are also very common. Immobile or paralyzed patients often develop pressure ulcers, which are caused by restricted blood flow due to prolonged pressure between repositioning intervals. The tropical disease *leishmaniasis*, transmitted by the bite of the female sandfly, is another major cause of cutaneous chronic wounds [4].

Evidently, 3D wound measurement approaches are advantageous as they offer much more insight into wound healing, e.g. for deep ulcers, where much of the early healing progress takes place at the bottom of the wound bed [5]. Therefore, an affordable, light-weight 3D wound capturing system has a lot of potential, especially against the background of a growing number of elderly patients and the requirement to provide treatment in underdeveloped and/or rural regions [4].

Wound healing is a highly complex process and the treatment of chronic wounds requires close monitoring over a long time period, sometimes years. Besides careful qualitative observation and documentation by medical staff, reliable quantitative measurements are very important in order to monitor the wound's healing, i.e. its change in size and shape over time. Numerous different measurement techniques exist, ranging from very simple ruler-based size estimates to advanced multi-sensor 3D systems utilizing state of the art computer vision algorithms. However, simple methods are unreliable, imprecise, and uncomfortable for the patient, while advanced systems are often expensive, inefficient to use, and no gold standard has yet been established for wound measurement so far [6].

Current computer vision based approaches either use stereo vision [7], also available in commercial systems like *MAVIS II*[1] or Time-of-Flight (ToF) measurements [8]. However, stereo imaging as well as ToF range measurement requires specific camera devices and neither stereo nor ToF can be considered ubiquitous sensors, so far. Furthermore, compared to current RGB cameras in mobile phones, ToF suffers from low

[1] imaging.research.southwales.ac.uk/projects/wm/mavis

image resolution and from depth measurements errors due to subsurface scattering [9].

In this paper, we present a light-weight wound measurement system based on RGB images captured using standard smartphone cameras, thus solving the wound measurement problem within the emerging domain of mobile health (mHealth) [10]. The integrated sensors in smartphones deliver images of decent quality, allowing for high quality photography at very low cost.

Our 3D wound reconstruction pipeline comprises *Structure-from-Motion (SfM)* as monocular 3D-reconstruction technique. The further processing of the resulting sparse point clouds involves a segmentation into skin and wound regions, the extraction of the wound's contour, and a mesh reconstruction of the wound. Based on the resulting 3D reconstructions of wound and a surface fitting of the skin in order to estimate the healthy state, our approach performs automatic length, area and volume measurements. For evaluation, we produced a sequence of wound phantoms for which we acquired several series of images.

2 PRIOR WORK

Mukherjee et al. [11] give an overview on contact-free, sensor-based techniques for wound measurement, discussing optical approaches as well as approaches that involve more exhaustive sensors such as hyperspectral, thermal and laser doppler imaging or confocal microscopy. While *2D wound measurement* approaches have been developed for more than two decades and are still being researched [12, 13], this paper focuses on *optical approaches for 3D wound measurement*.

The idea of using standard 2D RGB imagery for 3D wound reconstruction and measurement dates back more than one decade. These early approaches reconstruct sparse 3D point clouds from stereo or multi-view imagery taken from the wound. Albouy et al. [14] use a Harris corner detector, cross-correlation, outlier removal and homography estimation in order to compute a sparse 3D point cloud that is used for estimating the wound's volume. Treuillet et al. [5] use stereophotogrammetry in order to reconstruct a sparse 3D point cloud. More recently, general approaches for 3D geometry reconstruction using 2D images acquired with smartphones have been proposed [15, 16]. While Kolev et al. [15] create a point-based 3D model by integrating multiple stereo-based depth hypotheses into a compact and consistent 3D model, Muratov et al. [16] use an SfM like approach that involves additional IMU data. Sirazitdinova and Deserno [7, 17] propose to use similar approaches in 3D wound assessments. They opt for motion stereo [18] in order to reconstruct 3D geometry from 2D RGB imagery acquired with a smartphone. So far, there is no publicly available documentation regarding the system's implementation or evaluation.

Gaur et al. [8] describe an alternative 3D wound measurements system based on RGB-D imagery captured with an Intel ToF camera. After registration of the RGB and the depth images, the wound is segmented directly in image space using standard filter and morphological operators. Then, they identify the wound's boundary pixels and fit a plane to the boundary's range data. Both, the wound and the estimated healthy skin are modeled as quadratic surfaces, in order to measure the wound's volume.

Summary. Current 3D wound measurement approaches mainly rely on special-purpose hardware for wound acquisition, such as stereo vision or time-of-flight cameras. While early approaches, involving 2D RGB images, deliver only sparse 3D geometric information and are therefore rather inaccurate, there are no reports on successfully applying recent improvements in dense 3D scene reconstruction from RGB images on mobiles to 3D wound measurement. In this paper we present an SfM pipeline that can be successfully applied to 3D wound reconstruction.

3 METHOD

In this section, we present our light-weight wound measurement system. Our approach uses a sequence of RGB images of a chronic wound that has been captured using standard smartphone cameras. Fig. 1 depicts the main components of our wound measurement system that can be summarized as follows:

Acquisition: A sequence of images or a video is taken from the considered chronic wound. Additionally, a marker is located close to the wound in order to solve for the scale ambiguity inherent to SfM surface reconstruction.

Structure from Motion (SfM): Based on features extracted from the input images, image pairs with large overlap are identified, camera poses are estimated and a 3D point cloud is computed (see Sec. 3.1).

Point Cloud Preprocessing: Depending on the input imagery, the resulting raw point cloud contains a significant number of outliers that are removed in this stage. Furthermore, the scale ambiguity is resolved using the marker that has been placed in the scene (see Sec. 3.2).

Wound Segmentation & Fitting: The clean point cloud is segmented in wound and skin using a color thresholding and clustering. Furthermore, the contour of the wound is extracted and a surface is fitted to the skin region, which represents the condition of healthy skin (see Sec. 3.3).

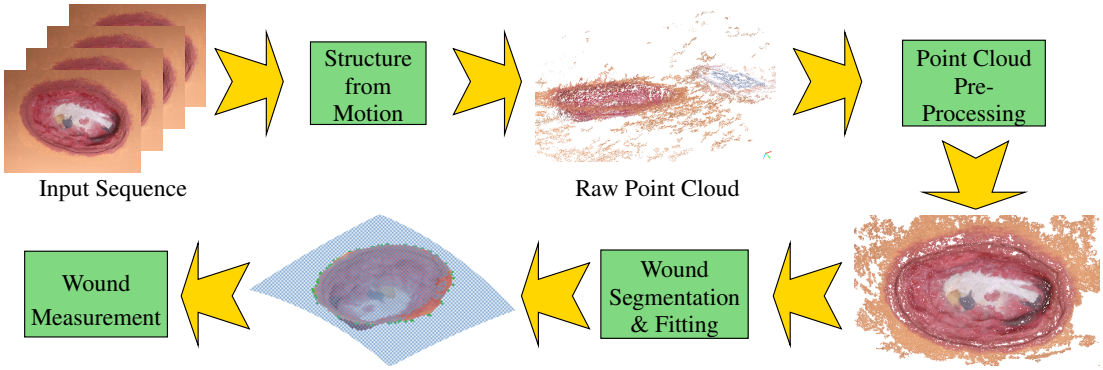


Figure 1: Overview of the proposed 3D wound reconstruction method.

Wound Measurement: In this stage the segmented wound points are converted into a mesh. Based on the meshed wound region, the wound's size, area and volume are deduced (see Sec. 3.4).

3.1 Structure from Motion (SfM)

Structure-from-Motion (SfM) is a well established technique from scene reconstruction that uses monocular RGB image sequence as input. This section briefly describes the main concept behind SfM based on Schoenberger and Frahm [19]. The SfM process consists of two main stages, correspondence finding and incremental reconstruction.

