

Proceedings of the 20th Central European Seminar on Computer Graphics

April 24 - 27, 2016
Smolenice, Slovakia
Co-organized with SCCG



Institute of Computer Graphics and Algorithms
Vienna University of Technology



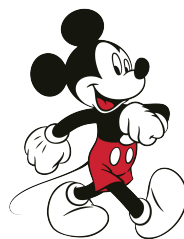
Spring
conference
on computer
graphics



Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava



Sponsors



Waltzing
Atoms
the atomic toolbox



Disney Research



Slovak Society of
Computer Science



NVIDIA



OESTERREICHISCHE
COMPUTER GESELLSCHAFT
AUSTRIAN
COMPUTER SOCIETY

Impressum

Vienna University of Technology
Institute of Computer Graphics and Algorithms
Favoritenstraße 9-11/186
1040 Vienna

ISBN 978-3-9502533-8-2

Welcome to CESC G 2016!

This book contains the proceedings of the 20th Central European Seminar on Computer Graphics, short CESC G, which continues a history of very successful seminars. Again this year, CESC G proceedings have an ISBN (978-3-9502533-8-2) and will therefore remain retrievable as long as there are libraries!

The long history of CESC G has started in 1997 in a medium-sized lecture room in Bratislava, bringing together students from Bratislava, Brno, Budapest, Graz, Prague, and Vienna. The idea found wide appraisal and the seminar moved to the beautiful castle of Budmerice, where it was held for 8 consecutive years, constantly growing in size and attraction. It was just in the 10th anniversary year 2006 that CESC G had to take a detour to move to Častá-Papiernička Centre, while it was back in Budmerice castle in 2007. Unfortunately, since 2011 the Budmerice castle is not available for scientific activities. After spending the one year in Viničné, in 2012 we moved to the beautiful castle in Smolenice.

Who are the CESC G heroes who made this year's seminar happen? In no particular order – because many people were involved equally – we would like to thank the organizers from Vienna: **Michael Wimmer**, Anita Mayerhofer, Katharina Krösl, Thomas Trautner, and Werner Purgathofer. Special thanks goes to **Martin Ilčík** for taking care of the complete reviewing process and scientific program preparation. We are very thankful to the CESC G organizers from Bratislava, mainly **Andrej Ferko**, always an inspiration to CESC G; Ela Šikudová, and Michal Ferko for the excellent preparations and on-site organization.

The main idea of CESC G is to bring students of computer graphics together across boundaries of universities and countries. We mainly focus on sustainable academic and research development in the field of Computer Graphics in Visegrad Countries and Austria. Our mission is to support undergraduate talents in their future careers. Therefore, we are proud to state that we have achieved again a high number of 16 participating institutions and a tight time schedule of 19 valuable student works, 3 specialized interactive workshops, and two invited talks. We welcome groups from Bratislava (UK and STU), Slovakia; Brno (VUT and MU), Plzeň and Prague (CTU and KU), Czech Republic; Budapest (BME), Hungary; Bonn, Germany; Graz and Vienna (TU and VRVis), Austria; Szczecin, Poland; Bergen, Norway; Maribor, Slovenia; and Sarajevo (UnSa), Bosnia and Herzegovina.

We assembled an International Program Committee of 14 members, allowing us to have each paper reviewed by three IPC members during the informal reviewing process. We would like to thank the members of the IPC for their contribution to the reviewing process. The IPC of CESC G 2016 consists of:

Vanda Benešová
Jiří Bittner
Andrej Ferko
Helwig Hauser
Michael Kenzel
Ivana Kolingerová
Radosław Mantiuk

Selma Rizvić
Michael Schwärzler
Jiří Sochor
László Szirmay-Kalos
Michael Wimmer
Borut Žalik
Pavel Zemčík

The reviewing process was further supported by: Zuzana Berger Haladová, Michael Birsak, Martin Brunnhuber, Zuzana Černeková, Michal Domanski, Michal Ferko, Christian Freude, Christian Hafner, Michael Hecher, Bernhard Kerbl, Michael May, Haichao Miao, Przemysław Musialski, Stefan Ohrhallinger, Michal Piovarči, Reinhold Preiner, Mohamed Radwan, Hiroyuki Sakai, Elena Šikudová, Bernhard Steiner, Ivana Uhlíková, Ivana Varhaníková, and Károly Zsolnai-Fehér.

The first invited talk “Visual data science – Advancing science through visual reasoning” will be held by Torsten Möller from the Research Group Visualization and Data Analysis of the Vienna University, Austria. The second invited talk by Jaroslav Krivánek from the Computer Graphics Group at the Department of Software and Computer Science Education of the Charles University, Czech Republic, will be about “Realistic Rendering in the ArchViz and Visual Effect Industries: When Academic Research Meets Practice”. Martin Ilčík from TU Wien will train the students “How to pitch” and he will also guide the students in the second workshop called “The Hero’s Journey in Science”.

To celebrate the 20 years of the seminar, Martin initiated the CESC G EXPO project. 10 companies specialized on visual computing will present their innovative products in a small interactive exhibition on Sunday afternoon. Our partners are:

Bohemia Interactive, Prague,	Photoneo, Bratislava,
Capturing Reality, Bratislava,	Vectary, Bratislava,
Escape Motions, Piešťany,	Vis Gravis, Bratislava,
Keen Software House, Prague,	VRVis, Vienna,
Lost in the Garden, Vienna,	Waltzing Atoms, Vienna.

The seminar is co-organized with the Spring Conference on Computer Graphics (SCCG), which takes place right after. The organization of a seminar where there are only low expenses for the students requires funding. We are very thankful to the sponsors of CESC G 2016:

- **Disney Research**, Innovations in Entertainment Technologies,
- **NVidia**, The Way It’s Meant to Be Played,
- **VRVis**, Research Center for Virtual Reality and Visualization,
- **OCG**, The Austrian Computer Association,
- **SISp**, Slovak Society for Computer Science,
- **Waltzing Atoms**, The atomic toolbox,
- **Eurographics**, The European Association for Computer Graphics.

Please note that the electronic version of these proceedings is also available at <http://www.cescg.org/CESC G-2016/>.

April 2016,

Michael Wimmer, Jiří Hladůvka, Martin Ilčík.

Table of Contents

Invited Talks

Realistic rendering in the ArchViz and visual effect industries: When academic research meets practice	3
<i>Jaroslav Křivánek. Charles University, Czech Republic</i>	
Visual data science – Advancing science through visual reasoning	5
<i>Torsten Möller. Vienna University, Austria</i>	

Real-time Rendering

Rendering high detail models from displacement maps	9
<i>Martin Volovář. Slovak University of Technology, Slovakia</i>	
Real-time cast shadow contours	17
<i>Péter Barabás. Budapest University of Technology and Economics, Hungary</i>	
Configurable rendering effects for mobile molecule visualization	25
<i>Lukas Prost. Vienna University of Technology, Austria</i>	

Visualization

Sonoco: Interactive visual comparison of filtering operations on time-dependent medical imaging data	35
<i>Deniz Gezin. University of Bergen, Norway</i>	
Interactive visual analysis of animal trajectories in a T-Maze	39
<i>Fabrizia Bechtold. VRVis, Austria</i>	

Image Processing & Vision

Recognition of important features of triangulated human head models	47
<i>Kateřina Kubásková. University of West Bohemia, Czech Republic</i>	
Segmentation of brain tumors from magnetic resonance images using adaptive thresholding and graph cut algorithm	55
<i>Zuzana Bobotová. Slovak University of Technology, Slovakia</i>	
Classification of built-up areas in LiDAR data based on second-generation connectivity filters	63
<i>Robi Cvirn. University of Maribor, Slovenia</i>	
Wavelet-based hierarchical heightmap compression method	71
<i>Michal Lařan. Charles University, Czech Republic</i>	

Augmented Reality & Interaction

Foreground detection and prototyping of photographic composition on Android	81
<i>Marek Salát. Brno University of Technology, Czech Republic</i>	
Natural interaction with small 3D objects in virtual environments	87
<i>Irfan Prazina. University of Sarajevo, Bosnia and Herzegovina</i>	
Generation of lecture notes as images from recorded whiteboard and blackboard based presentations	93
<i>Ondrej Jariabka. Comenius University, Slovakia</i>	

Modeling & Simulation

Dynamic simulation of virtual agents and obstacles in virtual cities	103
<i>Roman Mankovecký. Masaryk University, Czech Republic</i>	
Procedural generation using grammar based modelling and genetic algorithms	111
<i>Karl Haubenwallner. Graz University of Technology, Austria</i>	
Guided 2D modeling of 3D buildings using oriented photos	119
<i>Lisa Kellner. VRVis, Austria</i>	

Perception

Simulation of the luminance adaptation of the human visual system to varying background illumination	129
<i>Marek Wernikowski. West Pomeranian University of Technology, Poland</i>	
Using perception-based filtering to hide shadow artifacts	137
<i>Felix Kreuzer. Vienna University of Technology, Austria</i>	
Acceptable system latency for gaze-dependent level of detail rendering	145
<i>Michał Chwesiuk. West Pomeranian University of Technology, Poland</i>	

State of the Art

State of the art in real-time registration of RGB-D images	155
<i>Patrick Stotko. University of Bonn, Germany</i>	

Sponsors of CESC G 2016

Invited Talks

Realistic Rendering in the ArchViz and Visual Effect Industries: When Academic Research Meets Practice

Jaroslav Křivánek

Charles University
Czech Republic

Abstract

Research and practice of computer graphics has witnessed a renewed interest in realistic rendering based on physics-based Monte Carlo light transport simulation. This effort is propelled by the desire to accurately render arbitrary environments with complex geometry, materials and light sources, which is often difficult with the once industry-standard, but now obsolete, ad hoc rendering solutions. For this reason, the movie and archviz industries now rely on physically-based rendering methods, which poses new challenges in terms of strict requirements on image quality, algorithm efficiency and robustness, as well as usability.

In this talk, I will summarize some of my research contributions in the area of realistic rendering using physically-based light transport simulation that have been adopted by some of the major companies in the field such as Weta, PIXAR or Chaos Group. I will then juxtapose these academic results to my industry experience gained through the design and development of Corona Renderer in the company Render Legion, that I've recently co-founded. I will discuss the applicability and relevance of my research results to the world of production rendering for architectural visualization. I will conclude with some open challenges both in research and practice of physically-based realistic rendering.

Bibliographical Details

Jaroslav Křivánek is a researcher and associate professor of Computer Science at the Faculty of Mathematics and Physics of Charles University in Prague, and a co-founder of the the Render Legion company - developer of Corona Renderer. Prior to these appointments, he was a Marie Curie research fellow at Cornell University, and a junior researcher and assistant professor at Czech Technical University in Prague. Jaroslav received his Ph.D. from IRISA/INRIA Rennes and Czech Technical University (joint degree) in 2005. His primary research interests are global illumination, radiative transfer (including light transport), Monte Carlo methods, and visual perception. His research is driven by the goal of developing novel practical ways of producing realistic, predictive renderings of virtual scenes. The technologies he has co-developed are used, among others, by Weta Digital, PIXAR Animation Studios, or Sony Pictures Imageworks, and, of course, in Corona. In 2014, Jaroslav was selected for the New Europe 100 list, “a list of outstanding challengers who are leading world-class innovation from Central and Eastern Europe for taking computer graphics to the next level”.

Visual Data Science – Advancing Science Through Visual Reasoning

Torsten Möller

University of Vienna
Austria

Abstract

Modern science is driven by computers (computational science) and data (data-driven science). While visual analysis has always been an integral part of science, in the context of computational science and data-driven science it has gained new importance. In this talk I will demonstrate novel approaches in visualization to support the process of modeling and simulations. Especially, I will report on some of the latest approaches and challenges in modeling and reasoning with uncertainty. Visual tools for ensemble analysis, sensitivity analysis, and the cognitive challenges during decision making build the basis of an emerging field of visual data science which is becoming an essential ingredient of computational thinking.

Bibliographical Details

Torsten Möller is a professor at the University of Vienna, Austria, since 2013. Between 1999 and 2012 he served as a Computing Science faculty member at Simon Fraser University, Canada. He received his PhD in Computer and Information Science from Ohio State University in 1999 and a Vordiplom (BSc) in mathematical computer science from Humboldt University of Berlin, Germany. He is a senior member of IEEE and ACM, and a member of Eurographics. His research interests include algorithms and tools for analyzing and displaying data with principles rooted in computer graphics, image processing, visualization and human-computer interaction.

He heads the research group of Visualization and Data Analysis. He served as the appointed Vice Chair for Publications of the IEEE Visualization and Graphics Technical Committee (VGTC) between 2003 and 2012. He has served on a number of program committees and has been papers co-chair for IEEE Visualization, EuroVis, Graphics Interface, and the Workshop on Volume Graphics as well as the Visualization track of the 2007 International Symposium on Visual Computing. He has also co-organized the 2004 Workshop on Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration as well as the 2010 Workshop on Sampling and Reconstruction: Applications and Advances at the Banff International Research Station, Canada. He is a co-founding chair of the Symposium on Biological Data Visualization (BioVis). In 2010, he was the recipient of the NSERC DAS award. He received best paper awards from IEEE Conference on Visualization (1997), Symposium on Geometry Processing (2008), and EuroVis (2010), as well as two second best paper awards from EuroVis (2009, 2012).

Real-time Rendering

Rendering High Detail Models from Displacement Maps

Martin Volovár*

Supervised by: Peter Drahoš†

Institute of Applied Informatics
Faculty of Informatics and Information Technologies
Slovak Technical University
Bratislava / Slovakia

Abstract

In this paper, we present a method to generate a high resolution mesh from low poly mesh directly on GPU to reduce bandwidth overhead between GPU and CPU. We use known methods such as subdivision surface, displacement mapping and adaptive tessellation to generate more geometry in certain parts where it is necessary. This method is suitable for animation because small numbers of control points are modified. The main aim of this work is effective render a high quality mesh in the real time.

Keywords: Vector displacement map, Adaptive tessellation, Feature adaptive subdivision

1 Introduction

Since the first GPU has been released, GPU performance has been highly increased. Modern high-end GPUs can render around 6 billion triangles per second [11]. Memory bandwidth and an I/O latency has been improved too, but not as much as GPU render speed. What was not limiting factor before, is now a performance bottleneck. Transferring data between CPU and GPU is not a problem if a model geometry is static. In case of a model animation, modifying complex objects on CPU and updating GPU buffers can be impossible for every frame [9]. Motivation is to transfer only small parts of data and calculate model on GPU instead of transferring whole updated model. These parts could be changed position of control vertices or a changed sub-image of a displacement map.

Further motivation is to take advantage of adding detail dynamically in certain parts of model. This allows to change a complexity of a model according to its flatness and a screen space area. There is a similar method named LOD (Level of detail), which uses pre-generated models in different resolutions, but that method requires more memory and there is a problem in a continuity when the resolution is switched.

2 Background

This section describes methods to generate high detail mesh from control mesh.

2.1 Subdivision surfaces

Smooth surfaces often occur in the nature. Traditional method, polygon surface, requires many polygons to approximate a smoothness [4]. Geometric modelling of complex models is problematic. In the past, memory for storing complex models was expensive. Using subdivision surfaces it is possible save memory storage.

Subdivision surfaces are a curved-surface representation defined by a control mesh [2]. Subdivision surface smooths initial model using recursive subdivision algorithm [3]. Subdivision level depends on how many subdivision steps are required. Subdivision step has two stages: mesh refinement and vertex placement. Mesh refinement subdivide every face and edge. Vertex placement set vertex position according to subdivision rule. Position is calculated by linear interpolation of neighbour vertices.

Hypothetical surface created after an infinite number of subdivision steps is called limit surface [2]. The limit surface has often C^2 continuity everywhere, except at extraordinary vertices [10]. The most well-known subdivision algorithm for quad meshes is Catmull-Clark.

Adaptive subdivision allows to use different subdivision level on certain parts of mesh. Adaptive subdivision uses flatness test to avoid subdividing flat parts of model [2].

In feature adaptive subdivision method, the limit subdivision surface is described by a collection of bi-cubic B-spline patches [8]. This is advantage because patches can be rendered directly using hardware tessellation. Instead of uniform subdivision, where geometry complexity grows exponentially, using feature adaptive subdivision, complexity is close to linear [6].

2.2 Tessellation

Tessellation is the process of breaking patch into many smaller primitives [11]. Patch is defined as set of control points. Patch type can be line, triangle, quad, B-Spline,

*xvolovar@stuba.sk

†peter_drahos@stuba.sk

Bézier, etc. Tessellation factor controls fineness of patch. Adjacent edges should have the same tessellation factor because T-vertices could create a crack. Tessellation can convert quads to triangles, but common usage is to add geometric detail. Tessellation is now hardware accelerated.