Correspondence Finding. Given a set $\{I_i \mid i = 1 \dots N_I\}$ of unordered images, feature sets $\mathscr{F}_i = \{(\mathbf{x}_j, \mathbf{f}_j) \mid j = 1 \dots N_{F_i}\}$ representing features $\mathbf{f}_j$ at image locations $\mathbf{x}_j$ are extracted for each image I_i. Frequently, the scale-invariant feature transform (SIFT) is used [20]. Based on the feature sets $\mathscr{F}_i$, image pairs with sufficient spatial overlap, i.e. with sufficient common features are identified using, e.g., the Lukas-Kanade tracker [21]. Afterwards, mismatching feature correspondences are removed using RANSAC, and the pairwise image transformations are established.

Incremental Reconstruction. The scene model is described by a set of points $\mathscr{X}$ and a set of camera poses $\mathscr{P}$. SfM is initialized with a carefully selected image pair and its reconstructed 3D points using triangulation. Further images are registered using the feature correspondences. As the incremental nature of this approach and the inherent imprecision of point estimates causes accumulation of errors and point drift, the reconstruction parameters, i.e., the camera poses $\mathscr{P}$ and the 3D point locations $\mathscr{X}$ are regularly refined using bundle adjustment [22]. This process minimizes the reprojection error applied to each 3D point $\mathbf{X}_j^{\text{corr}} \in \mathscr{X}$ corresponding to the feature location $\boldsymbol{x}_j$ in image I_i, i.e.

$$E = \sum_{i=1}^{N_I} \sum_{j=1}^{N_{F_i}} \rho \left(\| \pi(\mathbf{P}_i, \mathbf{X}_j^{\text{corr}}) - \mathbf{x}_j \|_2^2 \right), \qquad (1)$$

where π is the projection function defined by the camera pose $\boldsymbol{P}_i \in \mathscr{P}$ for image I_i and ρ is a loss function. This results in a non-linear optimization process.

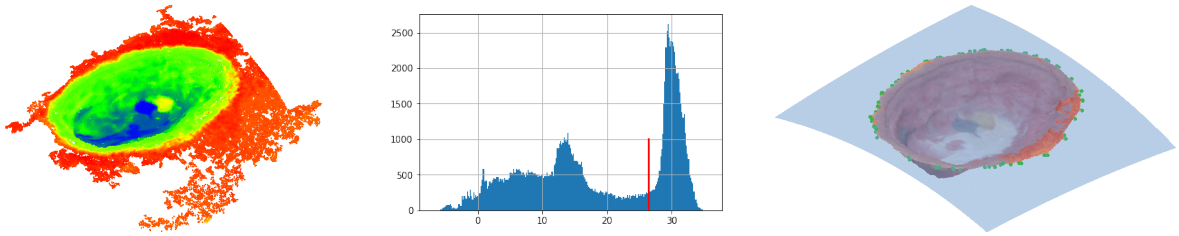
3.2 Point Cloud Preprocessing

As the SfM method is intrinsically scale ambiguous, wound measurement requires proper rescaling of the scene, i.e. of the point cloud reconstructed by SfM. Therefore, we place a 30×30 mm *ArUco* marker [23] that is printed on a 45×45 *mm* board close to the wound. Due to the high contrast of the marker, the marker's points can be segmented easily using fixed color thresholds. Afterwards, a plane is fitted to the marker points using RANSAC and the scene is transformed into the marker's plane. Then, the marker dimensions are determined and the point cloud is rescaled using the resulting isotropic scale factor. Finally, the marker region is removed from the point cloud using a simple spatial cropping.

Although SfM produces dense and comparably precise point clouds, they frequently contain patches of outlier points that are caused by faulty correspondences due to, for example, specular reflections in the input images. Most outliers form small clusters or spurious sheets disjunct from the main point cloud representing the captured object. Therefore, we apply a clustering based on the density-based algorithm proposed by Ester et al. [24].

3.3 Wound Segmentation, Surface Fitting & Meshing

In order to perform wound measurement, the wound must be segmented from the surrounding skin. As the wound's boundary does not necessarily have distinct geometric features, we opt for segmentation using color features. Therefore, we transform the point colors into the CIELAB color space since it emphasizes the color difference between healthy skin and wound tissue [25] (see Fig. 2a). The color disparity in the b^*-channel allows for a clear distinction between wound and skin using a simple dynamic threshold applied to the histogram



(a) Color mapped CIELAB b^* channel (b) Histogram with threshold (c) Surface fitting & contour

Figure 2: Segmentation and surface fitting: The color mapped visualization of the CIELAB b^* channel (2a), the histogram with threshold for segmentation (2b), and the finally segmented wound points with the contour points (green) and the fitted skin surface (2c).

of all colors in the point cloud using the triangle algorithm [26] (see Fig. 2b).

Color segmentation may result in spurious points that have been wrongly classified. This mainly happens in the wound region in which points occasionally are classified as skin. In order to remove residual clusters or falsely classified points in either region, we extract the largest connected skin and wound region using the clustering approach from Ester et al. [24].

After segmentation, we identify the points on the wound's contour. Therefore, we compute the 4-neighborhood for each point and select the points as contour, if two of their neighbors are classified as skin and two as wound. The resulting set of contour points is coarsened by subsampling. To this end, we use a coarse 16^3 voxel grid and select in each cell the contour point closest to the voxel center. Furthermore, we apply a statistical contour smoothing approach by analyzing the discrete curvature of each contour point $\boldsymbol{c}_i$ defined as $\text{curv}(\boldsymbol{c}_i) = \frac{\|\boldsymbol{c}_{i+1} - 2\cdot\boldsymbol{c}_i + \boldsymbol{c}_{i-1}\|}{\boldsymbol{c}_{i+1} - \boldsymbol{c}_{i-1}}$. Calculating the mean μ and standard deviation σ of the curvature values, we remove all contour points above one sigma, i.e. with $\text{curv}(\boldsymbol{c}_i) > \mu + \sigma$.

Next, we apply least square fitting to extract two surfaces, i.e. a planar surface to the contour points and a quadratic surface to the points classified as skin. The latter resembles the healthy state of the wound region. Both surfaces are utilized in the subsequent wound measurement (see Sec. 3.4 and Fig. 2c).

Lastly, the wound's point cloud is transferred into a triangular mesh using Poisson surface reconstruction [27]. This is necessary in order to perform a volume measurement of the wound (see Sec. 3.4).

3.4 Wound Measurement

Based on the reconstructed mesh, the segmentation of the points into subsets $\mathscr{S}$ for skin, $\mathscr{C}$ for contour and $\mathscr{W}$ for wound, and the fitted planar surface (w.r.t. the contour points) and quadratic surface (w.r.t. the skin points), we extract the wound's *length, area* and *volume*.

The length of the wound is simply the maximum distance between contour points, i.e. $\max\{\|\mathbf{x}-\mathbf{y}\| \mid \mathbf{x},\mathbf{y} \in C\}$. The area is computed by projecting the contour points $\mathscr{C}$ onto the planar surface resulting in $\mathscr{C}'$. Calculating the center of gravity $\mathbf{g}$, the area is computed by summing up the area of all triangles $\triangle(\mathbf{c}_1, \mathbf{c}_2, \mathbf{g})$ formed by any two adjacent contour points $\mathbf{c}_1, \mathbf{c}_2$ and the center of gravity [28].