2.3 Displacement mapping

Displacement mapping is using with tessellation to add high frequency detail with low memory I/O [7]. Instead of creating a smooth surface with subdivision surface, using displacement mapping it is possible make a rough surface.

A displacement map is a special type of texture where the stored values refer to a displacement of a vector. A commonly used type of displacement maps is the scalar displacement map, where each value corresponds to a displacement along the normal vector of a vertex. Scalar displacement map is easy to compute since normal vectors are cached. With vector displacement map it is possible displace vertex to any direction. Vector displacement map requires more memory than scalar because it uses all tangent vectors - t , b , n .

3 Our Contribution

Our solution combines feature adaptive subdivision scheme and displacement mapping with adaptive tessellation. Solution scheme is shown in the Figure 3. We use OpenSubdiv¹ library, because it supports subdivision and tessellation. An input for our program is a displacement map and control mesh. The mesh contains vertex positions, UV coordinates and indices. The input model should have a quad topology because we use a Catmull-Clark subdivision scheme. Catmull-Clark scheme can produce undulating artifacts (Figure 1). Faces should not overlap in texture space because we need 0..1 to 1 mapping between surface and texture space. The output is effective rendering of high resolution model.

3.1 Preprocessing

Input mesh is subdivided by feature adaptive subdivision algorithm. Current implementation of OpenSubdiv produces only bi-cubic patches for feature adaptive refinement². Bi-cubic patches can approximate smooth surface like subdivision scheme [5]. UV coordinates of the new vertices are linearly interpolated from control points.

Input displacement map is filtered by Laplace filter (Equation 1) with the aim to identify which parts require a higher tessellation factor. Instead of filtering displacement map it is possible filtering the normal map. The advantage of doing this is that values in Laplace map depends

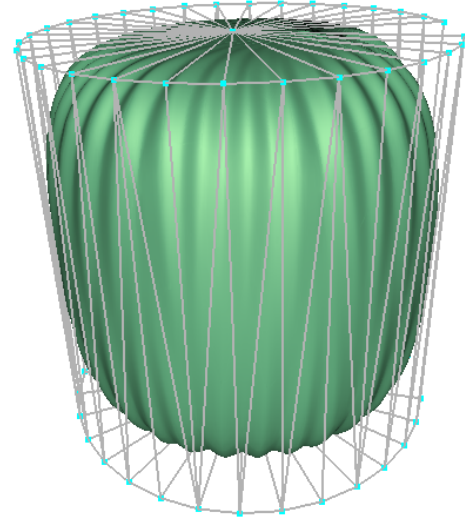


Figure 1: Catmull-Clark subdivision can behave poorly on triangle topology. Wireframe model is control mesh.

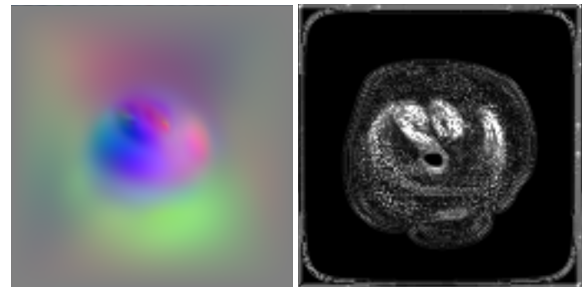


Figure 2: Example of displacement map and calculated Laplace map. The Laplace map is used in adaptive tessellation to affect tessellation fineness.

on displacement map strength. In the Figure 2 is example of displacement map³ and calculated Laplace map.

$$D^2xy = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (1)$$

We generate a normal map for correct lighting, because application of displacement map can change direction of normal vectors. Normal map is obtained using FBO (Frame buffer Object). When we render the model and use normal map, we apply the normal map shading in Fragment Shader. This can reduce continuity problems when adaptive tessellation is on. Model also looks more detailed when it is low poly. We use two approach of generating the normal map.

First approach is described in Algorithm 1. Model is tessellated with high tessellation factor to get high quality model, so we can use the normal map for different LOD.

¹<http://graphics.pixar.com/opensubdiv/docs/intro.html>

²http://graphics.pixar.com/opensubdiv/docs/subdivision_surfaces.html

³<http://content.luxology.com/asset/exref/fd5171e820982995daaf5e15db00955d.png>

Next, we apply displacement mapping. Calculating normal vectors is done by using Geometry Shader. This is because Geometry Shader has access to all points in triangle. Normal vector is obtained using cross product of two vectors that lied on triangle. Normal vector is normalized to unit length. Since all calculations are done in object space, normal vector is transformed to tangent space. To get normal map, whole model is rendered in texture space and colors are set respectively to normal vector.

Data: BSpline patches, displacement map
Result: normal map
foreach *patch* **do**
 | tessellate with high tessellation factor;
end
foreach *tessellated triangle* **do**
 | apply displacement mapping;
 | calculate normal vector via cross product;
 | transform normal vector to tangent space;
 | set position to UV coordinate;
 | transform position from UV space to NDC
 | (Normalized Device Coordinates);
 | set normal vector as output color;
end

Algorithm 1: Generating normal map from a displacement map.

Problem with this approach is that shading is naturally flat. This is because whole tessellated triangle have the same normal vector. Shading artifact depends on texel size of tessellated triangle. Figure 4 shows shading aliasing, where is normal map with different resolution applied.

Second approach uses linear interpolation of normal vectors of vertices in tessellated triangle. In this approach we do not use Geometry Shader. In the Figure 6 is described how we get the normal vector of vertex P . We use two near points P_A and P_B , where distance from actual point P is e in direction t and b tangent vectors. Like in the Section 3.3 we apply displacement mapping and calculate new positions P' , P'_B and P'_A . Normal vector is obtained with cross product of two vectors that lies on triangle defined by vertices P' , P'_B and P'_A . Unlike, in the first approach all calculations are in tangent space, so there is no need to transform the normal vector. Parameter e affects blurriness of normal map (Figure 5).

3.2 Tessellation

For patches we use B-Splines, because they can approximate smooth surface. It is important to have the same outer tessellation factors along adjacent edges of patches. Otherwise, cracks can appear. There is also problem that patches can have different size. Two adjacent patches can have the same subdivision level or level differs by one. In the Figure 9 two cases are present, where T-vertices appears.

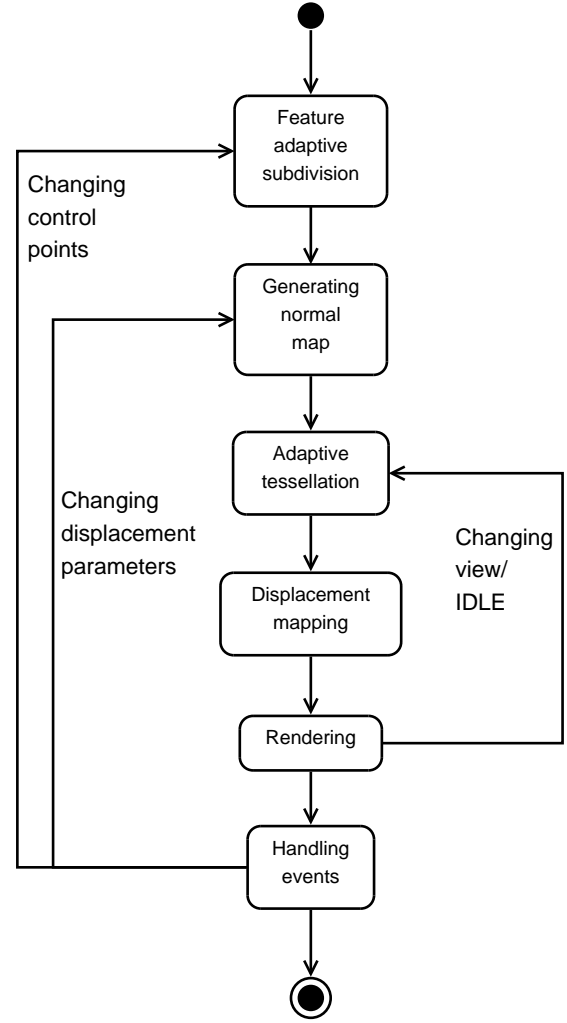


Figure 3: Overview of our method.

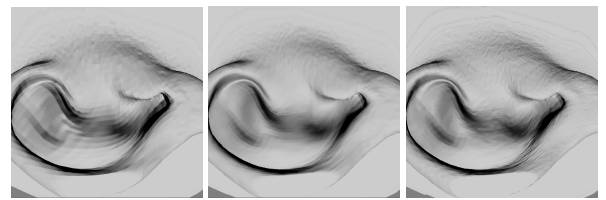


Figure 4: Shading aliasing of first approach of generating normal map depends on displacement map size (from left - 128×128 , 256×256 , 512×512).

Tessellation fineness of our solution depends on patch level of detail T_{LOD} , patch flatness T_F and tessellation quality T_Q . Tessellation coefficients T_{LOD} and T_F are used because adaptive tessellation requires them. Initially we measured T_{LOD} as edge length in screen space. There was problem if angle between patch edge and view vector was small. It is shown in the Figure 8. To avoid this artifact T_{LOD} is calculated as l/w in center of edge, where l is dis-

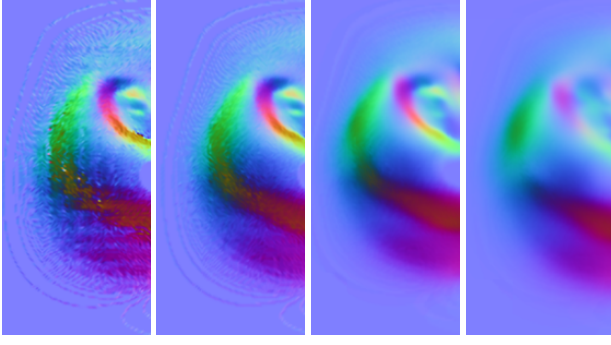


Figure 5: Parameter e (from left - 0.01, 0.02, 0.1, 0.2) affects blurriness of normal map in second approach of generating the normal map.

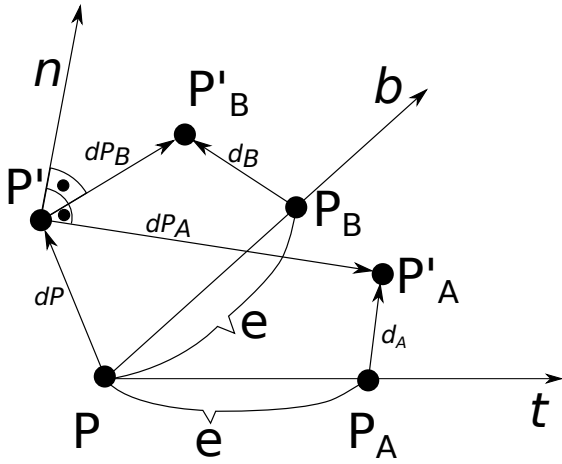


Figure 6: Calculating normal vector n in tangent space adding epsilon value e to P for vertices P_A and P_B on t and b axis. P' , P'_A and P'_B are positions after displacement mapping of vertices P , P_A and P_B . Normal vector is obtained using normalized cross product of vectors dP_B and dP_A .

tance of two corner points that lies on the edge. We use 14 tessellation factors 2 inner and 12 outer (Figure 7).

Inner tessellation factor for horizontal and vertical direction is calculated as:

$$T_{in} = T_{LOD} \cdot T_F \cdot T_Q, \quad (2)$$

where:

- T_{LOD} is average of l/w of middle edge points of patch in screen space. l is distance between two corner points that lies on patch edge in object space. l is scale correction, so T_{LOD} depends of patch size.
- T_F is texture value in Laplace map of center patch UV coord.
- T_Q is global value of patch quality. Slow GPU should has this value low.

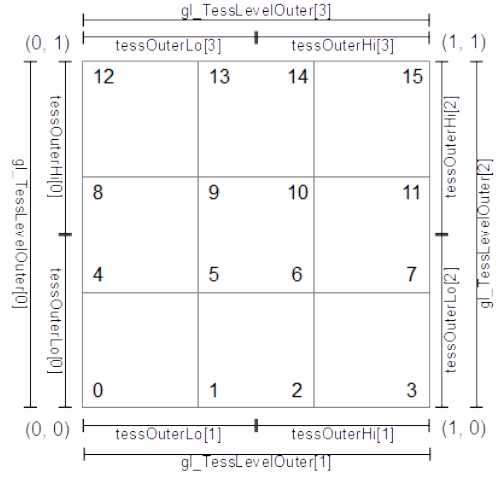


Figure 7: Outer tessellation factors for B-Spline patch used by Opensubdiv to avoid tessellation cracks. $tessOuterLo$ and $tessOuterHi$ are used in case transition edge (Edge connected with two smaller patch) [1].

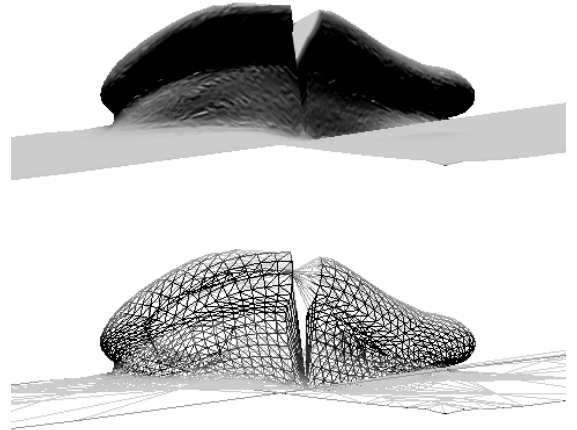


Figure 8: Tessellation artifact caused by using adaptive tessellation, where tessellation factor of edge is calculated as distance of two corner points, lies on this edge in screen space. Distance between corner points is small and edge has small outer tessellation factor.

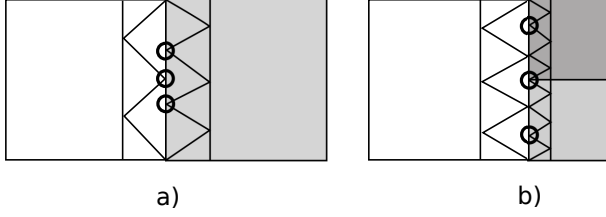


Figure 9: T-vertices occur in edge where patch has a) same subdivision level, but different outer tessellation factor on adjacent edge b) subdivision level that differs by one and *tessOuterLo*, *tessOuterHi* tessellation factors are not the same as *tessOuter* of adjacent edge of smaller patch.

T_{in} is rounded to nearest integer value and clamped, so minimal value can be 1. This is because inner tessellation with level less than 1 is undefined.

The easiest solution to avoid T-vertices is to set global outer tessellation factor to a constant value and in case of transition edge to double it. Problem with this solution is that inner tessellation factors change over surface and outer tessellation factor does not have to suit well. Our advanced method is based on simple idea that adjacent vertex has the same UV coordinate. We use the same Equation 2 like on inner tessellation factor, but coefficients T_{LOD} and T_F are calculated differently. For neighbour patches that uses same subdivision level, T_{LOD} is l/w of middle edge point in screen space. Else T_{LOD} is calculated as l/w of center in edge for *Lo* and *Hi* segments. If adjacent patch have the same subdivision level then T_F uses UV coordinate of middle point edge. In other case UV coordinate is chosen with weight $1/4$ and $3/4$ of corner UV coordinates because it is center of edge in smaller patch.

3.3 Displacement mapping

We use a vector displacement map. Tangent vectors - t , b , n are calculated from barycentric patch coordinates and patch parameters. Calculating a new position P' of a vertex in an object space is in the Equation 3.

$$P' = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + s \cdot \begin{bmatrix} t_x & b_x & n_x \\ t_y & b_y & n_y \\ t_z & b_z & n_z \end{bmatrix} \begin{bmatrix} d_r - o \\ d_g - o \\ d_b - o \end{bmatrix}, \quad (3)$$

where:

- P' is new position in object space after the displacement mapping.
- P is an old position in object space before the displacement mapping.
- t , b , n are unit tangent vectors defined in object space.
- s is a strength of displacement.
- d is a vector displacement value in displacement map.

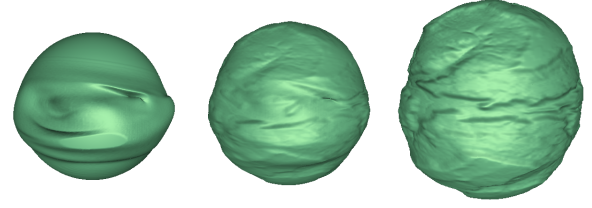


Figure 10: Morphing animation using linear interpolation of two displacement map.