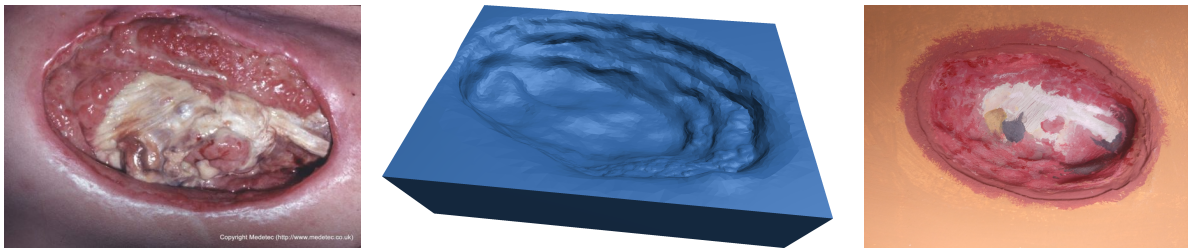
In principle, the wound's volume is enclosed by the reconstructed surface mesh including the wound and contour points, and the triangulated quadratic surface that estimate the healthy skin. However, both geometries do not perfectly intersect at the wound's contour line. Creating a single, closed mesh is a non-trivial task, as direct mesh intersection and hole filling may not lead to a proper solution, e.g. to spurious volume fractions, in case of non-planar contours. Instead, we use the reference plane generated from the contour points and compute the wound's volume based on the height fields $s(\boldsymbol{x}), w(\boldsymbol{x})$ of the skin and the wound surfaces parameterized above the reference plane, respectively. Using the triangulated wound surface, the volume is calculated as

$$\sum_{\triangle \in \mathscr{W}} \cos(\phi) \cdot \text{area}(\triangle) \cdot (s(\boldsymbol{c}_\triangle) - w(\boldsymbol{c}_\triangle)),$$

where $\boldsymbol{c}_\triangle$ is the center of the triangle in the parameter plane and ϕ is the angle between the triangle normal and the normal of the reference plane. We make sure to consider only wound regions that lie within the wound's contour in the reference plane.

4 IMPLEMENTATION

The pipeline presented in Sec. 3 has been implemented as prototype on a PC/laptop environment in order to verify its proper operation. We use the *COLMAP* approach that incorporates the SfM optimizations presented by Schoenberger and Frahm [19] and their multi-view-stereo approach [29]. To enable a fast and flexible development of the further pipeline stages, the programming language *python* together with the *scipy* [30]



(a) Large pressure ulcer (b) 3D ulcer model (c) The final phantom

Figure 3: Creating the wound phantom: The reference photograph of a large pressure ulcer (a), the 3D model sculpted in *blender* (b), and the final painted clay model created using a 3D printed mold (c).

software package was selected. The *pyntcloud* [2] library as well as the *pandas* data structure [31] was used for most point cloud processing operations. For effective and efficient code development, the entire pipeline, with the exception of the Poisson surface reconstruction, has been integrated into a *jupyter notebook*[3]. For color CIELAB-based segmentation we used the *skimage* library [32] that provides the function for color transformation and dynamic thresholding.

5 RESULTS

5.1 Phantom Creation & Manual Wound Measures

Unfortunately, there are no publicly available ground truth data sets for chronic wounds. Therefore, we fabricated wound phantoms based on an available photograph of a pressure ulcer[4] that was sculpted using the 3D modeling software *blender*[5]. A negative of the digital model was 3D-printed and used as a mold to create physical models with modeling clay. The clay was subsequently painted to roughly resemble the wound photograph and glued to a bent cardboard. Based on the first model resembling the photograph, two further wound models were created, simulating healing and shrinking of the wound area and volume. Thus, we finally have three wound phantoms **Large**, **Medium**, and **Small**. For scale disambiguation, a marker was placed in the scene (see Sec. 3.2).

We manually determine reference values for the wound measures. The length measurements were taken with a household tape measure at one millimeter precision. The area was determined by placing a sheet of millimeter paper on the wound, tracing the contour with a pen and manually reading the area. Both approaches for manually estimating the length and the area of the wound are frequently used in clinical practice. The wound volume was determined by filling the phantom with water using a 0.1ml precision syringe. All three manual reference measurements have been conducted by a student with background in medical informatics w/o any practical experience in wound measurement.

5.2 Image Acquisition

In order to assess the quality of the reconstruction pipeline, we acquired three image sequences using a OnePlus 3 (A3003) smartphone with a Sony IMX298 16 megapixel CMOS sensor paired with a F/2.0 aperture lens and phase detection focus. For each wound phantom, we acquired two sets of images (@4640×3480 resolution) and one video sequence (@1920×1080 resolution). The **Img_Fast** image sequence was captured within 30s, while the **Img_Acc** image sequence focuses on the precise image acquisition. The video sequence **Video** is approximately 30s long, captured at some 5 frames per second. Tab. 1 states the precise number of images in column **#im**).

5.3 Evaluation

3D Reconstruction Quality

Since the precise geometry of the clay wound phantom is unknown and no alternative (or gold standard) 3D reconstruction method was available, we directly evaluated the quality of the 3D reconstruction by comparing the clean point cloud produced by our SfM method to the original 3D wound model created with *blender*. Thus, the resulting error also incorporates imprecision introduced by the 3D printing and the molding process. The differences between the reconstructed point clouds and the original 3D models were calculated with *CloudCompare*[6]. Tab. 1 depicts the resulting geometry error in column **geom.err**. We find a very good agreement of the reconstruction with respect to the digital wound model of a standard deviation below 1 mm. Note, that the mean error is close to zero, as the reconstructed point clouds and the original 3D models are co-registered for comparison.

[2] github.com/daavoo/pyntcloud
[3] jupyter.org
[4] www.medetec.co.uk/slide%20scans/pressure-ulcer-images-a/target92.html
[5] www.blender.org
[6] www.cloudcompare.org

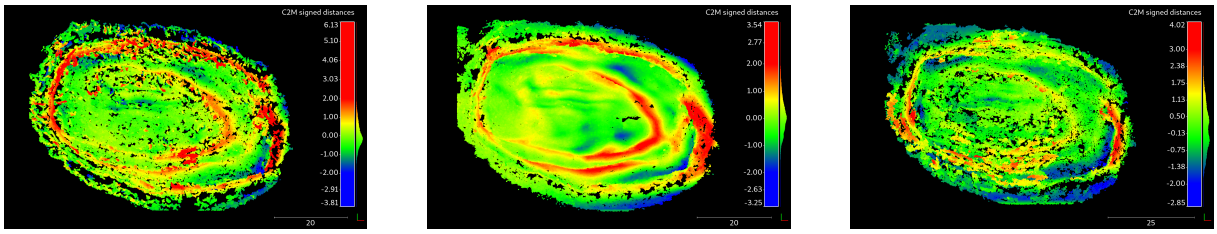


Figure 4: 3D Reconstruction Quality. Color-coded geometric error of the reconstructed wound **Medium** for the image sequence **Img_Fast** (left), **Img_Acc** (middle), and **Video** (right).

Wound/Sequ.	**#im**	**#recon.points**		**scale**	**geom.err.** [mm]		**segm.qual.** [%]			**meas.comp.** [%]		
		raw	clean	**acc**[%]	mean	std	TPR	TNR	ACC	*l*	*A*	*V*
Large/Img_Fast	13	232,213	208,149	86.5	0.06	0.58	99.6	82.4	94.9	6.3	-46.1	-45.6
Large/Img_Acc	50	768,990	581,003	98.0	0.04	0.61	99.8	89.8	96.7	-2.1	-10.9	-10.1
Large/Video	154	158,119	108,892	99.9	0.05	0.71	94.2	88.7	92.0	-1.4	-11.9	-14.3
Medium/Img_Fast	30	536,395	337,581	98.0	0.10	0.91	96.8	86.1	92.5	-1.1	-10.3	-9.8
Medium/Img_Acc	50	924,083	810,568	100.1	0.09	0.76	99.6	92.1	96.0	4.5	-12.9	-20.6
Medium/Video	150	250,629	207,327	93.4	0.10	0.93	95.5	89.8	92.5	3.8	-8.6	-16.4
Small/Img_Fast	50	401,751	322,663	93.1	0.04	0.44	99.0	91.3	95.6	11.4	- 3.6	2.2
Small/Img_Acc	50	986,796	486,801	99.3	0.15	0.80	99.4	90.5	95.4	4.4	-17.7	-20.2
Small/Video	143	286,819	224,004	97.3	0.12	0.12	93.5	93.6	93.6	3.3	-16.0	-18.0

Table 1: Qualitative results for all three wound phantoms and all three image sequences captured for each phantom. The columns contain the number of images in the sequences (**#im**), the number of reconstructed points before and after outlier removal and cropping (**#recon.points**), the accuracy of the geometric scale factor (**scale acc**), the geometric error w.r.t. the digital wound model (**geom.err.**), the segmentation qualitative (**segm.qual.**), and the wound measurement comparion w.r.t. the manually deduced wound measures (**meas.comp.**).