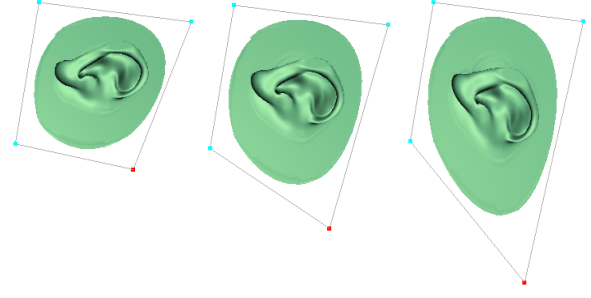


Figure 11: Animation, where control vertex is changing.

- o is a texture value, where displacement is zero. For unsigned displacement map it should be zero and for signed displacement map it should be set to 0.5.

4 Results

We can animate model changing displacement parameters or changing control points position. Our solutions can with a small amount of control vertices change shape of model (Figure 11). It is also possible to create morphing animation with linear interpolation of two displacement maps (Figure 10). Our solution uses adaptive tessellation, so generated geometry is view dependent (Figure 12). In our tests we use Nvidia GT 740M GPU and i7-4702MQ CPU.

Generating subdivision surface from control mesh is fast (Table 2) because we use feature adaptive subdivision rather than uniform subdivision. Feature adaptive subdivision is usually faster than uniform because feature adaptive subdivision uses fewer patches than uniform subdivision uses quads (Table 1). This is because bi-cubic patches can better approximate limit surface. However, quad uses only 4 vertices instead of 16 in case of B-Splines. Model complexity of feature adaptive subdivision grows linearly instead of exponentially. CPU time is measured via high resolution timer⁴ in beginning and ending of generating function. OpenGL functions can be asynchronous, so GPU time is measured using `GL_TIME_ELAPSED` query.

⁴<https://msdn.microsoft.com/en-us/library/windows/desktop/dn553408>

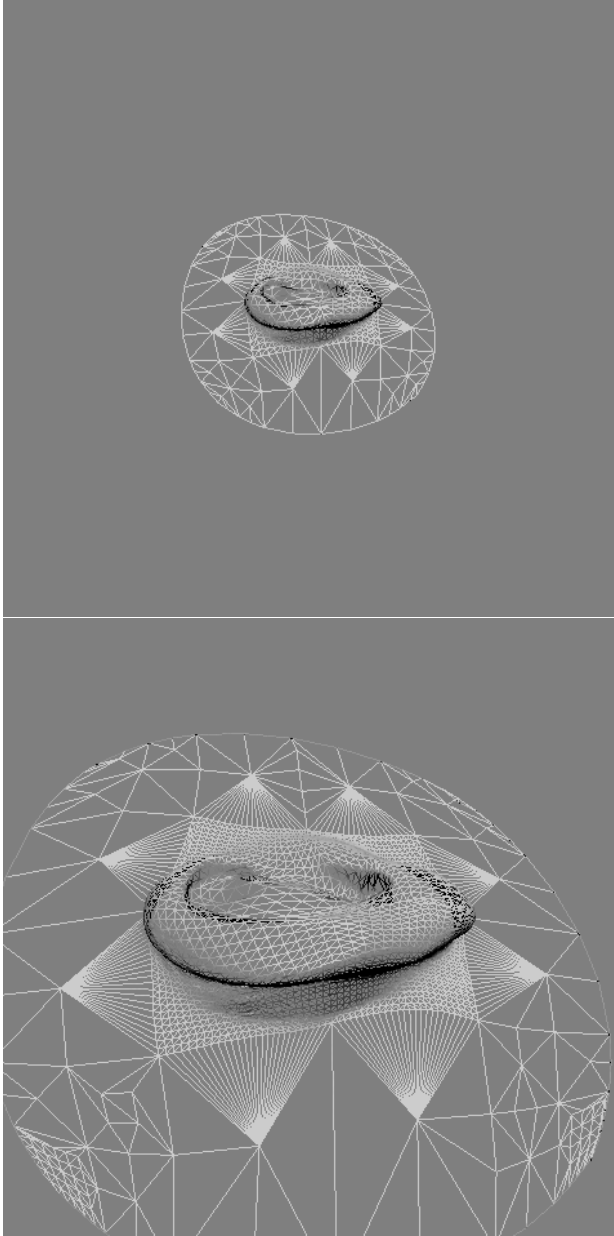


Figure 12: Generated geometry is view dependent.

Table 1: Geometry complexity of Feature Adaptive subdivision and Uniform subdivision.

Control mesh		Subdivision		
		level	Feature Adaptive	Uniform
Faces	Vertices		Patches	Quads
1	4	3	28	256
1	4	6	64	4096
1	4	8	88	65536
406	450	3	442	25984
406	450	6	478	1,663 M
406	450	7	490	6,652 M
4697	450	2	8926	74848
4697	4678	3	11098	0,299 M
4697	4678	5	15442	4,790 M

Table 2: Generating subdivision surface from control mesh.

Control mesh		level	Subdivision			
			Feature Adaptive		Uniform	
Faces	Vertices		Generating time [ms]		Generating time [ms]	
			CPU	GPU	CPU	GPU
1	4	3	1,62	0,018	0,016	0,004
1	4	6	1,71	0,019	0,767	0,001
1	4	8	1,74	0,017	11,42	0,017
406	450	3	1,79	0,003	9,139	0,003
406	450	6	1,80	0,065	649	0,002
406	450	7	1,86	0,064	2755	0,001
4697	450	2	1,73	1,024	231	0,001
4697	4678	3	1,72	3,670	110	0,010
4697	4678	5	2,14	5,865	1966	0,020

We use GPU Evaluator in feature adaptive subdivision, so GPU time grows faster than CPU.

Table 3 shows that second approach of generating the normal map is faster. We assume it is because first approach uses transformation to tangent space. In second approach is one more pipeline stage - Geometry Shader. Time of generating normal map depends mainly on model complexity.

Table 4 shows generating time of Laplace map in different resolutions.

In other test we compare render time between our method with tessellation and drawing raw array of triangles. We also test input memory size of our method and polygon surface method. We would rather compare render time between our method and polygon surface, but all connectivity information is lost. Our test model contains 4 control vertices with position and UV coordinate. Model size in both axis is one unit. Input control of our method mesh takes 0,1 kB. We used 128×128 , 256×256 and 512×512 RGB displacement texture. Displacement tex-

Table 3: Normal map generating time.

Triangles	Patches	Texture size	Time of first approach [ms]	Time of second approach [ms]
7048	28	256×256	0,541	0,986
7048	28	512×512	0,618	1,048
7048	28	1024×1024	0,719	1,282
1,037 M	442	256×256	9,256	21,348
1,037 M	442	512×512	9,438	21,277
1,037 M	442	1024×1024	9,954	21,544
29,623 M	11098	256×256	186,671	451,689
29,623 M	11098	512×512	188,061	451,899
29,623 M	11098	1024×1024	191,264	455,060

Table 4: Laplace map generating time.

Texture size	Time [μ sec]
256×256	73
512×512	261
1024×1024	970

ture has size of 48, 192 and 768 kB. We capture generated tessellated surface as transform feedback and draw again with polygon surface method. Render speed is measured using `GL_TIME_ELAPSED` between draw command. In case that generated triangles is more than value around 9000 our method is faster. Maybe, it is because of the memory overload. Number of triangles is obtained via `GL_PRIMITIVES_GENERATED` query.

In Tables 5, 6 and 7 estimated mesh size is present, if we used polygon surface method. We assume using uint16 index buffer, 32-bit float vertex buffer, 8 vertex attributes (3 - position, 3 - normal vector, 2 - UV coordinate) and triangle grid topology.

5 Conclusions

Our method allows to generate mesh from low poly control mesh. This method has some advantages: automatic generation LOD, animation with changing small number of control points, sculpting surface, faster animating, etc. It is possible that our method uses less memory.

There are some artifacts: aliasing in the normal map, continuity in adaptive tessellation, problematic UV mapping between closed surface and texture space. Future work can try to avoid these artifacts.

Table 5: Generated geometry and render time (displacement map 128×128)

Distance	Triangles	Our method render time [μ sec]	Triangle array render time [μ sec]	Polygon surface mesh estimated size [kB]
3,138	1258	53	19	27,0
1,722	3546	77	59	76,2
1,0	9834	158	187	211,3
0,513	11858	230	263	254,8

Table 6: Generated geometry and render time (displacement map 256×256)

Distance	Triangles	Our method render time [μ sec]	Triangle array render time [μ sec]	Polygon surface mesh estimated size [kB]
3,138	1202	54	20	25,8
1,722	3312	81	58	71,2
1,0	9386	163	185	201,7
0,513	11412	233	261	245,2

6 Acknowledgments

This work was supported by the Grant VEGA 1/0625/14.

References

- [1] Osd tessellation shader interface. http://graphics.pixar.com/opensubdiv/docs/osd_shader_interface.html. Accessed: 2016-02-10.
- [2] Michael Bunnell. Adaptive tessellation of subdivision surfaces with displacement mapping. *GPU Gems*, 2:109–122, 2005.
- [3] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design*, 10(6):350–355, 1978.
- [4] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice (2Nd Ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
- [5] Charles Loop and Scott Schaefer. Approximating catmull-clark subdivision surfaces with bicubic

Table 7: Generated geometry and render time (displacement map 512×512)

Distance	Triangles	Our method render time [μ sec]	Triangle array render time [μ sec]	Polygon surface mesh estimated size [kB]
3,138	1246	55	19	26,8
1,722	3462	84	60	74,4
1,0	9363	181	188	201,2
0,513	11464	254	264	246,3

patches. *ACM Transactions on Graphics (TOG)*, 27(1):8, 2008.

- [6] Matthias Nießner. *Rendering subdivision surfaces using hardware tessellation*. Verlag Dr. Hut, 2013.
- [7] Matthias Nießner and Charles Loop. Analytic displacement mapping using hardware tessellation. *ACM Transactions on Graphics (TOG)*, 32(3):26, 2013.
- [8] Matthias Nießner, Charles Loop, Mark Meyer, and Tony DeRose. Feature-adaptive gpu rendering of catmull-clark subdivision surfaces. *ACM Transactions on Graphics (TOG)*, 31(1):6, 2012.
- [9] Henry Schäfer, Benjamin Keinert, Matthias Nießner, and Marc Stamminger. Local painting and deformation of meshes on the gpu. In *Computer Graphics Forum*, volume 34, pages 26–35. Wiley Online Library, 2015.
- [10] P. Schröder, D. Zorin, T. DeRose, DR. Forsey, L. Kobbelt, M. Lounsbery, and J Peters. Subdivision for modeling and animation. *ACM SIGGRAPH Course Notes*, 12, 1998.
- [11] Graham Sellers, Richard S. Wright, and Nicholas Haemel. *OpenGL SuperBible: Comprehensive Tutorial and Reference*. Addison-Wesley Professional, 7th edition, 2015.

Real-time Cast Shadow Contours

Péter Barabás*

Supervised by: László Szécsi†

Computer Graphics Group

Department of Control Engineering and Information Technology

Budapest University of Technology and Economics

Budapest / Hungary

Abstract

This paper presents a real-time algorithm for drawing shadow contours in non-photorealistic rendering. We use the straightforward idea of intersecting shadow volumes with shadow receiver surfaces, proposing a practical scheme for accelerating the process on the GPU. Real-time operation is achieved by building a 2D bounding volume hierarchy (BVH) that relies on implicit spatial coherence in triangle mesh models.

Keywords: NPR, outline rendering, shadow volumes

1 Introduction

Photo-realism has been in the focus of rendering systems for decades. Photo-realistic rendering aims at creating images that are indistinguishable from real-world photographs, which is made possible by the precise simulation of physics laws during the rendering process. How accurately physics is applied in the rendering algorithm determines the level of realism of the result.

Computer graphics also tries to mimic artistic expression and illustration styles [6, 16, 18]. Such methods are usually vaguely classified as *non photo-realistic rendering* (NPR). While the fundamentals of photo-realistic rendering are in optics that are well understood, NPR systems simulate artistic behavior that is not mathematically founded and often seems to be unpredictable. Therefore, the first step of NPR is to model the artist by establishing a mathematical model describing his style, and then solve this model with the computer. The result will be acceptable if our model is close to the not formally specified artistic behavior. During the history of NPR, many individual styles were addressed. Many of those styles employ pen lines, pencil lines, or brush strokes to build an image. These elements are often used to draw outlines.

Outline visualization is extensively used in a wide range of applications, from CAD systems to stylized rendering. It can clarify the shape of a complex 3D object or may

highlight essential features. The human visual system processes seen images by identifying shapes separated by discontinuities. Outline rendering provides strong cues for shape separation, substituting for subtle and expensively rendered real-world cues like scattered lighting and shadows, and providing a stronger visual language in stylistic rendering. Cartoon shading, in particular, relies on edge visualization to convey shape information, in lieu of realistic shading.

This paper proposes a stylized rendering method where outlines are drawn to emphasize the contours of shadows, and describes a GPU-based real-time implementation. The organization of the paper is as follows. In Section 2 we summarize the related previous work on NPR, and outline rendering in particular. We explain why cast shadow contours received little attention, and evaluate the fitness of existing methods for this purpose. Section 3 introduces our approach. A detailed description of the final algorithm, and the discussion of results and future work conclude the paper.

2 Previous work

There are two well known approaches to outline rendering. The first one works in image space with the use of color, normal, and depth maps [15]. Edge pixels—those that lie near discontinuities in these maps—can be found using edge detection filters. What level of image-space discontinuity warrants outline edges must be adjusted by fine-tuning filter parameters and applying mask textures [17]. Object-space consistency of outlines during animations is also subject to those parameters. Cast shadow contours can easily be drawn if edges are detected on an untextured but shadowed rendering of the scene. The main problem with this approach is the excessive texture access bandwidth and the absence of real scalability in line features.

The other approach works in world space and generates new triangle strip geometry to visualize the outlines. In the following, we discuss methods in this category in greater detail.

There are two basic classes of outlines that are always drawn in line art, both indicating some kind of perceived

*medve9213@gmail.com

†szecsi.laszlo@gmail.com

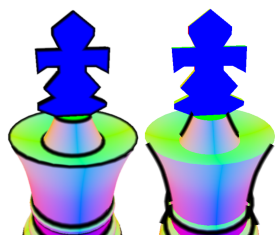


Figure 1: Crease (left) and silhouette (right) outlines.

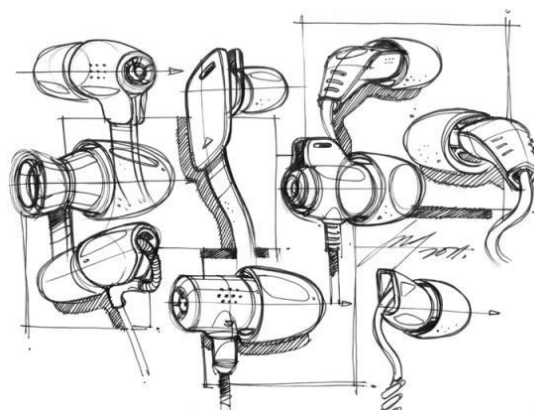
discontinuity (see Figure 1). *Silhouettes* appear at discontinuities in image space, where a continuous object surface appears to end. For manifold surface models this can happen only where the surface folds behind itself, meaning that outlines are located on the border of the visible (camera-facing) and not visible (back-facing) part of the object surface. The other class of displayed outlines—called *creases* or feature lines—indicate discontinuities in the surface normals, and they are defined by the topology of the mesh itself, independent of the view direction or the camera settings. In this paper, we take some ideas from conventional silhouette identification approaches, and discuss which are useful in finding shadow contours. Our presented results also include conventional outline rendering in addition to the newly proposed shadow contour outlines.

In addition to the silhouettes and creases discussed above, outline drawings may feature further lines. Suggestive contours [4] and apparent ridges [10] define outlines based on surface curvature characteristics. While these can provide superior visual cues, especially in absence of additional shading, they are less fit if we aim at minimal-overhead real-time rendering [3].

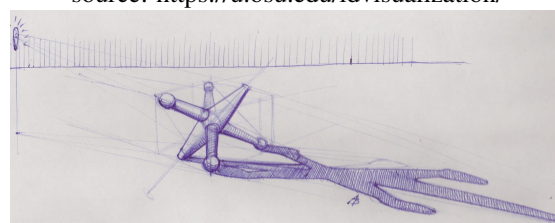
Cast shadow contours are yet another class of outlines that can appear in drawings. They have received less attention in research, both for artistic and technical reasons.

On the artistic side, cast shadow contours are relatively rarely drawn in technical or artistic images. Quite often, under natural illumination, shadows are supposed to have soft edges, and it is undesirable to draw attention to discontinuities in brightness due to cast shadows. When shadows need to appear hard, they are often rendered in solid black, making outlines not very prominent, even if drawn. However, where shadows need to be emphasized, especially in architectural or artistic sketches (Figure 2), shadow outlines are often drawn. Even in paintings, some strokes aligned on cast shadow contours are present (Figure 3).

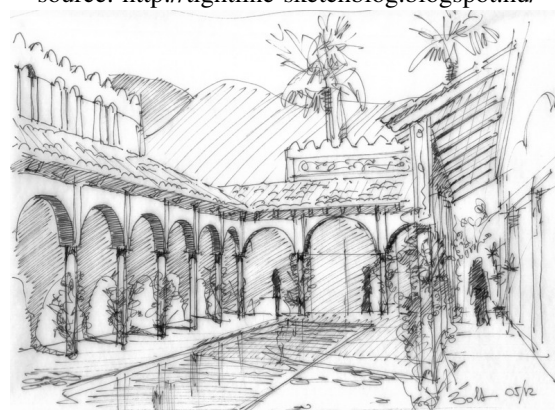
On the technical side, cast shadow contour generation is theoretically straightforward, and less prone to artifacts than silhouette outlines. Eisemann et al. [5] described the process of intersecting shadow volumes with the meshed shadow receiver surface. It requires the identification of the shadow caster silhouette as seen from the light source, and projecting it onto shadow receiver surfaces. We discuss these two phases in the following subsections.



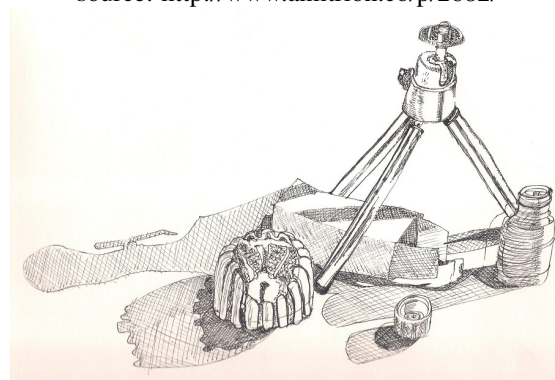
source: <https://u.osu.edu/idvisualization/>



source: <http://tightline-sketchblog.blogspot.hu/>



source: <http://www.anfitrion.co/p/2682/>



source: <https://alison512480.wordpress.com>

Figure 2: Architectural or artistic sketches with cast shadow contours.



Figure 3: The Night Café in Arles by Vincent van Gogh, watercolor.

2.1 Shadow volume generation

Extraction of the shadow caster silhouette is a well-known operation in *shadow volume* computation used for *stencil shadows*, which can be implemented in a GPU shader pass [2].

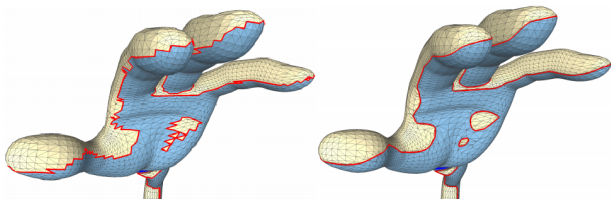


Figure 4: Markosian [14] (left), and Hertzmann–Zorin silhouettes [9] (right) seen from a viewpoint different from the camera position used for silhouette extraction. Figure taken from Benard et al. [1].

There are two basic ways to define silhouettes on triangle meshes. The approach by Markosian et al. [14] operates on the discrete triangle mesh geometry itself, selecting those edges as silhouette edges which separate front-facing and back-facing triangles. When rendered in solid color, the silhouette looks smooth, but—as shown in Figure 4—it is often ragged on the object surface. In the image plane, the edge loop can even turn back along the object silhouette multiple times, which becomes apparent if strokes are rendered semi-transparently. The definition by Hertzmann and Zorin [9] avoids this problem, as it considers the smooth surface instead of the triangulated one, reconstructing silhouettes from the vertex normals. For a given vertex \mathbf{a} with normal \mathbf{n}_a and vector \mathbf{c}_a to the camera, we define the scalar field $f(\mathbf{a}) = \mathbf{n}_a \cdot \mathbf{c}_a$, extending f to triangle interiors by linear interpolation. Silhouettes are taken to be the zeroset of f , yielding clean, closed polylines whose segments traverse faces in the mesh (rather than following edges, as in the Markosian method).

For stencil shadows, the Markosian-style silhouettes are extruded, as they provide artifact-free self-shadowing, and overly complex or back-tracking shadow volume boundaries do not influence the quality of the projected shadows. However, for the purposes of cast shadow contour rendering, these problems are just as relevant as for straight-forward silhouette rendering. Therefore, Hertzmann-and-Zorin-style silhouettes should be preferred.

When rendering silhouettes, hidden outlines should be removed. This is expensive to solve geometrically, thus screen-space methods are preferred. Depth testing is quite unreliable, and ID buffers were more successfully used [13]. For cast shadow outlines, there is no established practice. Geometric processing of shadow volumes would not be real-time, and adapting the ID buffer method is also not straightforward. Thus, we propose to solve the problem of removing cast shadow outlines due to hidden-from-light silhouettes in screen space (in Section 3.2).

2.2 Intersection

Eisemann et al. [5] described the process of intersecting shadow volumes with the meshed shadow receiver surface. Their purpose for extracting cast shadow contours was to transform 3D objects into 2D clip art. Therefore, real-time performance was not targeted and no acceleration scheme for the intersection was proposed.

Performing intersection in real time, however, is challenging in practice, as it is a crossbar on shadow volume and surface mesh faces, resulting in a naive algorithm of $\mathcal{O}(n^2)$ time complexity. Intersection tests can be accelerated using spatial subdivision schemes, but in dynamic scenes the cost of constructing those may be prohibitive.

In this paper, we show that we get a reasonably tight bounding volume hierarchy over the shadow volume faces, if we apply the *object median* subdivision scheme on the primitive stream generated by a contour-extruding geometry shader, without any additional ordering or cost heuristics. We compare intersection performance with that of a proper top-down object median subdivision scheme with object sorting to show that there is no significant performance penalty incurred.

3 New method

Generating cast shadow contours is a simple and straightforward problem in theory. By using the geometry of shadow volumes, we can easily find the intersections between shadow volume faces and shadow receiver faces. This would mean, however, that we would need to check every shadow volume face against every shadow receiver face. Even a moderately complex scene would impose a prohibitively large computation time, if we were to utilize this naive approach.

3.1 Intersection acceleration

We assume a perspective pinhole camera for a point light source, or an orthographic camera for a directional light source. Shadow caster silhouette edges—which are also faces of the shadow volume, when extruded—appear as 2D line segments when projected on the camera’s image plane. Our method relies on building a 2D *bounding volume hierarchy* (BVH) over these line segments. Traversing the BVH allows us to reduce the number of shadow volume faces we need to check for each receiver face. The contour edges constitute the leaves of the BVH tree, and nodes are *axis-aligned bounding boxes*, or AABBs—rectangles in this 2D case—, enclosing all child leaves. During traversal, we project each shadow receiver face to the camera plane, and check its AABB against the cells of the BVH recursively. If there is no intersection with a cell, its children are not traversed, filtering out most of the shadow volume faces that our shadow receiver face does not intersect.

A BVH is useful if it eliminates costly intersection computations. Thus, BVH cells should be as small as possible. This is often achieved by separating primitives into two locally coherent clusters, and repeating the process recursively to obtain a hierarchy. The clustering can be done by sorting primitives according to one (or more) of their spatial position coordinates, and splitting the sorted list into two parts. Splitting may be based on cost heuristics or simple strategies like the spatial median (similarly sized cells) or the object median (same number of elements in both cells) [8, 19].

Building a BVH that is efficient, however, suffers from the same performance problems that we aim to solve. Various methods for interactively building BVHs with GPU support have been proposed, one of the most relevant being *linear bounding volume hierarchies* (LBVH) [12, 11]. However, even this method requires sorting primitives at least once, with a severe performance impact for our application. In this paper, we investigate the effect of relying on the inherent spatial coherence in typical triangle mesh models, completely forgoing the sorting step. This means that we use the cast shadow contour’s edge primitives in the order they are written by the GPU after the silhouette detector shader. We use the object median splitting scheme, to avoid computation of cost heuristics, and obtain a balanced BVH tree, which is both easy to store and efficient to traverse on the GPU. This allows us to create the BVH in a bottom-up fashion, which can be solved efficiently using parallel *reduction* [7].

As triangle mesh geometries are typically modelled with some inherent spatial coherence or even optimized into triangle strips, our intuition was that even if the silhouette detection and the parallel stream processing introduce some randomness, the output contour segments would still exhibit sufficient coherence on a local scale. This would mean that the BVH built using this ordering is only sub-par on a few of the highest levels, compared to one built with

proper sorting, introducing a fairly constant, but relatively small overhead.

3.2 Hidden caster silhouette removal

Shadow caster silhouettes hidden from the light source appear on the shadow receiver surface as shadow contours that fall inside already shadowed areas. These inner shadow contours need to be filtered out, otherwise multiple objects casting overlapping shadows or concave shadow casters would cause erroneous contours to appear on receivers. To solve this, we utilized the information already available to us via shadow volume generation—the stencil buffer. In the stencil buffer, each texel has a value corresponding to the amount of shadow volumes it is contained in. This means we can use that information to check if a contour edge is inside a single shadow volume or not. Some ambiguity would be present, as all contours are exactly on the boundary of the shadow volume. In order to avoid the flickering caused by these inaccuracies, we offset all contours towards the inside of the shadowed surface, using the receiver surface normals and the shadow volume face normals.

4 Implementation

The implementation of such an algorithm is inherently multi-pass. The following steps provide an overview of what an implementation entails:

- shadow volume generation,
- rendering shadow volume faces and computing axis aligned bounding rectangles in the light source camera’s screen space,
- building the bounding volume hierarchy,
- checking for shadow volume–shadow receiver intersections using the bounding volume hierarchy.

Each step is implemented as a separate shader pass. Since we want to implement the bounding volume hierarchy builder using parallel reduction, using the GPU and a very short compute shader is a natural choice. Normally, only the result in the stencil buffer is used when implementing shadow volume based shadows, the geometry is thrown away. For our purposes, we need to access the geometry of the shadow volume in later passes. For this we use the stream output functionality of GPUs, emitting the world positions of the vertices composing the extruded faces of the volume.

Once we have the shadow volume faces, we feed the contents of the stream output buffer back into the pipeline for a pass rendered from the light source. The light source requires its view and projection matrices to be set-up—similar to the shadow mapping technique—since we are rendering from the viewpoint of the light. The shader run

in this pass is very straightforward, we just transform the primitives to the space of the light source, project them, and create the AABBs we need in the next step. The end results are streamed out as well, which gives us the actual order of the leaves later in the bounding volume hierarchy.

4.1 Building the BVH

We set up the buffer that will contain the bounding volume hierarchy using a representation that is similar to the array representation of the heap data structure. This helps us to easily calculate indexes of child nodes. The bounding volume hierarchy is created using a compute shader, which processes each level above the leaf level. The buffer containing the hierarchy is filled up from back to front, with leaves being the elements at the end, and their parents occupying the previous elements.

The only guarantee in stream output primitive order is that subsequent draw calls will take up subsequent regions in the stream output buffer. Since the primitive order defines the spatial coherence of the bounding volume hierarchy, we will need to measure performance against a bounding volume hierarchy constructed with proper sorting.

4.2 Tree traversal and edge generation

The final pass is where we find the actual shadow contours. A geometry shader runs on the shadow receiving meshes. As visible cast shadow outlines appear on faces that appear as front faces as seen both from the camera and from the light source, back faces in either aspect are culled. Then, we generate the light-screen-space axis aligned bounding rectangle of the primitive, and traverse the bounding volume hierarchy while checking against the axis aligned bounding rectangles of its nodes. For each traversed leaf we precisely calculate if there actually is any intersection with the primitive. If there is, we have to determine whether the contour is completely inside the primitive, or just partially, and calculate the actual intersection points accordingly.

Traversing a tree-like data structure is usually achieved by recursion, which is forbidden in shader code. Therefore, we implemented recursion using a small local stack. At each intersected node of the tree we push one child node on our self-managed stack and evaluate the other. After a traversal branch terminates because of a non-overlapping AABB or because of reaching a leaf, we pop a node from the top of the stack. Traversal is complete when no nodes remain on the stack. We also skip faces with normals facing away from our light source to eliminate intersections on sides opposite from the light, and to save performance.

There is a practical limitation with regards to using the geometry shader. The buffer we output vertices into is limited in size, so depending on the amount of data we want to stream out—position, in our case—we can only

output a limited number of vertices. With a single position tuple of four floats, we can currently output 256 vertices, which means 128 contour edges for each triangle. This limitation could require us to increase polygon count on shadow receiving geometry to prevent the GPU from discarding contour edges. To lower the impact of this limitation, we quantize the positions and discard contour segments which have no length after quantization. We achieve this by defining a grid in 3D world space with a small enough resolution to be unnoticeable—usually around 1/10 of the contour stroke width—and then each contour segment point is snapped onto the nearest grid point. If the length of the contour segment is zero after quantization, we do not render it.

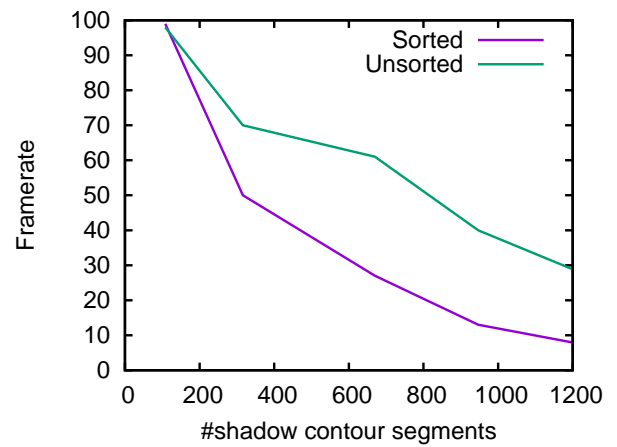


Figure 5: Overall performance of shadow outline rendering with our GPU bottom-up bounding volume hierarchy construction without sorting, and with the CPU top-down solution with sorting, on an NVidia 970 GTX and an i7 3770K. The shadow receiver had 2182 faces.

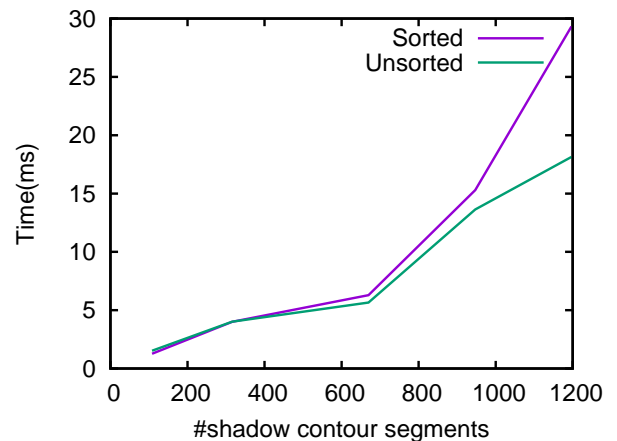


Figure 6: Tree traversal times for trees constructed with our GPU bottom-up approach without sorting, and with the CPU top-down solution with sorting. Surprisingly, the unsorted tree often outperformed the sorted one.

5 Evaluation of unsorted bounding volume hierarchy performance

For the purpose of comparison, we needed to implement a bounding volume hierarchy building algorithm that does not exploit pre-existing locality, but sorts the silhouette segments before subdividing them. Numerous such algorithms exist, with different schemes used for partitioning objects (Sulaiman [19] offers an overview). These may have significant differences in performance. We opted for a simple one that is close to our method in that it builds a balanced tree, and does not use expensive cost estimates to improve partitioning: the object median method. This is a top-down method, sorting the objects according to a coordinate axis, and splitting them so that the two partitions have the same number of elements. We implemented this method on the CPU. There is little doubt that comparison with a more sophisticated bounding volume hierarchy construction scheme (e.g. with surface area heuristics) could provide better traversal statistics—at least in theory. In practice, traversing unbalanced trees would require a larger local stack and a less direct tree representation, both of which would impact GPU performance, which is why we focused on the most practical object median method.