Furthermore, Fig. 4 shows the color-coded geometric reconstruction error for geometric error of the reconstructed wound **Medium** for all three image sequences. As to be expected, our approach reconstructs more points in case of the accurate image sequence. However, the video acquisition yields comparable results regarding the geometric error.

Ignoring the **Large/Img_Fast** experiment, which will be discussed below, the scale factor has been determined at least with 93% accuracy (see Tab. 1, col. **#scale acc**). Furthermore, $50-90\%$ of the reconstructed points have been finally segmented as part of the skin or the wound, i.e. they have passed outlier removal and the cropping of the marker (see Tab. 1, col. **#recon.points**).

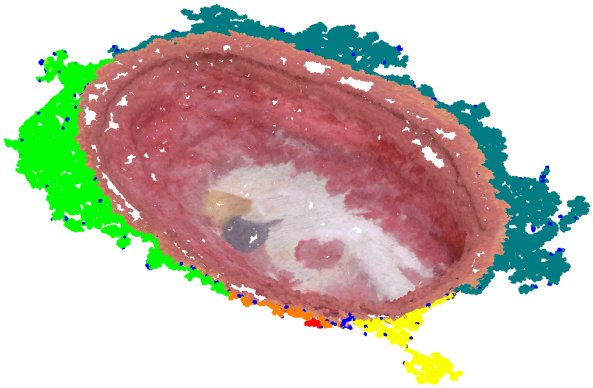


Figure 5: Segmentation result for the skin region of the experiment **Large/Img_Fast**.

Segmentation Quality

Consulting the segmentation quality (column **segm.qual.** in Tab. 1), we find that for all wound phantoms and all image sequences our approach achieves good to very good results. The true positive rate (TPR) is $93.5-99.8\%$, the true negative rate (TNR) is in the range of $82.4-93.6\%$, and the accuracy is between 92% and 96.7%.

Wound Measurement Quality

Tab. 1, column **meas.comp.**, depicts the wound measurement comparison to the manually deduced wound measures. First, we will discuss the results except for **Large/Img_Fast**, for which we get significant larger errors than for the other experiments. Compared to the manual measurement, the wound's length has been estimated fairly accurate, only the **Small/Img_Fast** experiment results in an error of 11%. In general, the length is slightly over-estimated. Regarding the wound area, our approach delivers clearly under-estimated values. At first glance, this under-estimation seems to contradict the over-estimation of the wound's length. The manual wound measurement approaches described in Sec. 5.1 are, however, fully independent, i.e. the manual measurements for length and area do not necessarily correlate.

Regarding volume measurement, we first have to consider the fact that our volume measurement approach described in Sec. 3.4 uses a bent quadratic surface as estimate for the healthy skin, while the manual reference approach to volume measurement based on the wound phantoms rather refer to a planar wound cap (see Sec. 5.1). Therefore, we additionally estimate the wound's volume with respect to the planar surface fitted to the wound's contour (see Sec. 3.4) for comparison with the manually estimated volume. Still, the comparison is not very expressive, since the manual approach using water filling results in over-estimated volumes as the water's surface can easily exceed the wound contour level due to the water's surface tension. From this perspective, it is not extremely surprising, that our approach "underestimates" the wound's volume if compared to the manual reference.

Comparing the wound measurement for the individual phantoms in Tab. 1, column **meas.comp.**, we realize that the **Img_Acc** and the **Video** sequences deliver quite comparable results.

Having a closer look at the **Large/Img_Fast** experiment, we find that the number of images acquired (13) is significantly lower compared to the other image sequences. While the scale accuracy, the geometry error and the segmentation quality indicate a successful segmentation, our approach could not retrieve the wound's contour properly. This is due to the fact, that the clustering result for the skin region failed to identify a single region, as the number of skin points is very low. Subsequently, the final skin region does not enclose the wound completely and, consequently, the contour has not been extracted properly (see Sec. 3.3 and Fig. 5). Still, if a sufficient number of images is acquired, the **Img_Fast** method also produces reasonable results, as has been demonstrated by the **Img_Fast** wound acquisitions for the **Medium** and the **Small** phantom. Note that the calculation of the segmentation quality in Tab. 1, column **meas.comp.**, for the **Large/Img_Fast** experiment accounts for all skin points, not only the main cluster. This, however, does not have any significant impact on the segmentation quality, as the evaluation assumes a binary classifier w/o a miscellaneous class.

Public Science

We will make all digital data, i.e., the digital models of the wound phantoms, the photographs, and the wound measurement prototype publicly available via GitLab upon publication.

5.4 Conclusion & Limitations

In this paper we present a novel method for 3D measurement of chronic wounds that solely relies on standard RGB imagery. The approach incorporates structure-from-motion as 3D reconstruction, CIELAB-based color segmentation of the wound and skin region, wound contour reconstruction, and surface fitting to emulate the healthy skin state. Compared to prior techniques presented in literature, our approach has minimal requirements regarding image acquisition, i.e. standard cameras in mobile devices such as smartphones are sufficient. The quantitative and qualitative evaluation of our approach using realistic wound phantoms and different image acquisition modes yields very robust results, if the number of images acquired is large enough. Ignoring cases of insufficient image counts, all three image acquisition types, i.e. **Img_Fast, Img_Acc**, and **Video**, yield good segmentation results and wound measures. A higher number of input images and larger image resolution for the **Img_Acc** method, however, requires more time to capture and increases computational cost.

The main limitation of our approach is that although we have used statistical approaches to determine thresholds and other parameters wherever possible, there are still some parameters that need to be adjusted manually. This mainly refers to clustering approach [24] and wound segmentation. Furthermore, we so far have not been able to apply our approach to real wounds.

6 REFERENCES

[1] M. Augustin and K. Maier. Psychosomatic aspects of chronic wounds. *Dermatology and Psychosomatics/Dermatologie und Psychosomatik*, 4(1):5–13, 2003.

[2] A. R. Siddiqui and J. M. Bernstein. Chronic wound infection: Facts and controversies. *Clinics in Dermatology*, 28(5):519–526, 2010.

[3] C. K. Sen, G M. Gordillo, S. Roy, R. Kirsner, L. Lambert, T. K. Hunt, F. Gottrup, G. C. Gurtner, and T. Longaker. Human skin wounds: A major and snowballing threat to public health and the economy. *Wound Repair and Regeneration*, 17(6):763–771, 2009.

[4] L. Casas, B. Castaneda, and S. Treuillet. Imaging technologies applied to chronic wounds. In *Proc. Int. Symp. Applied Sciences in Biomedical and Communication Technologies (ISABEL)*, pages 1–5. ACM, 2011.

[5] S. Treuillet, B. Albouy, and Y. Lucas. Three-dimensional assessment of skin wounds using a standard digital camera. *IEEE Transactions on Medical Imaging*, 28(5):752–762, 2009.

[6] R. Khoo and S. Jansen. The evolving field of wound measurement techniques: A literature review. *Wounds: a compendium of clinical research and practice*, 28(6):175, 2016.

[7] E. Sirazitdinova and T. M. Deserno. Introducing low-cost stereo imaging for cutaneous wound as-

sessment. In *Proc. IEEE Symp. Computer-Based Medical Systems*, pages 138–139. IEEE, 2016.

[8] A. Gaur, R. Sunkara, A. Raj, and T. Celik. Efficient wound measurements using RGB and depth images. *Int. Journal of Biomedical Engineering and Technology*, 18(4):333–358, 2015.