We measured our bottom-up GPU implementation against the top-down CPU algorithm. Not surprisingly, the overall frame rate was much more favorable with the GPU algorithm, especially as the face count of the shadow volume increased (Figure 5). This is easy to explain, as CPU sorting made the application CPU-limited, and bounding volume hierarchy construction stalled the rendering process. This could be somewhat mitigated by parallel sorting on the GPU—making the solution much more complex and difficult to implement—but tree construction times will always remain relatively high.



Figure 7: Eagle shadow caster test scene with cast shadow contours on double ellipsoid receiver.

More interestingly, we measured the tree traversal times for the two methods. Test scenes are shown in Figures 7, 8, 9, 10, 11. Table 1 shows scene characteristics and

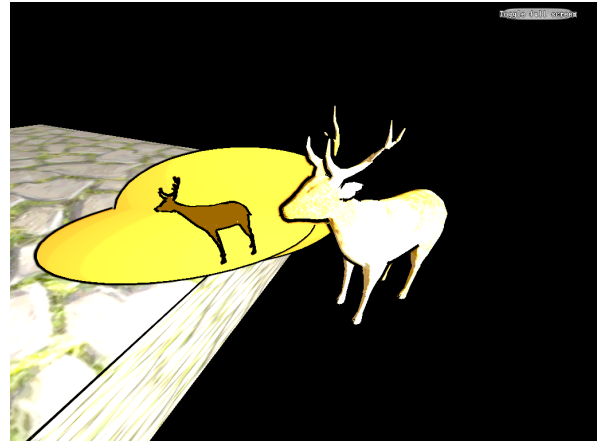


Figure 8: Deer shadow caster test scene with cast shadow contours on double ellipsoid receiver.

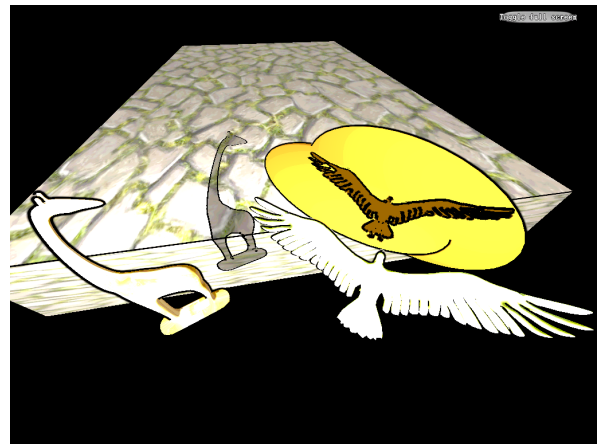


Figure 9: Eagle and giraffe shadow casters test scene with cast shadow contours on double ellipsoid receiver.

times for tree traversal with the sorted and unsorted trees. We expected the sorted tree to perform better, but of course not so much as to validate construction overhead. Surprisingly, we found that more often than not the unsorted tree performed even better than the sorted one. Thus, we conclude that not sorting the objects is perfectly sound in this application.

6 Conclusion

We have shown that during shadow contour rendering, it is unnecessary to include an expensive sorting step, when building an acceleration hierarchy over the contour edges. We have presented an algorithm for rendering cast shadow contours exploiting this fact. Comparison with the object median split scheme using sorting revealed that traversal times remain similar, while tree construction times are much lower, allowing for real-time operation for scenes of about thirty thousand triangles.

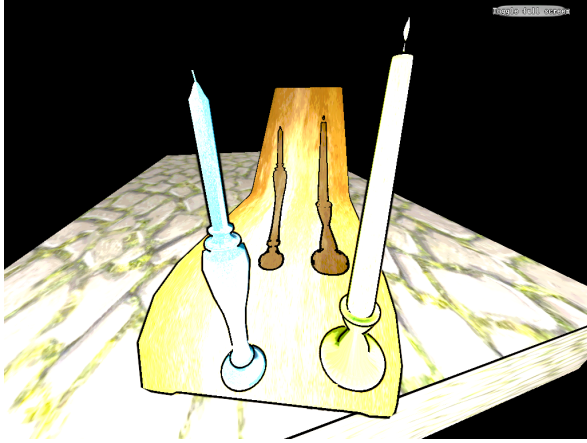


Figure 10: Candle shadow casters test scene with cast shadow contours on deck chair.

Table 1: Tree traversal times (in ms) for the test scenes. Shadow caster polygon counts and the number of shadow contour edges are given for every scene.

Test scene	polys	lines	sorted	unsorted
Eagle	21328	707	18.66	19.13
Deer	28434	797	30.25	25.23
G&E	25930	954	55.23	50.69
Candles	21594	512	28.3	19.24
Heart	20468	293	5.11	6.93

7 Future work

Stylization of the shadow contour strokes should be improved to mimic actual artistic work.

Cast shadow outlines should integrate smoothly with other kinds of strokes in artistic rendering, offering a special tool for emphasizing shadows. Therefore, we need to integrate our method with NPR techniques other than outline rendering. Filling the shadows with hatching is the most obvious task. In architectural rendering, researching ways to render outlined shadows with precise hatching strokes may be interesting.

We could further compare the performance of the unsorted bounding volume hierarchy against more sophisticated methods, like the linear bounding volume hierarchy [12, 11]. This could provide some additional insight into the locality requirements characteristics. However, if not sorting works adequately, it is hard to envision a scenario where devoting resources to build a better bounding volume hierarchy could pay off—at least in a dynamic environment where shadow contours change in every frame.

8 Acknowledgements

This work has been supported by OTKA PD-104710.

References

- [1] Pierre B  nard, Aaron Hertzmann, and Michael Kass. Computing smooth surface contours with accurate topology. *ACM Transactions on Graphics (TOG)*, 33(2):19, 2014.
- [2] Stefan Brabec and Hans-Peter Seidel. Shadow volumes on programmable graphics hardware. In *Computer Graphics Forum*, volume 22, pages 433–440. Wiley Online Library, 2003.
- [3] D. DeCarlo, A. Finkelstein, and S. Rusinkiewicz. Interactive rendering of suggestive contours with temporal coherence. In *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*, pages 15–145. ACM, 2004.
- [4] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 848–855. ACM, 2003.
- [5] Elmar Eisemann, Holger Winnem  ller, John C Hart, and David Salesin. Stylized vector art from 3d models with region support. In *Computer Graphics Forum*, volume 27, pages 1199–1207. Wiley Online Library, 2008.
- [6] Paul Haeberli. Paint by numbers: Abstract image representations. In *ACM SIGGRAPH Computer Graphics*, volume 24, pages 207–214. ACM, 1990.
- [7] Mark Harris et al. Optimizing parallel reduction in CUDA. *NVIDIA Developer Technology*, 2(4), 2007.
- [8] Vlastimil Havran. *Heuristic ray shooting algorithms*. PhD thesis, Citeseer, 2000.
- [9] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526. ACM Press/Addison-Wesley Publishing Co., 2000.
- [10] T. Judd, F. Durand, and E. Adelson. Apparent ridges for line drawing. In *ACM Transactions on Graphics (TOG)*, volume 26, pages 19–19. ACM, 2007.
- [11] Tero Karras. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics*, pages 33–37. Eurographics Association, 2012.
- [12] Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David Luebke, and Dinesh Manocha. Fast bvh construction on gpus. In *Computer Graphics Forum*, volume 28, pages 375–384. Wiley Online Library, 2009.

- [13] L. Markosian and J.F. Adviser-Hughes. *Art-based modeling and rendering*. Brown University, 2000.
- [14] L. Markosian, M.A. Kowalski, D. Goldstein, S.J. Trychin, J.F. Hughes, and L.D. Bourdev. Real-time nonphotorealistic rendering. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 415–420. ACM Press/Addison-Wesley Publishing Co., 1997.
- [15] M. Nienhaus and J. Doellner. Edge-enhancement—an algorithm for real-time non-photorealistic rendering. *Journal of WSCG*, 11(2), 2003.
- [16] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 581–581. ACM, 2001.
- [17] J. Shin. A stylised cartoon renderer for toon shading of 3d character models. Master’s thesis, University of Canterbury, UK, 2006.
- [18] Thomas Strothotte and Stefan Schlechtweg. *Non-photorealistic computer graphics: modeling, rendering, and animation*. Elsevier, 2002.
- [19] Hamzah Asyrani Sulaiman. *Bounding Volume Hierarchies for Collision Detection*. INTECH, 2012.

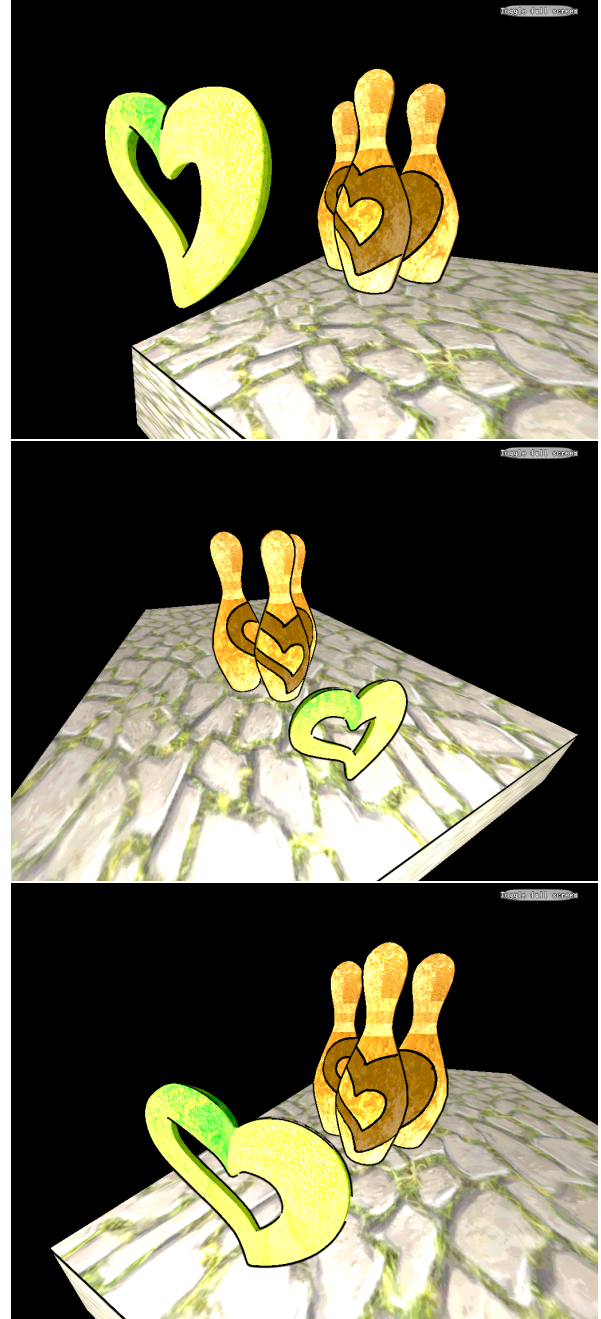


Figure 11: Heart shadow caster test scene with cast shadow contours on three bowling pin receivers.

Configurable Rendering Effects For Mobile Molecule Visualization

Lukas Prost*

Supervised by: Reinhold Preiner†

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Vienna / Austria

Abstract

Due to their omnipresence and ease of use, smart phones are getting more and more utilized as educational instruments for different subjects, for example, visualizing molecules in a chemistry class. In domain-specific mobile visualization applications, the choice of the ideal visualization technique of molecules can vary based on the background and age of the target group, and mostly depends on the choice of a graphical designer. Designers, however, rarely have sufficient programming skills and require an engineer even for the slightest adjustment in the required visual appearance. In this paper we present a configuration system for rendering effects implemented in Unity3D, that allows to define the visual appearance of a molecule in a JSON file without the need of programming knowledge. We discuss the technical realization of different rendering effects on a mobile platform, and demonstrate our system and its versatility on a commercial chemistry visualization app, creating different visual styles for molecule renderings that are appealing to students as well as scientists and advertisement.

Keywords: Molecule Shading, Mobile, Unity3D

1 Introduction

Mobile molecule visualization can be useful for many different groups e.g. scientists and students. Yet, every target group has their own requirements due to their different purposes. Often it is up to a designer to create a visual appearance that best meets those requirements. Designers, however, rarely have the technical skills to realize their design in a graphical rendering framework on their own. An engineer has to build the design and alternate it every time the slightest adjustment has to be made. As a result, there are always at least two people required to maintain an application's update life cycle.

In this paper, we present a mobile molecule visualization implemented in Unity3D, that allows to easily modify

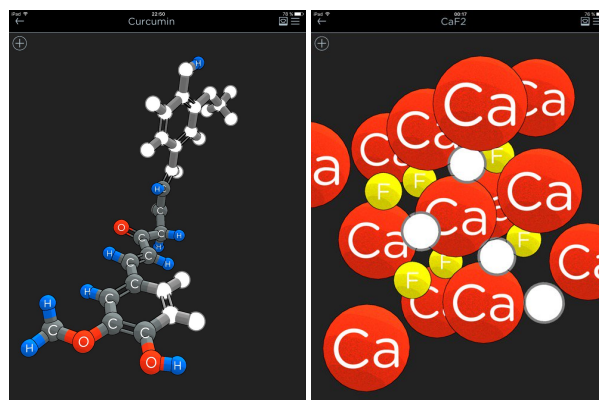


Figure 1: Exemplary screenshots of a commercial chemistry visualization app.

the visual appearance with the help of a JSON configuration file. Designers can change the rendering by setting parameters in these files e.g. which shaders to use or where lights should be placed with no required programming skill whatsoever. In the remaining paper, we will demonstrate how to apply high quality rendering effects like screen space ambient occlusion, depth of field and comic shading/outline rendering in a mobile commercial chemistry visualization app (see Figure 1) and how to make them configurable using JSON files.

The rest of this paper is structured as follows: Section 2 reviews some related mobile molecule visualization apps, gives background in Unity and JSON, and discusses the related work on the rendering effects used by our system. Section 3 shows the JSON file that is used for appearance parametrization and how the textual information is used for molecule rendering. In Section 4 different rendering techniques are explained in more detail that are used to visualize molecules. Finally, in Section 5 we present some results and show different rendering styles that can be achieved by our system.

*lukas.prost@tuwien.ac.at

†preiner@cg.tuwien.ac.at

2 Background and Related Work

2.1 Molecule Visualization Systems

NDKmol [3] is an open source visualization app for smart phones. It supports many different visualization techniques like bond and ribbon diagrams and has direct access to the *RCSP Protein Data Bank* (RCSP PDB). The rendering, however, looks very plain and scientific. *RCSB PDB Mobile* [4] is a molecule visualization app that is officially provided by RCSP PDB. Because it is based on *NDKmol*, it has the same visual appearance. *Molecules* [2] is an alternative open source app that can load molecules directly from RCSP PDB. Unfortunately, it is only available for iOS. *Atomdroid* [10] is an app for Android with a lot of functionality besides visualization. Among other things it allows to build molecules and do trajectory analysis. The last update, however, was 2012.

2.2 Unity3D

Unity3D is a free to use game engine which supports deployment for many different platforms. A *Unity3D* project consists of different *scenes* which be thought of as levels in a game. They contain all elements, e.g. scripts and models, and information, e.g. level architecture, required to run the scene as a program. All objects appearing in a scene are *Game Objects* (GO). A GO is the core element of *Unity3D* and can be thought of a container for components. The properties of a GO are defined by the components that are attached to it. These components can e.g. be a transform component, defining the GO's position, orientation and scale, or the camera component that enables the GO to render the scene. Later in the paper we will show how to manipulate a *Unity*-based system to define a GO's visual appearance using JSON config files.

2.3 JSON

The *JavaScript Object Notation* (JSON) [1] is an up-to-date, easy to read file format for transferring data and is mainly used in web development. It is lightweight and widely supported. JSON is used in the presented system to store visualization meta data. XML would have been the other option, yet it was dismissed because it is verbose and therefore not as legible as JSON. Data is stored as name/value pairs. While the *name* is always a string, the *value* store different types of data, ranging from simple types (number, string) to complex types like arrays or objects. An array can contain values, arrays and objects. Objects can store name value pairs. For more details about JSON and its syntax, see the JSON specification [1].

2.4 Realtime Rendering Effects

Comic Shading One technique to shade objects with a flat cartoon look is *hard shading* presented by Lake et al.

[15]. The shading is done by a texture lookup based on the dot product between the normal vector and the light direction, but without interpolation resulting in a shading with few solid colors. Mitchell et al. [17] create a cartoon look without hard shading by using a 1D lookup texture and a modified Lambert lighting model. Vanderhaeghe et al. [20] present an approach for creating stylized renderings (including toon shading) dynamically by composing procedural primitives. A primitive describes a shading behavior and its parameters can be defined dynamically.

Outline Rendering Akenine-Möller, Haines and Hoffman [5, p.512] describe a heuristic method, that marks surface points as part of an object's silhouette if the dot product between the view and the normal vector is close to zero. Isenberg et al. [13] mark edges that share a front facing and a back facing polygon relative to the viewer as silhouette edges. Another approach presented by Akenine-Möller et al. [5] is the *halo* or *shell* method. An object is rendered by two passes, where the first pass renders the front faces of an object and the second pass renders its enlarged back faces. Kolivand and Sunar [14] detect silhouette edges for shadow volumes by sending a ray for each edge from the light source to one along this edge translated end vertex of the edge. If the ray does not intersect with any face of the object, the processed edge is a silhouette.

Ambient Occlusion The concept of ambient occlusion (AO) and its benefits are described by Landis [16]. An implementation is given by Pharr and Green [19]. A technique that enables dynamic real time computation is *screen space ambient occlusion* (SSAO) which first was presented by Mittring [18]. It simulates occlusion from nearby surfaces by using the depth buffer to approximately reconstruct local geometry. To do so, random samples are placed around each fragment's view space position which is then compared against the depth of the surrounding geometry using simple depth buffer lookups. The more samples are covered by the surrounding geometry, the more the fragment is occluded. Filion and McNaughton [11] describe an improved version of Mittring [18] by aligning the samples on a hemisphere around the surface normal reducing self occlusion dramatically.

Depth of Field Physically correct *Depth of Field* (DoF) rendering is presented by Cook, Porter and Carpenter [7], who simulate light distortion by ray tracing through a virtual lens. Haeberli and Akeley [12] render the scene from several slightly different view points and use the accumulation buffer to blend the renderings together into a final image. Demers [8] simulates DoF in screen space by separating the scene into layers based on the depth buffer. After blurring these layers based on their depth, the scene is composed back together resulting in a visual appealing DoF effect. Filion and McNaughton [11] present an implementation of this approach that uses five layers.

3 Appearance Parametrization

In this Section we will demonstrate how we parameterize the visual appearance of a molecule by a simple JSON configuration file, and how it is integrated in a Unity3D application to control its scene rendering.

Listing 1: Template for the JSON configuration file

```
{
  "camera" : {
    "orthographic" : <boolean> ,
    "bgcolor" : [ <integer> , <integer> , <integer> ]
  }

  "mapping" : {
    <<string> : <string>>*
    "post_effects" : [ <string>* ] ,
  }

  "shaders" : [
    <{
      "name" : <string> ,
      "shader" : <string> ,
      "properties" : { ... } ,
    }>* ]

  "lights" : [
    <{
      "type" : <string> ,
      "position" : [ <integer> , <integer> , <integer> ] ,
      "color" : [ <integer> , <integer> , <integer> ]

      "intensity" : <float> ,
      "shadow" : <string> ,
      "strength" : <float> ,
      "movable" : <boolean> ,
    }>* ]
}
```

3.1 JSON Config File

The whole visual appearance of a scene is stored in a JSON config file (JCF). A template of its structure and syntax is shown in Listing 1. Our configuration file has four main name/value pairs:

- **Camera** has an object that stores a background color and a boolean defining whether the projection is orthographic or perspective.
- **Shaders** stores an array of shader objects. A shader object contains a name for the shader (*name*) and the name of the used shader (*shader*). The first one functions as a reference/id which is valid inside the current JCF, whereas later one is the actual name of the used shader inside the application. Moreover, a shader object contains a property object storing parameters for each individual shader. The available shaders are the Unity3D default shaders as well as custom shaders described in Section 4.
- **Mapping** has an object with name/value pairs where the name refers to an actual object in the scene and the value is the reference to a shader object in *Shaders*. The available scene objects depend on the application. *post_effects* stores an array of post processing shader references that will be applied in the order of the array.

- **Lights** stores an array that contains light objects. A light object contains all properties of a light source e.g. its type (point or directional), its position and what kind of shadow it casts (none, soft or hard).

3.2 Integration in Unity3D

The system that applies the JCF described in Section 3.1 to the Unity3D Scene consists of the three independent modules: a *ShaderProvider*, a *CameraProvider* and a *LightingProvider*. The interaction of these modules with the core entities of an Unity application is illustrated in Figure 2.

The *ShaderProvider* works with the data stored in *Shaders* and *Mapping*. Every Game Object (GO) that will be rendered requests a shader from this module, either by specifying its defined type or by asking for a specific shader reference. The *ShaderProvider* first checks if the requested shader is available. If this is the case, it loads the shader from the system, sets the properties stored in the corresponding JCF shader object and then applies it to the requesting GO. Besides providing shaders for GOs, it can also provide post processing shaders for the camera.

The *CameraProvider* reads the parameters specified by *Camera* in the JCF and modifies the parameters of the Unity camera component accordingly. In a similar way, the *LightingProvider* module processes the data given by *Lights*. For each JCF light object a new light is placed in the Unity scene.

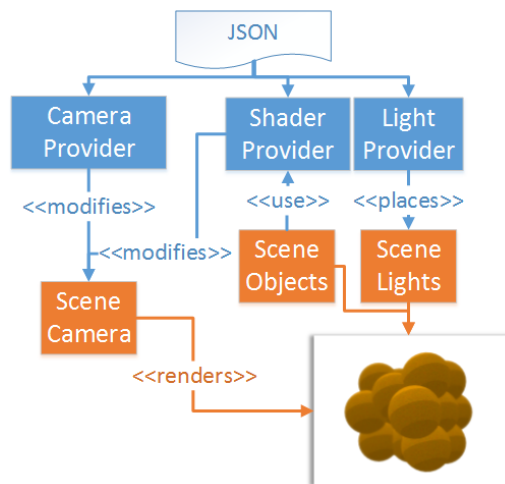


Figure 2: Relation of the Config Loader Modules (blue) to the Unity3D Scene Elements (orange).

4 Molecule Rendering Effects

In this section, we describe three major rendering effects that are supported in our molecule visualization app, and discuss their realization in a mobile real-time rendering framework.

4.1 Comic Shading and Outlining

To support a non-photo realistic, stylized look, a highly customizable shader was implemented to provide a high degree of visual variety to the designer. The shader does actual hard shading and allows outline rendering in object space. For the latter, both the dot product method and the hull method of Akenine-Möller et al. [5] were implemented. Both provide a good trade-off between performance and results depending on the model.

Comic Shading To achieve an efficient comic style that is easy to configure by a designer, we implemented hard shading without texture lookup. Using a texture-based approach, the designer would have to provide a texture for every single GO, which would not be possible just based on a modification in the JCF. The basis for the brightness calculation is a modified Lambert term, shown in Equation 1 [17]. The original Lambert reflection model is extended by scale constant α , a bias β and an exponent γ .

$$(\alpha(\hat{n} \cdot \hat{l}) + \beta)^\gamma \quad (1)$$

These constants can also be configured by the user in the JCF and allow to modify the distribution of the hard shading borders. To achieve a hard shading look, Equation 1 is discretized by clamping, resulting in the final shading formulation:

$$\frac{(\alpha + \beta)^\gamma}{s} \left\lceil \frac{s(\alpha(\hat{n} \cdot \hat{l}) + \beta)^\gamma}{(\alpha + \beta)^\gamma} \right\rceil \quad (2)$$

The subdivision parameter s defines the number of gradients/shading borders and is again configurable by the JCF.

Figure 3 shows examples of hard shaded atoms with different subdivision parameters s .



Figure 3: Comic shaded spheres ($\alpha = 0.5$, $\beta = 0.5$ and $\gamma = 2$) with $s = \{2, 4, 8\}$ from left to right.

Outline Rendering For outline rendering [5], we need to calculate the dot product between the view vector and normal vector at each pixel. If the result lies below a specific user defined threshold (typically values between 0.25 and 0.5), the pixel gets rendered in a border color. Both the threshold and the border color are parameters that can be set by the user in the JCF. This algorithm is very efficient because it adds only one additional dot product and comparison evaluation to the pixel color. An outline of a sphere rendered with this method can be seen in the upper row of Figure 4.

The second outline rendering technique available in our system is the hull method [5]. This method creates an outline by first enlarging a model and then rendering its backfaces. The enlarging is done by a vertex translation along the vertex normal. To do so, the vertex and its corresponding normal vector need to be transformed into view space. Then, the vertex is translated along the x and the y coordinate of the normal vector. The length of the translation defines the hull size and can be modified by the user. This value depends on the size of the objects because it is happening in view space. For the atoms, it is normally between 0.005 and 0.03. After the translation, the front faces are culled and the back faces are rendered with the defined border color. The results can be seen in the lower row of Figure 4.

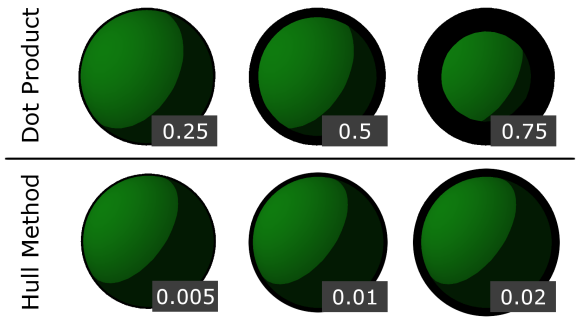


Figure 4: Outlines rendered with different methods. The numbers show the dot product threshold (upper row) and the hull size (lower row).

4.2 Screen-Space Ambient Occlusion

SSAO is a fast screen space effect suitable for mobile real time applications, that can greatly enhance the visual quality of the resulting images. It uses the depth buffer as a discretized representation of the scene, based on which it estimates the ambient occlusion for each pixel in screen space.

To this end, we first need to calculate the view space position of the pixel. Samples are then placed around this point by adding a set of predefined offset vectors to their view-space position. Each new sample point is then projected back to screen space where their z-values are compared to the stored depth at their target screen space position (similar to shadow mapping). Each neighboring sample with a z-value larger than the stored depth increases the ambient occlusion value of the center pixel. To keep the ambient occlusion value independent of the number N of used samples, it is normalized by N . Because this procedure is repeated for every pixel with the same samples, this process can also be seen as a convolution of the discretized scene with a sample kernel.

For high-quality AO effects, Christensen [6] recommends at least 256 samples. Yet, such a high sample count is not feasible for mobile real-time rendering. Therefore,

Engel [9] presents an SSAO implementation that achieves a moderate result with only 16 samples. Without further regard, such a small sample count would lead to a visible pattern of the sample kernel. For this reason, this technique randomly rotates the sample kernel for each pixel using a random rotation matrix. Filion and McNaughton [11] suggest using a random vector provided by a noise texture instead of random rotation matrices. Each offset vector of the sample kernel is reflected by this offset vector resulting in a semi randomization of the kernel per pixel. Since it is the most efficient SSAO variant, we choose this random-vector-based technique for our mobile real-time rendering system.

The randomization of the sample kernel reduces kernel artifacts, but results in a coarse SSAO image. For this reason, the buffer that stores the SSAO values are blurred with a kernel that should be small enough (4x4 pixels) to preserve borders as good as possible.

Finally, each pixel of the rendered scene is darkened by its corresponding value in the SSAO texture. Because these values lie in the interval $[0, 1]$, they can simply be multiplied to the pixels color. The effect of SSAO on the visual expressiveness of a scene is demonstrated in Figure 5.

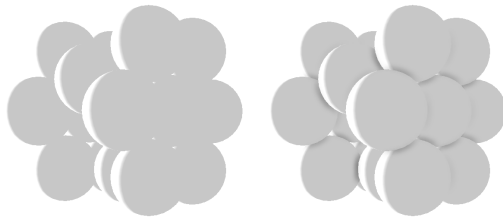


Figure 5: Scene rendered without SSAO (left) and with SSAO (right)

4.3 Depth of Field

Ray-tracing Depth of Field effects, as suggested by Cook et al. [7], or accumulation of multiple render passes as proposed by Haeberli and Akeley [12] would be too costly for a real-time performance on mobile platforms. Therefore, DoF is applied as a post processing effect based on the method of Filion and McNaughton [11].

Based on the values in the depth buffer, the screen-space representation of the scene is divided into five depth layers, as shown in Figure 6. Each layer is defined by a range that can be set by the designer in the JCF. A texel is assigned to a layer if its depth value falls into the layers depth range. The designer can define the five layers by setting the four border depth values $TR[0]$, $TR[1]$, $TR[2]$ and $TR[3]$ between them. The relation between the TR values have to be $TR[0] \leq TR[1] \leq TR[2] \leq TR[3]$.

The DoF effect is applied in four steps. The first step separates the scene based on the given ranges into layer's. The near and the far layer are stored in two separate frame

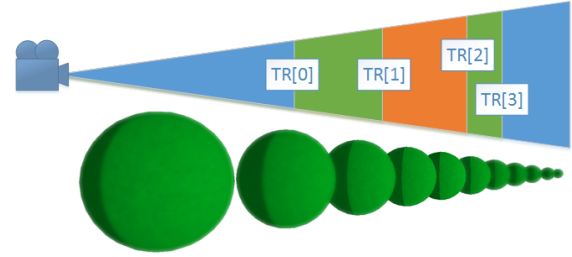


Figure 6: DoF layers (l.t.r): near, transition near to focus, focus, transition focus to far, far.

buffers. If a texel's depth is an element of e.g. the near layer, its color and depth are rendered into the near layer frame buffer. Otherwise, the texel is rendered with the camera's clear color. The focus layer does not get stored in a separate frame buffer. Instead, the unprocessed input frame buffer is used. After the pixels are assigned, the near and the far layer frame buffers are blurred with a separated Gaussian. Finally, the layers are composed together based on the same ranges as were used for their separation. If a depth value is in the range of a transition layer, the resulting texel is determined by interpolating between the texels of the two neighbor layers.

5 Results

Our configurable rendering system can be easily used by people without any programming skills. An engineer has to implement shaders in our system only once. After he made them accessible for the configuration loader, the designer can apply and modify them as he wishes. In the following, we will give an example of four different designs that can be produced in our system, and show their performance on several mobile devices.

5.1 Visual Designs

Table 1 presents four different looks that were produced in our system only by manipulating the JSON config file. The table shows an outline of these config files, and illustrates the resulting visual appearance on four different molecules. The shader referred to as *basic* is the standard shader provided by Unity3D. The shader's *dofPost* and *ssaoPost* denote the post-processing DoF and SSAO shaders, respectively. Finally, *toon* addresses the object space comic shader.

Education The visual appearance for students was created with a simple and flat design in mind. Flat and tactile looking interfaces are currently modern and widely used. Moreover, the design tries to support a visual gamification to be appealing for this target group. To this end, comic shading and outline rendering was used.