[9] D. Bulczak and A. Kolb. Efficient subsurface scattering simulation for time-of-flight sensors. In *Proc. Vision, Modeling & Visualization (VMV)*, 2018. DOI: 10.2312/vmv.20181259.

[10] M. Kay, J. Santos, and M. Takane. mHealth: New horizons for health through mobile technologies. *Global Observatory for eHealth Series*, 64(7):66–71, 2011.

[11] R. Mukherjee, S. Tewary, and A. Routray. Diagnostic and prognostic utility of non-invasive multimodal imaging in chronic wound monitoring: a systematic review. *Journal of Medical Systems*, 41(3):46, 2017.

[12] M. Herbin, F.-X. Bon, A. Venot, F. Jeanlouis, M.L. Dubertret, L. Dubertret, and G. Strauch. Assessment of healing kinetics through true color image processing. *IEEE Transactions on Medical Imaging*, 12(1):39–43, 1993.

[13] S. H. Sprigle, M. Nemeth, and A. Gajjala. Iterative design and testing of a hand-held, non-contact wound measurement device. *Journal of tissue viability*, 21(1):17–26, 2012.

[14] B. Albouy, S. Treuillet, Y. Lucas, and J C Pichaud. Volume estimation from uncalibrated views applied to wound measurement. In *Proc. Int. Conf. Image Analysis and Processing*, pages 945–952, 2005.

[15] K. Kolev, P. Tanskanen, P. Speciale, and M. Pollefeys. Turning mobile phones into 3D scanners. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 3946–3953, 2014.

[16] O. Muratov, Y. Slynko, V. Chernov, M Lyubimtseva, A Shamsuarov, and V. Bucha. 3DCapture: 3d reconstruction for a smartphone. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR) – Workshops*, pages 893–900, 2016.

[17] E. Sirazitdinova and T.M. Deserno. System design for 3D wound imaging using low-cost mobile devices. In *SPIE Medical Imaging*, page 1013810, 2017.

[18] P. Ondrúška, P. Kohli, and S. Izadi. Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE Trans. Visualization and Computer Graphics*, 21(11):1251–1258, 2015.

[19] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

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

[21] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Int. Joint Conf. Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[22] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment-a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[23] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 2014. DOI: 10.1016/j.patcog.2014.01.005.

[24] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. Conf. Knowledge Discovery and Data Mining (KDD)*, volume 96, pages 226–231, 1996.

[25] H. Lee, B. Lee, J. Park, and W. Sun. Segmentation of wounds using gradient vector flow. In *Proc. Int. Conf. Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, pages 390–391, 2015.

[26] G. W. Zack and E. Rogers. Automatic measurement of sister chromatid exchange frequency. *The Journal of the Histochemistry and Cytochemistry*, 1977. DOI: 10.1177/25.7.70454.

[27] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. *Proc. Symp. Geometry Processing*, pages 61–70, 2006.

[28] C. Zhang and T. Chen. Efficient feature extraction for 2d/3d objects in mesh representation. In *Proc. Int. Conf. Image Processing*, volume 3, pages 935–938, 2001.

[29] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Lecture Notes in Computer Science*, volume 9907, pages 501–518, 2016.

[30] T. E. Oliphant. SciPy: Open source scientific tools for python. *Computing in Science and Engineering*, 2007. DOI: 10.1109/MCSE.2007.58.

[31] W. McKinney. Data structures for statistical computing in python. In *Proc. Conf. Python in Science*, volume 445, pages 51–56, 2010.

[32] S. van der Walt, J. L. Schönberger, J Nunez-Iglesias, F. Boulogne, J.D. Warner, N. Yager, E. Gouillart, and T. Yu. scikit-image: Image processing in python. *PeerJ*, 2:e453, 2014.

ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.27, No.1, 2019

Curb Detection for a Pedestrian Assistance System Using End-to-End Learning

Hasham Shahid Qureshi
Technische Universität Berlin
Marchstr. 23
10587, Berlin
qureshi@tu-berlin.de

Rebecca Wizcorek
Technische Universität Berlin
Marchstr. 23
10587, Berlin
wiczorek@tu-berlin.de

Abstract

Our goal is to develop an assistance system for supporting road crossing among older pedestrians. In order to accomplish this, we propose detecting the curb stone from the pedestrians' point of view. Curb detection plays a significant role in road detection and obstacle avoidance, etc. However, it also presents significant challenges such as the small size of the target as well as, obstacles and different structures. To tackle these problems, we chose to fuse two sensors, a Camera and a Leddar, and use an algorithm that applies an end-to-end learning approach. The convolutional neural network was chosen to process the images acquired from the mono camera by filming the curb and its surroundings. The artificial neural network was selected to process the point cloud data of the Leddar acquired in the form of arrays from the 16 channels of the Leddar. A prototype was developed for data collection and testing purposes. It consists of a structure carrying both sensors mounted on a walker. The data from both sensors were collected with multiple factors taken into consideration, such as, weather, light conditions and, approaching angles. For the training of algorithms, an end-to-end learning approach was selected where we labelled the complete image or array rather than labelling the individual pixels or points in the data. The networks were trained and, the features from the parallel networks were concatenated and given as the input to the fully connected layers to train the complete network. The experimental results show an accuracy of more than 99% and robustness of the end-to-end learning approach. Both sensors are relatively inexpensive and are in fusion together, they are able to efficiently accomplish the task of detecting the curb stone from the pedestrians' point of view.

Keywords

Deep learning, curb detection, pedestrian assistance system, end-to-end learning, Leddar, monocamera, multi sensor data fusion, convolutional neural networks, artificial neural networks

1 INTRODUCTION

For people of all ages, out-of-home mobility is an indispensable part of leading an independent and self-determined life [Lim09a]. Mobility and participation in the society is a crucial part in keeping functionality and also to prevent the disability [mol04a]. This makes mobility of older pedestrians an important topic. Since the elderly demographic of the population (adults of 65 years and older) in the world is increasing, older pedestrians' mobility in the traffic environment requires special attention.

Older pedestrians were involved in 20% of all road-traffic accidents in Germany in 2013 [Bun13a]. They are involved in more accidents than middle-aged people when walking distance is taken into account [Ryt06a]. Moreover, in comparison to other age groups, older pedestrians require a longer recovery time after road-traffic accidents and their fatality rate is four times higher [Bun13a]. According to official police statistics in Berlin (Germany), most of these accidents happen at official crossings such as zebra crossings and traffic lights, and the main cause is the lack of attention older pedestrians pay to oncoming traffic [Bra15a].

These facts mean it is important to understand the underlying problems older pedestrians face in traffic in order to develop appropriate solutions to help prevent these accidents. In order to increase safety and support older pedestrians, in our project FANS (Fußgänger-Assistenzsystem für ältere Nutzerinnen und Nutzer im Straßenverkehr - Pedestrian Assistance System for Older Road Users) we are currently developing an assistance system. This work comprises a user-centred approach which includes the future target group in the design process of the prototype.

Our project initially involved investigating the reasons behind older pedestrians' lower attention to traffic. Two reasons were established. Firstly, older people tend to examine the terrain more frequently and attentively than younger people, which demands visual attention. This additional attention-demanding task impares their ability to detect hazards in the street environment [Wic16a]. Secondly, when approaching the road, most

people watch out for cars while walking, which can be viewed as multitasking behaviour. The act of walking requires cognitive resources and thus decreases the frequency at which visual targets such as passing cars are detected [Pro17a]. Based on the findings mentioned above and the analysis of official accident statistics [Sta13a], we can outline the requirements for the assistance system.

The purpose of the assistance system is to warn users when they are approaching the road. The warning reminds them to stop whatever else they are doing (i.e., walking, scanning the ground) and direct all their attention towards the traffic. This notification should be given to the users at a predefined distance in order to prevent them stepping onto the road without checking for traffic first.

The most important requirement for the system is for it to work as reliably as possible. That means reducing the number of events where the system fails to detect the curb stone (misses) to a minimum. However, the frequency of false alarms (occurring when the user is not close to a road) should also be kept fairly low. This is because the experience of false alarms can lead to the ignorance of warnings due to users' distrust of the system [Dix07a], [Mad06a], [Bli95a].

In order to generate appropriate warnings, the assistance system must be able to effectively identify when users are approaching a road. It was decided that this should done by detecting the curb stone using a suitable sensor solution. The sensors should not be too heavy to avoid imposing excessive weight upon the users, as well as not being too expensive for the older target group to afford.

2 RELATED WORK

Curb stone detection is an important research aspect in the field of mobile robotics and is especially important in the field of autonomous vehicles. It is a crucial component in ADAS (Advanced Driver Assistance Systems) such as parking assistance, vehicle positioning, etc. However, this research focuses on curb detection from the perspective of the driver (i.e., the car). Research into curb detection from the point of view of pedestrians is relatively rare. This is because curb detection from a pedestrians' perspective proposes a separate relevance and, in particular, a pedestrian's angle of view is entirely different. However, we can still extract some useful information regarding the sensors, as well as theories proposed to solve this problem, from recent research into mobile robotics and intelligent vehicle systems.

For curb detection, methods vary with regard to types of sensors and processing methods, which all have several advantages and drawbacks. It can be categorized based on the types of sensors used. For example, standard approaches using an inexpensive mono camera exploit the methods based on appearance information (i.e., image processing) [Pri16a]. Image-processing-based techniques can allow detection from long distances, but they are susceptible to decreased accuracy which can be caused by changes in the intensity of images such as shadows or changes in road surfaces, road markings, etc.

However, most methods rely on the 3D information extracted from LiDAR (Light Detection and Ranging) or imaging sensors. As opposed to monocular cameras, stereo vision cameras can exploit 3D geometry and are therefore more suited to detecting curbs [Kel15a], [Sod16a], [Fer14a], [Sei13a], [Enz13a], [Oni11a], [Hu12a], [Sie10a]. Stereo vision can provide high-resolution information which is not available in other 3D sensors. Since they provide a high resolution, appearance and geometry features are used actively to detect curbs using stereo vision. The geometry features, such as the height step [Kel15a], curvature [Sod16a], [Fer14a] and height variation [Kel14a], [Sei13a], [Hu12a] are commonly used with stereo-vision-based methods to detect curbs. These methods are relatively efficient, however the sensors used in these techniques are comparatively expensive and a 3-D sensor needs a 360° view which contradicts with our requirements.

Several mapping methods are used for curb detection. Digital Elevation Models (DEM) are the most widely used [Oni08a], [Kel14a], [Sie11a], [Enz13b]. All of these approaches can augment noisy sensor data through local or temporal filtering. However, they suffer the drawback that the cell sizes affect the accuracy of road-curb features. Therefore, small cell sizes are favoured, which tend to require higher computational efforts, such as higher memory consumption, making them difficult or even impossible to use in real time.

Considering the requirements of our assistance system, we decided to carry out the sensor fusion using the deep learning method. Therefore, we used a mono camera and a range sensor, assuming that the fusion of these two sensors could detect the curb more efficiently. In our case, the use of an expensive multi-layer LiDAR, which requires a 360-degree field of view, was not a feasible option. Hence, we decided to use a Leddar sensor. Leddar [Oli15a] is a propriety sensor from Leddartech which works based on the principles of LiDAR technology. It can detect, locate and measure objects in its field of view. These sensors are mounted on a walker (see section 3.2) to avoid older people having to carry the assistance system on their body.

3 MULTI-SENSOR DATA FUSION USING END-TO-END LEARNING

We started our work by implementing the detection system with only one sensor. An end-to-end learning approach using Convolutional Neural Network (CNN) was chosen to detect the road and its surroundings from the pedestrian's point of view via a mono camera. This work was inspired by the work of Bojarski et. al. [Boj16a]. The authors implemented an end-to-end learning approach using a CNN in the context of autonomous driving. If a network is used in the context of end-to-end learning, it learns the whole processing pipeline without the need to label explicit parts of the data. For example, in the case of the image dataset, it is sufficient to label the whole image rather than labelling the individual pixels in the image, which saves considerable time during the annotation of the data (for further details, see [Qur18a]). The camera which has been used has the focal length of $4.0mm$ with the optical resolution of 1280×960 and has the maximum frame rate of 30fps @ 640×480.

In order to further improve detection accuracy, we integrated the Leddar as the second sensor in the system. The Leddar sensor is based on the optical time-of-flight technology which sends short light pulses about 10,000 times per second to actively illuminate the desired area. The sensors then capture the light that is scattered back from objects and processes the signals to accurately determine their location and other attributes, such as shape and design. In our project, we are using the Leddar M16, which is a 16-segment solid-state LiDAR sensor module. The Leddar M16 sensor module uses 16 independent detection channels to deliver continuous and precise detection combined with exceptional lateral discrimination. It has a detection range of 146m and a data acquisition rate of up to 100 Hz [Oli15a].

The Leddar has been mounted on a walker so that the channels are facing the curb stone vertically with an approximate angle of 45° in the predefined distance of $2m \pm 1m$. The schematics can be seen in Figure. 1.

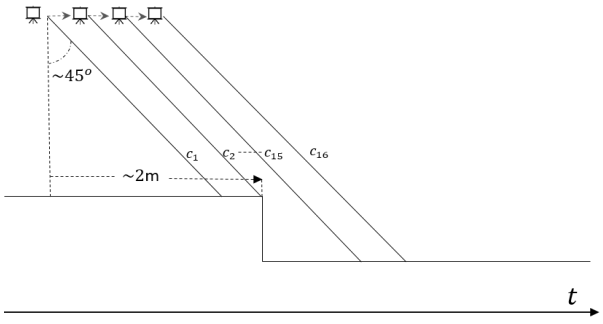


Figure 1: Schematics of Leddar sensor

In Figure 1, "c" represents the channel of the Leddar. If the Leddar is moved towards the curb stone and it approaches the predefined window, as there is a difference in height between the pavement and the road, there will be a change in value for one channel (channel 16) as the lateral distance between the Leddar and the deflecting surface increases. This means that, for channel number 16, at t = 0, the deflecting surface is the pavement and, at t = 1, the deflecting surface is the road. Therefore, a profile can be made where this channel indicates the difference, which provides an array of data in the shape $(16,1)$. Similarly, a profile can be made for each channel.

3.1 Proposed Methodology

For the camera images, we used the input plane of $(227,227,3)$, that means RGB images were used to train the CNN architecture in the first step. However, it is impossible to use the same CNN network to fuse both sensors because the dimensionality of the input data differs. Therefore, we decided to use two different algorithms to process the data of each sensor separately and merge them later at the classification stage. For the images, CNN was used as it provides the best results for learning the hidden patterns in images. In order to cater the effect of light in the RGB images, we decided to use the grey-scale images to train the CNN. In this way the algorithm neglects the effects of light (e.g. brightness, shadowing, etc.), especially in sunny conditions and this led to a better prediction. We also reduced the pixel size of the images to decrease the training time of the algorithm. This led to an input size of resolution $(225,225)$. However, Leddar information is comprised of point cloud data acquired from the 16 channels, with each channel representing one data point. Thus, the Artificial Neural Networks (ANN) were chosen because ANNs do not reduce the dimensionality of the input data, meaning no relevant information is lost. Moreover, ANNs are easy to fuse with CNNs and doesn't hinder the speed of the network.

To fuse the two sensors which have heterogeneous input streams, we trained two networks in parallel, before concatenating the features from both networks. These concatenated features were used as an input for the fully connected layers to establish the symmetry between both sensors. After this, the complete network was trained to tune the hyper parameters. These fully connected layers also serve the purpose of classification, however, in end-to-end learning, it is hard to determine which layers perform the feature-extraction task and which layers carry out the classification.