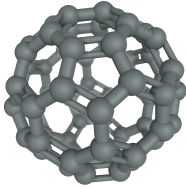
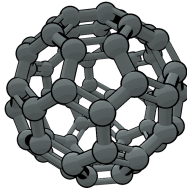
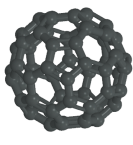
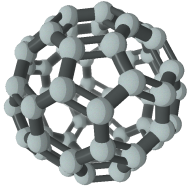
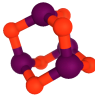
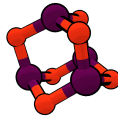

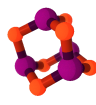
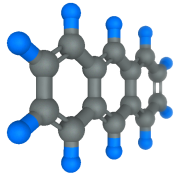
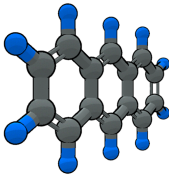
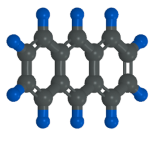
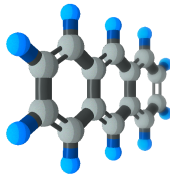
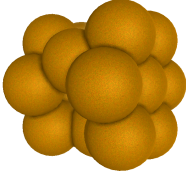
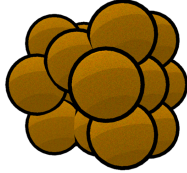


	Advertisement	Education	Scientific	Future
Fulleren(C_{60})				
Arsenik(As_2O_3)				
Anthracen($C_{14}H_{10}$)				
Magnesium(Mg)				
	<pre> "camera" : { <...> }, "mapping" : { "atom" : "advert", "connector" : "advert", "post_effects" : ["dof", "ssao"] }, "shaders" : [{ "name" : "advert", "shader" : "basic", "properties" : { "smoothness" : 0.15, "metallic" : 0.3 } }, { "name" : "dof", "shader" : "dofPost", "properties" : { "layers" : [[0,0,0.7,0.9]] }, { "name" : "ssao", "shader" : "ssaoPost", "properties" : { } }], "lights" : [<...>] </pre>	<pre> "camera" : { <...> }, "mapping" : { "atom" : "school", "connector" : "school", "post_effects" : [] }, "shaders" : [{ "name" : "school", "shader" : "toon", "properties" : { "color" : [0,0,0], "hull_size" : 0.45, "outline_bias" : 0.0, "scale" : 0.5, "bias" : 0.65, "exponent" : 1, "steps" : 4 } }, { "name" : "toon", "shader" : "toon", "properties" : { "color" : [0,0,0], "hull_size" : 0.45, "outline_bias" : 0.0, "scale" : 0.5, "bias" : 0.65, "exponent" : 1, "steps" : 4 } }], "lights" : [<...>] </pre>	<pre> "camera" : { "orthographic" : " true", ... }, "mapping" : { "atom" : "science", "connector" : "science", "post_effects" : [] }, "shaders" : [{ "name" : "science", "shader" : "basic", "properties" : { "smoothness" : 0.5, "metallic" : 0.2 } }, { "name" : "basic", "shader" : "basic", "properties" : { "smoothness" : 0.5, "metallic" : 0.2 } }], "lights" : [<...>] </pre>	<pre> "camera" : { <...> }, "mapping" : { "atom" : "future", "connector" : "basic", "post_effects" : ["dof"] }, "shaders" : [{ "name" : "basic", "shader" : "basic", "properties" : { "smoothness" : 0.5, "metallic" : 0.2 } }, { "name" : "future", "shader" : "toon", "properties" : { "color" : [0,0,0], "hull_size" : 0.0, "outline_bias" : 0.0, "scale" : 0.5, "bias" : 0.65, "exponent" : 1, "steps" : 4 } }, { "name" : "dof", "shader" : "dofPost", "properties" : { "layers" : [[0,0,0.7,0.9]] } }], "lights" : [<...>] </pre>

Table 1: Different molecules rendered with different styles and summarized JCF defining the visualizations.

Advertisement To sell a CG-related product, its visual output has to look as stunning as possible. Moreover, the quality of the renderings used in advertisement directly reflect the public image of the company producing the software. Therefore, SSAO as well as DoF are applied to the rendering. By doing so, the molecule looks much more plastic and "realistic".

Scientific For science, the visual appearance should support the understanding of the structure of a molecule. The rendering should avoid any additional effects that clutter the image of a molecule. For this reason, standard shading was used without any effects and the camera uses an orthographic projection. This aims at supporting the understanding of the structure of a molecule.

Future This look was created to demonstrate how object types can be shaded differently. The visibility of the atoms gets significantly enhanced by applying a bright comic shader to the atoms and a dark basic shader to the connectors,.

5.2 Performance

Performance data for the advertisement, education and scientific look is shown in Figure 7 for a smart phone and a tablet. The used smart phone was a *OnePlus One* with a resolution of 1920x1080. The tablet data was gathered from a *Nvidia SHIELD TABLET K1* with a resolution of 1920x1200. The performance data was gathered by rendering each example for a short period of time. The FPS value was taken in short intervals. The final results for each example are the average over the collected FPS values.

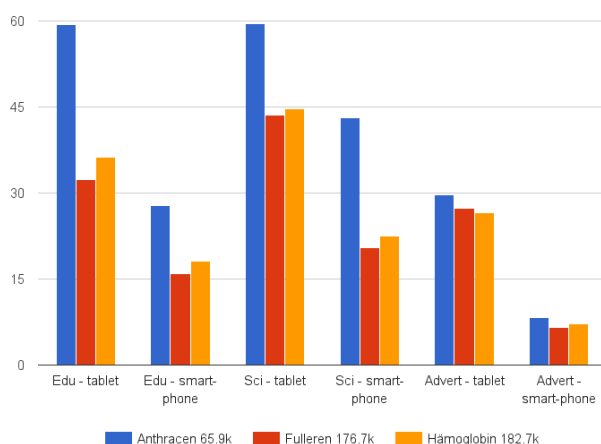


Figure 7: Performance data. Polygon number next to the name.

The performance values indicate an expected dependence on the polygon count of the model. The *Education* and *Science* styles run at acceptable rates even for big models. For all our models, the *Advertisement* style is the

computationally most demanding one, with under 10 FPS on smart-phones. This can be attributed to the usage of the Depth of Field effect. This performance is acceptable, as this style is mostly meant for creating still shots used in advertisement.

6 Conclusion and Future Work

We have presented a system that allows to define the visual appearance of rendered molecules only by modifying parameters in a JSON configuration file. The syntax and the structure of this file is simple to read and easy to understand, such that even people without deeper knowledge about rendering and shaders can change the visual appearance of a scene easily. Since the shown system reads the JSON file during run-time, the rendering style can be changed fast without the need of rebuilding the application. This supports prototyping with fast and easy adjustments. A GUI with live feedback would simplify the process even more. This is left open for future work.

References

- [1] Json specification. <http://www.json.org/>. Accessed: 13-02-2016.
- [2] Molecules. <http://www.sunsetlakesoftware.com/molecules>. Accessed: 13-02-2016.
- [3] Ndkmol. <http://webglmol.osdn.jp/>. Accessed: 13-02-2016.
- [4] Rcsb pdb mobile. <http://www.rcsb.org/pdb/static.do?p=mobile/RCSBapp.html>. Accessed: 13-02-2016.
- [5] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-time rendering*. CRC Press, 2008.
- [6] P. H Christensen. Global illumination and all that. *SIGGRAPH 2003 course notes*, 9, 2003.
- [7] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *SIGGRAPH Comput. Graph.*, 18(3), 1984.
- [8] J. Demers. Depth of field: A survey of techniques. *GPU Gems*, 1(375), 2004.
- [9] W. Engel. Shaderx7. *Charles River Media*, 2009.
- [10] J. Feldt, R. A Mata, and J. M Dieterich. Atomdroid: a computational chemistry tool for mobile platforms. *J. of chem. inf. and modeling*, 52(4), 2012.
- [11] D. Fillion and R. McNaughton. Effects & techniques. In *ACM SIGGRAPH 2008 Games*, SIGGRAPH '08. ACM, 2008.

- [12] P. Haeberli and K. Akeley. The accumulation buffer: Hardware support for high-quality rendering. In *Proc. of the 17th Annu. CC on CG and Interactive Techniques*. ACM, 1990.
- [13] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte. A developer's guide to silhouette algorithms for polygonal models. *CG and AP, IEEE*, 23(4), 2003.
- [14] H. Kolivand and M. S. b. Sunar. New silhouette detection algorithm to create real-time volume shadow. In *DMDCM, 2011 Workshop on*, 2011.
- [15] A. Lake, C. Marshall, M. Harris, and M. Blackstein. Stylized rendering techniques for scalable real-time 3d animation. In *Proc. of the 1st Int. Symp. on Non-photorealistic animation and rendering*. ACM, 2000.
- [16] H. Landis. Production-ready global illumination. *Siggraph course notes*, 16(2002), 2002.
- [17] J. Mitchell, M. Francke, and D. Eng. Illustrative rendering in team fortress 2. In *Proc. of the 5th Int. Symp. on Non-photorealistic animation and rendering*. ACM, 2007.
- [18] M. Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*. ACM, 2007.
- [19] M. Pharr and S. Green. Ambient occlusion. *GPU Gems*, 1, 2004.
- [20] D. Vanderhaeghe, R. Vergne, P. Barla, and W. Baxter. Dynamic stylized shading primitives. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Non-Photorealistic Animation and Rendering*, NPAR '11. ACM, 2011.

Visualization

Sonoco: Interactive Visual Comparison of Filtering Operations on Time-Dependent Medical Imaging Data

Deniz Gezgin*

Sergej Stoppel†

Supervised by: Stefan Bruckner‡

University of Bergen / Norway

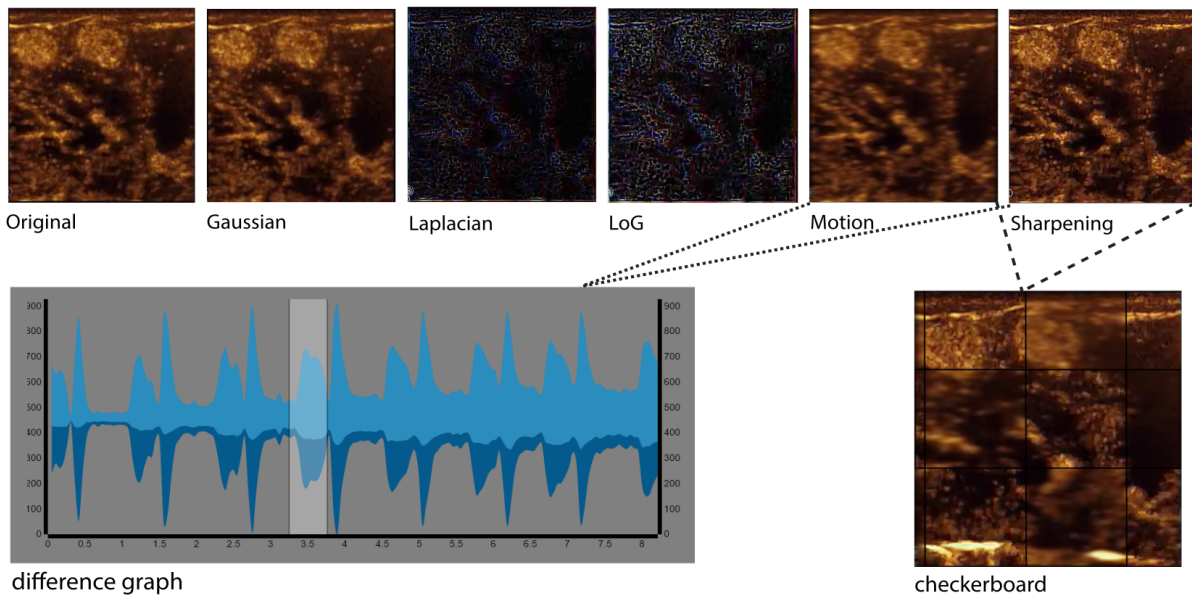


Figure 1: Sonoco Overview of a contrast-enhanced ultrasound video stream. The figure shows the different views. The used data is taken from Youtube. [7]

Abstract

Medical imaging often requires fine-tuned processing pipelines to reduce noise and to remove artifacts. Visual assessment by experts is a critical step in determining the performance of individual techniques or parameters, as goals are often ill-defined and trade-offs need to be considered. In this paper, we address the visual evaluation of filtering and other data enhancement operations on time-dependent medical imaging data. We present *Sonoco*, a web-based comparative visualization system which provides the user with flexible tools for comparing multiple filtering operations on one or several data streams. By providing a visual overview of temporal changes, our approach enables the quick identification of major differences which can then be explored in their spatial context.

*dgezgin90@gmail.com

†sergejsto@googlemail.com

‡stefan.bruckner@gmail.com

Keywords: Comparative Visualization, Visualization of time-dependent data, Web-based tool, Filtering, Image Processing

1 Introduction

Decision support based on visualization is very common in the medical area. In radiology, contrast-enhanced ultrasound has many applications like blood flow rate detection or organ edge delineation. The time-domain often needs to be preserved for different reasons, e.g., tracking the distribution of a contrast-agent. Image filtering is applied to highlight or smooth specific features which need to be inspected further and compared by the domain expert. Furthermore, it is often necessary to view multiple video streams simultaneously. The high spatial and temporal resolution of ultrasound makes detection of differences a

challenging task. A phenomenon called change blindness can occur, where visual changes in space and time are not noticed by the observer [4]. Time is also a very critical parameter. For example, the user has no time restriction to compare multiple images, but in video data, the displayed image is changing rapidly, depending on the frame rate. Gleicher et al. shows a variety of comparative visualization techniques [1]. However, these methods are not suitable for multiple temporal data. There is currently no application for image processing experts to compare filtered time-dependent ultrasound recordings.

In this paper we present an approach to combine comparative visualization and visualization of time-dependent image data in order to provide users with visual support for observing multiple video streams. *Sonoco*, our interactive system, provides a juxtaposed overview of the filter methods. Selected video streams are used to compute temporal differences. Observing these temporal changes helps to identify the impact of different methods and settings in a quantitative way. A superposition view supports the exploration of the spatial dimension in the same space.

Our system is based on a client/server architecture, which makes it easy to use and save resources on the client. It is directly linked to Matlab, a common environment for prototyping medical image processing filters.

2 Related Work

The subject of visual comparison has been extensively studied. In this paper we focus on the comparison of images or image series, i.e. videos, only. The visual comparison of images can be roughly classified in three categories: image variation measurement via image metrics, emphasis of differences in images and support for image comparison without difference computation methods.

Over the last few decades several **image metrics** have been developed, with different intentions. The image metrics can be classified into perceptual and non-perceptual metrics. We refer to Lin et al. [2] for an overview of the most common and important perceptual image metrics. In this paper we use non-perceptual mean squared metric to compute the differences between consecutive images.

Pixel based image metrics allow to find and **emphasize** local differences in images. A typical example of this approach is the work of Schmidt et al. [6], where the differences in large sets of images were emphasized. Many approaches use color to indicate differences between images, such as presented by Sahasrabudhe et al. [5], where the difference between the image and data-set was emphasized, or by Suomi et al. [8], where changes between MRI images were highlighted. However, these methods are sensitive to global intensity shifts, which occur in video data. We do not utilize direct emphasis of differences in this pa-

per.

In some cases image metrics are not suitable, in these cases no explicit support can be provided to the viewer. The viewer must rely on his or her memory to make the comparison. Several visualizations have been developed to aid the comparison and to reduce the memory effort of the user. Gleicher et al. [1] provides an overview of the most common techniques. In our approach we use a combination of superposition via checkerboard views and a side by side comparison of the video data.

3 Sonoco

Sonoco supports the analysis of time-dependent data. Manual filtering and comparison can take considerable effort and time. Synchronizing multiple video streams is challenging without the right tool. *Sonoco* helps the user to inspect the data by four unique views. The **thumbnail view** gives a first overview of the computed filters. Filter parameters and image properties can be customized. A **juxtaposition view** of selected videos can be set up via simple drag and drop operations. The **temporal difference graph** depicts the pixel changes per time-frame. Users can select regions in this view to loop over the corresponding frames. A **checkerboard view** combines the videos into one single view.

Our combination of the used methods results in a new and integrated visual analytics tool. Interaction between all views is shown in Figure 1. We have chosen this set of methods to fulfill the basic needs of an image processing expert and not overload the interface. Details about the views are described in the next subsections.

3.1 Thumbnail View

Filtered videos are shown on the left side of the interface as a thumbnail view. Filtering image data helps the user to get more information about the data. For example, smoothing helps to reduce noise, sharpening helps to detect edges. There is no right filter, but rather different methods for different problems. The user is very important in a filtering pipeline, since the filter results can not always be quantified. Fully automated calculations does not bring the best result. However, our tool offers basic filter function, implemented in Matlab (*Gaussian Filter, Laplace Filter, Laplace of Gaussian Filter, Motion Filter, Sharpening Filter*). Customization of the default settings is possible, if the parameters are not satisfying. The filtered streaming data is shown as a thumbnail view to compare the different methods right away. Selected filters can be dragged to a 2×1 or 2×2 grid in the center for juxtaposed comparison. Synchronized playback is possible for the dragged videos as well as the thumbnail videos.