3.2 Data Collection

The data was collected using a prototype consisting of the following modules:

- Walker (Invacare Banjo P452E/3)
- Leddar M16

- USB Webcam HD C270 from Logitech
- Notebook (Lenovo Y720-15IKB)

Figure 2: Prototype used for data collection and testing

A customised structure was added to the walker to carry the sensors, as shown in Figure 2. A housing for the camera was fabricated using a 3D printer. The Leddar housing was integrated with a rechargeable power supply. Both of the sensors were connected to the notebook using the USB interface.

3.3 Camera dataset

In order to train our CNN network, a dataset was needed to represent the task at hand (curb stone detection from a pedestrian's point of view). As there was no prior dataset of this kind available, a new dataset had to be constructed from scratch. This dataset took into account different light and weather conditions, as the majority of the relevant accidents occur in conditions with reduced visibility [Sta13a], as well as different road environments.

Because the system is developed for use in the city of Berlin, the dataset incorporates the different types of pavements and curb stones found in the streets of Berlin. In order to achieve this, it was necessary to carry out an analysis of the existing pavement structures and how common they are. For further details, see [Qur18a].

In order to train the network, the dataset was divided into two classes labelled "positive" and "negative". Images labelled as "positive" show scenarios where users walk towards the road, whereas "negative" images depict scenarios where users are walking along the pavement parallel to the road, as shown in Figures 3 and Figure 4 respectively.

Figure 3: A few examples of the positive labelled images

Figure 4: A few examples of the negative labelled images

3.4 Leddar dataset

Since this problem was tackled as a binary classification problem, two types of data were also collected for the Leddar data, namely "positive" and "negative". The "positive" class relates to situations where the Leddar is facing the curb and the user intends to cross the road. For the "negative" class, data was collected about situations the Leddar was not facing the curb and the person had no intention of crossing the road. We used 16 channels of the Leddar which generate the point cloud data. Data was collected from each channel and a profile was made which gave us an array of size $(16, 1)$. Each array represents one data sample. The data was collected on the streets of Berlin with the help of the walker mentioned in section 3.2. While collecting the data, multiple aspects were considered (e.g. height of curb stone, angle of approach, distance from curb stone, parked cars, etc.).

3.5 Training Of Algorithm

3.5.1 Selection Of The Data

In order to train the network, it was important to select the adequate data from both of the sensors. The data

from the Leddar and the camera were collected separately, which means there is no connection between the data streams of the Leddar and the camera. We chose 40,000 frames from each sensor. The data from each sensor was divided into positive and negative databases at a 50:50 ratio. The dataset was then further divided into two parts: training and validation, at 70% and 30% respectively. The data from the camera was augmented using data augmentation techniques by adding rotations, artificial shifts, zooming and sheer effect, etc. to overcome the overfitting problems and also to teach the network the various conditions which present themselves in real-life situations (e.g. insufficient illumination, motion blur, etc.). However, the data from the Leddar was not augmented as each channel gives a number and adding artificial noise entirely changed the outcome.

3.5.2 Network Architecture

After the selection of data, experiments were conducted to find the best suitable network architecture. The training was started with the simplest case and the difficulty was increased gradually to monitor the performance of the network. In the beginning, only datasets without obstacles and objects were introduced as a positive category. Afterwards, the difficulty was increased by adding different factors such as leaves, obstacles and various weather conditions. Similarly, for the "negative" class, the difficulty level was increased by adding a range of different pavements and angles.

After extensive experimentation with the aim of achieving maximum accuracy, the network consisted of the following configuration. The final CNN architecture contained 5 layers with strided convolutions in all the convolution layers with a size of 2×2 and with a kernel size of 3×3. The ANN architecture had 4 layers with a varying number of neurons. Three fully connected layers are used after concatenating the features from CNN and ANN networks. These layers were then trained on the combined features from both networks (i.e. CNN and ANN). The complete network architecture can be seen in Figure 5.

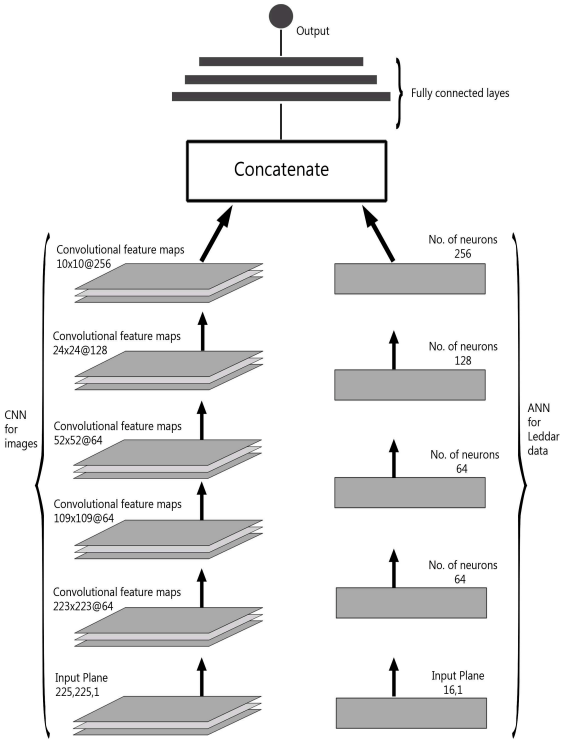


Figure 5: Network architecture

4 RESULTS

The efficacy of the model was determined through the accuracy and loss of training and validation. These values indicated the system's ability to learn the underlying features of the data. A validation accuracy of 99.04% and a validation loss of 0.043 were observed. The epoch by epoch analysis is shown in Figure 6. Another way to observe the efficacy of the model is the through confusion matrix. This confusion matrix demonstrates how many times the algorithm was not able to predict the correct label. The confusion matrix was plotted with 10,000 test samples and can be seen in Figure 7.

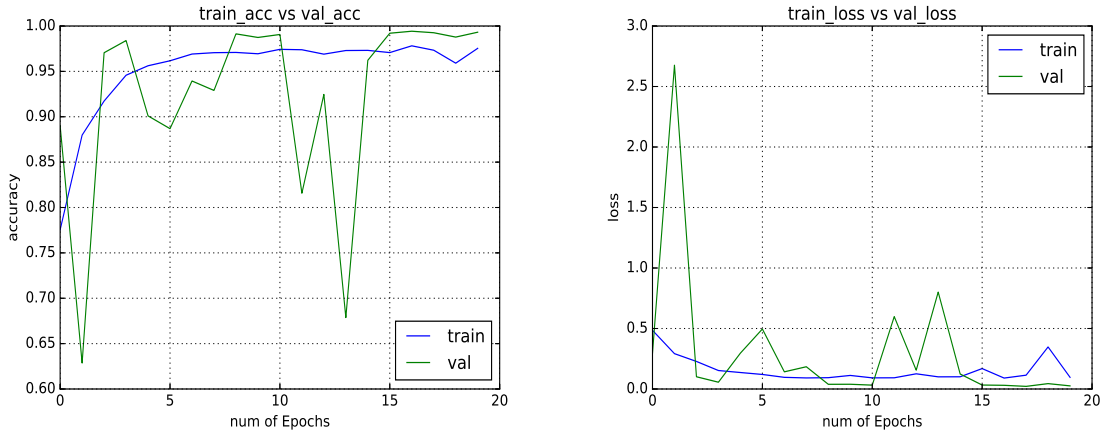
The Lenovo Y720-15IK notebook, which has NVIDIA GTX 1050 GPU, was used for real-time testing. The system runs at 22 Frames Per Second (FPS). In order to assess the performance of the system, we adopted the F1 score as an evaluation criterion for curb detection, which is calculated using precision and recall. The results are listed in Table 1.

Labels	Precision	Recall	F1 score
Positive	1.00	0.99	0.99
Negative	0.99	1.00	0.99

Table 1: Precision, recall and F1 score of the system

5 CONCLUSION

Based on the analysis of the target group, we deduced that the sensors chosen to detect the curb stone should be lightweight and inexpensive. In order to train the networks, the datasets were constructed from scratch, taking into account various factors such as light, weather and structural combinations. These datasets will be extended in future in order to account for a wider range of scenarios. Both of these datasets can be used as independent entities in other applications and systems and are therefore valuable irrespective of



(a) Accuracy graph (b) Loss graph

Figure 6: Simulation results for the network

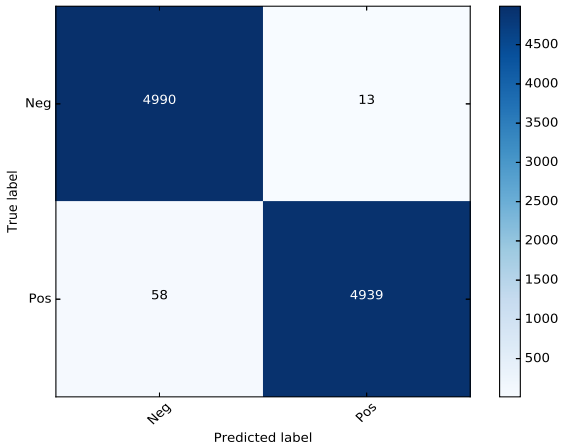


Figure 7: Confusion matrix for 10,000 test images

the algorithm. Additionally, a novel approach for the multi-sensor data fusion using the end-to-end learning technique is presented where two algorithms, CNN and ANN, were used to efficiently detect the curb stone in various scenarios. The fusion network was trained on a small amount of data, which in turn requires less training time. However, despite the minimal amount of data, the network was able to generalise well and detected the curb stone with an efficiency of more than 99%. The system worked reliably in different conditions, with very little time required for prediction. We believe that end-to-end learning can be effective for multi-sensor data fusion. This technique proved to be very fast and reliable as there was no need for hand-crafted rules or labels for the data, saving a tremendous amount of time. By fusing CNN and ANN using end-to-end learning algorithm, we proved that end-to-end learning is also capable of handling multiple algorithms at once. In future we will train the different algorithms to compare the accuracy and also observe whether end-to-end learning is able to handle more complex architectures. Finally, a field study will be conducted with the target group to evaluate the performance of the systems. We will also investigate the performance of the users in the detection of hazards as well as their acceptance and trust of the system.

6 REFERENCES

[Ryt06a] Rytz, M., .Senioren und Verkehrssicherheit. VCS Verkehrs-Club der Schweiz, Bern, 2006.

[Bun13a] Bundesamt, S., Unfallentwicklung auf Deutschen Strassen 2012: Begleitmaterial zur Pressekonferenz am 10. Juli 2012 in Berlin, 2013.

[Bra15a] S. A. B. Brandenburg, Verkehrsstatistiken. 2015.

[Wic16a] Wiczorek, R., Siegmann, J., and Breitinger, F. Investigating the impact of attentional declines on road-crossing strategies of older pedestrians. in Proceedings of the Human Factors and Ergonomics Society Europe, pp. 155-169, 2016.

[Pro17a] Protzak, J. and Wiczorek, R. On the Influence of Walking on Hazard Detection for Prospective User-Centered Design of an Assistance System for Older Pedestrians., i-com, vol. 16, no. 2, pp. 87-98, 2017.

[Dix07a] Dixon, S. R., Wickens, C. D., and McCarley, J. S. On the independence of compliance and reliance:Are automation false alarms worse than misses?, Human Factors, vol. 49, no. 4, pp. 564-572, 2007.

[Mad06a] Madhavan, P., Wiegmann, D. A., and Lacson, F. C. Automation failures on tasks easily performed by operators undermine trust in automated aids., Human Factors, vol. 48, no. 2, pp. 241-256, 2006.

[Bli95a] Bliss, J. P., Gilson, R. D., and Deaton, J. E. Human probability matching behaviour in

response to alarms of varying reliability., Ergonomics, vol. 38, no. 11, pp. 2300-2312, 1995.

[Kel15a] Kellner, M., Hofmann, U., Bouzouraa, M. E., and Stephan, N. Multi-cue, model-based detection and mapping of road curb features using stereo vision. in 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pp. 1221-1228, 2015.

[Sod16a] Sodhi, D., Upadhyay, S., Bhatt, D., Krishna, K. M., and Swarup, S. Crf based method for curb detection using semantic cues and stereo depth. in Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing, p. 41, 2016.

[Fer14a] Fernandez, C., Izquierdo, R., Llorca, D. F., and Sotelo, M. A. Road curb and lanes detection for autonomous driving on urban scenarios. in IEEE 17th International Conference on Intelligent Transportation Systems, pp. 1964-1969, 2014.

[Kel14a] Kellner, M., Bouzouraa, M. E., and Hofmann, U. Road curb detection based on different elevation mapping techniques. in IEEE Intelligent Vehicles Symposium, pp. 1217-1224, 2014.

[Enz13a] Enzweiler, M., Greiner, P., Knoppel, C., and Franke, U. Towards multi-cue urban curb recognition. in IEEE Intelligent Vehicles Symposium, pp. 902-907, 2013.

[Sei13a] Seibert, A., Hahnel, M., Tewes, A., and Rojas, R. Camera based detection and classification of soft shoulders, curbs and guardrails. in IEEE Intelligent Vehicles Symposium, pp. 853-858, 2013.

[Sie11a] Siegemund, J., Franke, U., and Förstner, W. A temporal filter approach for detection and reconstruction of curbs and road surfaces based on Conditional Random Fields. in IEEE Intelligent Vehicles Symposium, pp. 637-642, 2011.

[Oni11a] Oniga, F. and Nedevschi, S. Curb detection for driving assistance systems:A cubic spline-based approach. in IEEE Intelligent Vehicles Symposium, pp. 945-950, 2011.

[Hu12a] Hu, T. and Wu, T. Roadside curb detection based on fusing stereo vision and mono vision. in Fourth International Conference on Machine Vision (ICMV 2011): Computer Vision and Image Analysis; Pattern Recognition and Basic Technologies, 83501H, 2012.

[Sie10a] Siegemund, J., Pfeiffer, D., Franke, U., and Förstner, W. Curb reconstruction using conditional random fields. in 2010 IEEE Intelligent Vehicles Symposium, pp. 203-210, 2010.

[Oli15a] Olivier, P. Leddar optical time-of-flight sensing technology: A new approach to detection and ranging:A new approach to detection and ranging., 2015.

[Boj16a] Bojarski, M. et al. End to End Learning for Self-Driving Cars., CoRR, vol. abs/1604.07316, 2016.

[Qur18a] Qureshi, H. S., Glasmachers, T., and Wiczorek, R. User-Centered Development of a Pedestrian Assistance System Using End-to-End Learning. in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 808-813, 2018.

[Oni08a] Oniga, F., Nedevschi, S., and Meinecke, M. M. Curb detection based on a multi-frame persistence map for urban driving scenarios. in 2008 11th International IEEE Conference on Intelligent Transportation Systems, pp. 67-72, 2008.

[Enz13b] Enzweiler, M., Greiner, P., Knoppel, C., and Franke, U. Towards multi-cue urban curb recognition. in IEEE Intelligent Vehicles Symposium, pp. 902-907, 2013.

[Sta13a] Stab Des Polizeipräsidenten. (2013) Sonderutnersuchung fußgängerverkehrsunfälle in berlin 2013.

[Lim09a] Limbourg, M. and Matern, S., Erleben, Verhalten und Sicherheit älterer Menschen im Strassenverkehr: Eine qualitative und quantitative Untersuchung (MOBIAL). Köln: TÜV Media, 2009.

[Pri16a] Prinet, V., Wang, J., Lee, J., and Wettergreen, D. 3D road curb extraction from image sequence for automobile parking assist system. in 2016 IEEE International Conference on Image Processing (ICIP), pp. 3847-3851, 2016.

[mol04a] Mollenkopf, H. et al. Social and behavioural science perspectives on out-of-home mobility in later life:Findings from the European project MOBILATE., European Journal of Ageing, vol. 1, no. 1, pp. 45-53, 2004.