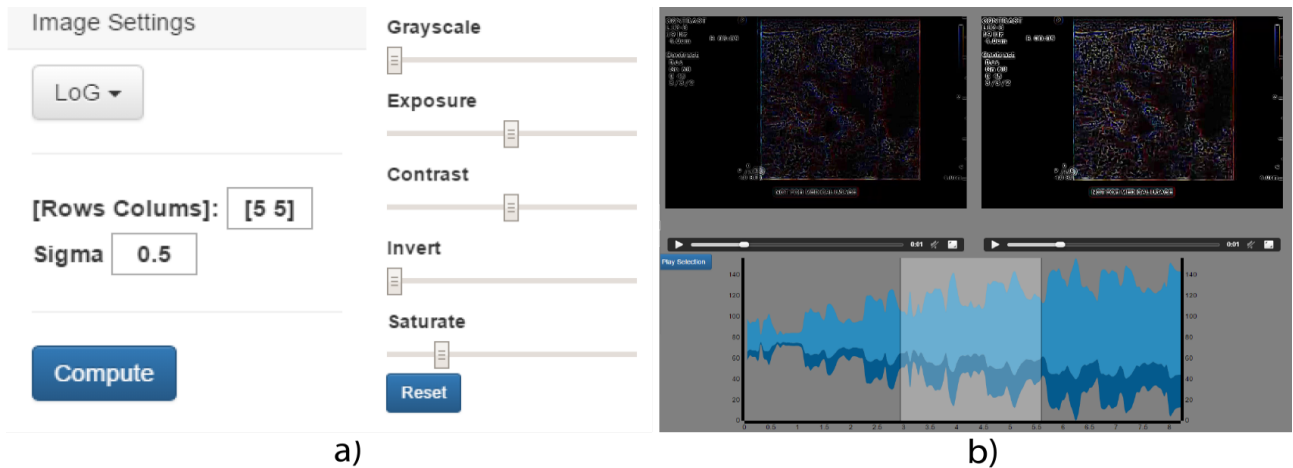


Figure 2: This figure shows the most important steps from section 5 Scenario. a) Original data from the user. b) Thumbnail view of filtered videos after upload. c) Mask for customizing filter parameters and change image settings. d) Juxtaposing two videos & difference graph

3.2 Temporal Difference Graph

Visual analytics supports the comparison process by showing differences over the time domain. The mean squared error (MSE) is a good non-perceptual metric to detect temporal changes. The difference of the greyscale value per pixel between two frames indicates a minor or major change. All these values are summarized per frame and displayed to the user. If two videos are selected with 10 frames each, the graph shows two line charts with 9 values per line. Our graph visualization technique is based on a streamgraph to compare the MSE between different filter methods. For example, a higher MSE of the same frame in another video indicates to a higher level of noise. The impact of different parameter settings can be measured by the MSE.

The shown graph allows the user to mark any region for further analysis. Selected regions are looped during playback to allow a detailed inspection of subtle temporal differences. Unimportant aspects of the data are skipped, which speeds up the analysis.

3.3 Checkerboard View

The checkerboard view superpositions multiple videos and automatically helps the human eye to detect differences. The same filter methods and just different parameter settings can lead to non-detectable differences for the observer. Furthermore, users can interactively change the size of the tiles and also move the checkerboard. This animated view shows the selected graph region mentioned above. The user can move the checkerboard and change size while the videos are still looped.

4 Architecture and Implementation

Figure 1 describes the architecture we used for the realization. The client can access our tool through the web and get all features. A web server provides the front-end. All image processing methods are implemented in Matlab. A Matlab function called *fspecial* creates predefined filter kernels which are used in our tool. *Matlab Jar Compiler* creates a Jar-File for the integration in Java Code. Every Matlab function is mapped to a Java method. Important user interactions trigger an *Ajax* request to the corresponding Java servlet. These servlets are provided by a *JBoss application server* and builds an interface between the browser and the Matlab computations.

Our tool does not require any client installation. Users can upload any known video format and all filtering processes are done on-the-fly. Even people with no MATLAB knowledge can filter their data and compare them.

5 Scenario

Sonoco offers many possibilities to modify and compare video streams. This sections describes a typical use-case, which helps to understand the interaction from the user's perspective.

One of our image processing experts received a contrast-enhanced ultrasound video from a radiologist. The original data doesn't give enough insight into the data, thus he needs image processing for further analysis. The top row of Figure 1 shows the thumbnail view of the filtered videos after uploading his data to our tool. Filter methods should improve the contrast of vessel boundaries. Therefore, edge preserving filters are chosen. Two selected filters, *Laplacian* and *Laplacian of Gaussian*, are

In-put	Length (sec)	FPS	Resolution	Upload (sec)	Graph (Sec)
V1	15	20	848x648	60	23
V2	3	25	848x648	23	8
V3	34	25	552x420	220	25
V4	145	5	850x480	160	40

Table 1: Computational Time. V1 - V3 are contrast-enhanced ultrasound data, varying in resolution, length and frames per second. V4 refers to a surveillance video with a low framerate.

dragged to the centered view for better comparison. The second filter highlights the vessels much better. Parameters can be finetuned, recomputed and the impact compared. The mask for customizing the parameters is shown in Figure 2(a). However, detecting differences on the same filter just by juxtaposition is a challenge, since the comparison relies on the memory of the user only. To get a measurement of the variation, he creates the difference graph to see which filter settings smooth out more artifacts than the other (Figure 2(b)). One specific second of the video shows high amplitude in the graph. This indicates the position, where the contrast-agent got distributed in almost all visible vessels. Sonoco now allows him to select a region and loop over this particular second. As mentioned above, Juxtaposition is not the best choice for comparison time-dependent data, so he opens the Checkerboard view to get an even better comparison for his selected region.

6 Conclusion and Future Work

Our presented scenario shows an easy way to compare time-dependent data. Reduction of the video duration supports the viewer's analysis and decision making. However, Mean Squared Error (MSE) is very sensitive to global changes, which could lead to inexpressive graphs. Ultrasound often comes with movements of the body part or sensor. The resulting MSE only shows high values, since artifacts appear in a different position for every time frame. Another difference detection method, like perceptual metrics, could be used to avoid the movement problem.

The Matlab computation time scales with the frame rate, duration and resolution. Table 1 shows an overview of the computation time. Our tests were done a machine with 32 Gigabyte of RAM and a 3GHz Intel(R) Core(TM) i7-5960X CPU. Matlab allows you to enable GPU rendering or parallel computing which speeds up the calculations. More efficient Matlab algorithms optimizes the computation time as well.

We are currently working on another way to superposition multiple videos. Malik [3] shows an extended approach of the checkerboard. Instead of quadratic patterns, his method uses hexagonal elements for comparison.

Sonoco was specifically designed for use by medical

image processing experts, but many of the employed visualization and interaction techniques may be helpful in other different fields, e.g. surveillance data. Our tool helps image processing experts to modify and compare time-dependent data without writing a line of code.

References

- [1] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D. Hansen, and Jonathan C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, October 2011.
- [2] Weisi Lin and C. C. Jay Kuo. Perceptual visual quality metrics: A survey. *J. Vis. Comun. Image Represent.*, 22(4):297–312, May 2011.
- [3] Muhammad Muddassir Malik, Christoph Heinzl, and M Eduard Groeller. Comparative visualization for parameter studies of dataset series. *Visualization and Computer Graphics, IEEE Transactions on*, 16(5):829–840, 2010.
- [4] Lucy Nowell, Elizabeth Hetzler, and Ted Tanasse. Change blindness in information visualization: A case study. In *infovis*, page 15. IEEE, 2001.
- [5] Nivedita Sahasrabudhe, John E. West, Raghu Machiraju, and Mark Janus. Structured spatial domain image and data comparison metrics. In *IEEE Visualization*, pages 97–104, 1999.
- [6] Johanna Schmidt, Meister Eduard Groeller, and Stefan Bruckner. Vaico: Visual analysis for image comparison. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2090–2099, 12 2013. Demo: <http://www.cg.tuwien.ac.at/jschmidt/vaico/>.
- [7] Trust Bio sonics Inc. Tbs-002 in rabbit liver tumor lesion detection (on philips cx50). <https://www.youtube.com/watch?v=vxI8nBSIjVc>.
- [8] K. Suomi and Jarkko Oikarinen. Visualization of changes in magnetic resonance image data. In *The 8-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2000, WSCG 2000*, 2000.

Interactive Visual Analysis of Animal Trajectories in a T-Maze

Fabrizia Bechtold*

Supervised by: Krešimir Matković†

VRVis Research Center
Vienna / Austria

Abstract

Research of animal behavior helps scientists to better understand different behavioral processes of animals and humans. The T-Maze is used to study the learning ability. Trajectories resulting from conducted runs in T-Maze are evaluated. Current state of the art methods analyze trajectories of individual animals separately, and then use only scalar features in the ensemble analysis. Interactive visual analysis can be used to improve the analysis of ensembles of trajectories and their features. In this paper, we introduce the *Gate-O-Gon*, developed to solve domain experts' tasks. The *Gate-O-Gon* is a specific visualization for T-Maze data showing overall direction characteristics of animals and it is integrated in a coordinated multiple views system. This paper presents the *Gate-O-Gon*, the integrated system, and analysis tasks which were identified by domain experts in related work.

Keywords: Interactive Visual Analysis, Animal Trajectories, Coordinated Multiple Views, Memory Learning

1 Introduction

To get a deeper understanding of the complex human being, its physiological processes, and eventually to develop better medical care, animal behavioral studies are often used. These studies enable research on neural mechanisms, underlying learning processes, or physiological processes, which can be very similar to those of humans [3]. One experiment commonly used in behavioral studies — used primarily on rodents — is the T-Maze. The T-Maze is a simple maze consisting of T-shaped segments with one right and one wrong path. By observing and tracking the rodents in the maze, researchers get insight on working memory and spatial learning. This enables a deeper insight on the animals' learning ability.

The T-Maze used in the experiments which are described in this paper consists of seven T-segments as shown in Figure 1. The animals are placed in the start area and recorded with a video tracking system until they reach the end area or time runs out.

The state of the art analysis commonly interprets the

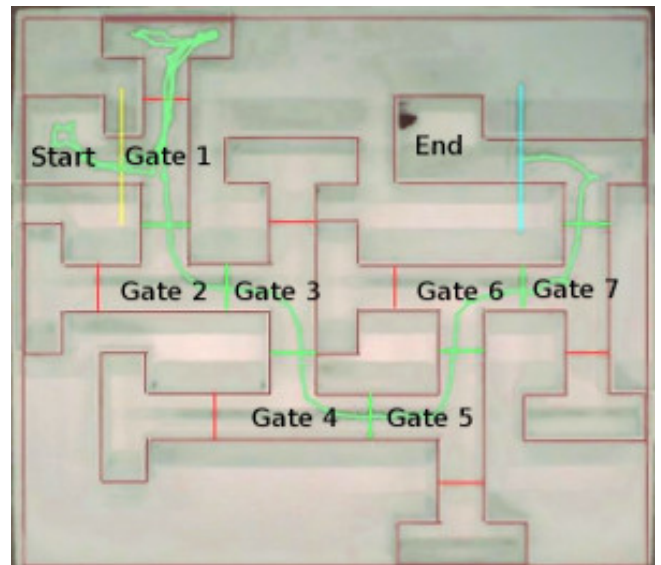


Figure 1: A picture of the T-Maze experiment showing the 7 T-segments, the start and the end area. The mouse is sitting in the end area after it successfully completed the task, the corresponding trajectory is also depicted.

rodents' trajectories separately, the analyzed results provide no information about the whole ensemble of trajectories. The analysis is based on scalar features of trajectories (such as length, average velocity, or time needed to reach the end area). Previous work has shown how interactive visual analysis can help in comprehending trajectory ensembles, originating from an open field experiment [5]. After domain experts gave positive feedback and motivation to continue the research, we extended it to the T-Maze data. The *Gate-O-Gon* is the first visual result motivated by specific tasks in the T-Maze experiments.

2 Interactive Visual Analysis and related work

Visual Analytics [4] is the science of analytical reasoning facilitated by visual interactive interfaces [8]. Interactive Visual Analysis gives insight on not apparent information contained in data. By interacting with the data and getting a prompt visual feedback new and different information

*fabrizia.bechtold@hotmail.com

†matkovic@vrvis.at

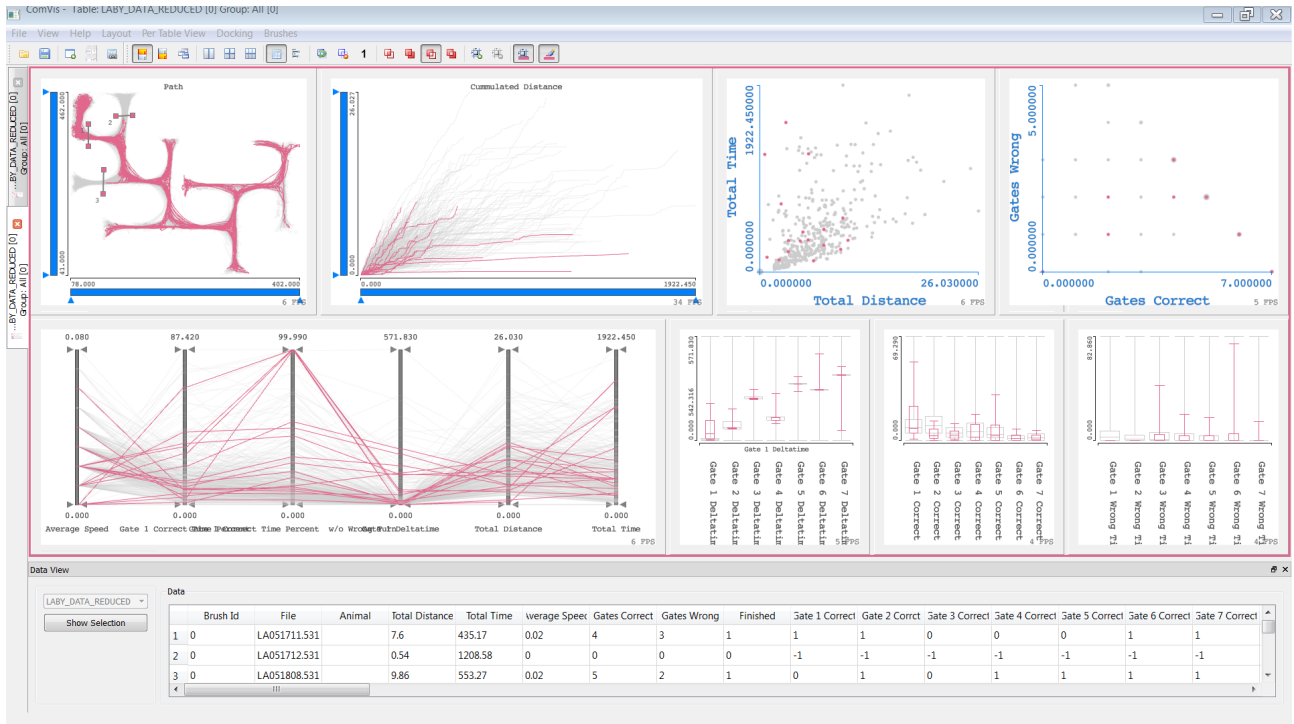


Figure 2: A screen capture of an analysis session using currently available views in the Coordinated Multiple View System 'ComVis'. Each view represents the whole ensemble of animal trajectories and gives different insight on it. Views such as 'Curve view', 'Box Plot', 'Parallel Coordinates' or 'Scatter Plot' are used.

can be obtained. Interactive Visual Analysis often relies on the well known mechanism of coordinated multiple views (CMV) [6, 1]. The coordinated multiple views display various attributes or dimensions of a data set by employing different views simultaneously. The views are linked, and user can brush—interactively select a subset of data in any view. Brushing (selecting) a subset of data in one view highlights the same subset in all other views. The *Gate-O-Gon* is integrated in the CMV-System *ComVis* (Figure 2) which supports composite iterative brushing, as well as an advanced data model which allows, in addition to scalar values, time series and trajectories as atomic units, i.e., attributes in a record. Andrienko et al. [2] describe visual analysis of movement data combining interactive visual displays, cognition, and reasoning with database operations and computational methods.

3 T-Maze Ensemble Data and Analysis Tasks

The T-Maze used for the underlying experiments measures 1.4×1.4 meters. It consists of seven T-Segments (gates), the start area, and the end area. A gate consists of a corridor splitting into two visually indistinguishable corridors, one leading to the next gate, the other in a dead end. The experiment was conducted for two weeks, the first week focusing on the animals' short term memory, and the sec-

ond week on their long term memory. In the first week the rats were put into the maze three times a day from Monday to Friday, in the second week only on Friday. The animals were motivated to run through the maze by giving them a reward (food placed in the end area). The runs were tracked with an infrared video camera in a dark room. Once an animal passed the correct path of the 7th gate (which is equivalent to successfully completing the run) or the time limit expired the tracking ended.

Several questions arose after the trajectories are collected. Do the animals pass the correct gate first, do they turn back and run in the wrong direction, how far do they run back, how far did they come before making a wrong choice etc. In order to answer these questions, the trajectories are first evaluated and several scalar features are computed. The features include, for example, total time spent in each gate, total time and distance traveled, number of right and wrong choices etc. A gate is evaluated as correctly passed if the animal crossed the correct gate first, even if turning back immediately after. The times spent in each gate are computed separately for running towards the end (correct direction) and towards the start (wrong direction). The time an animal spends in the overall wrong direction starts after it correctly passed a gate. If it then starts running in the wrong direction (towards the start area) this gate is memorized until the animal passes a correct gate again. All times spent in the traversed gates in between yield the total time spent in the wrong direction. For ex-

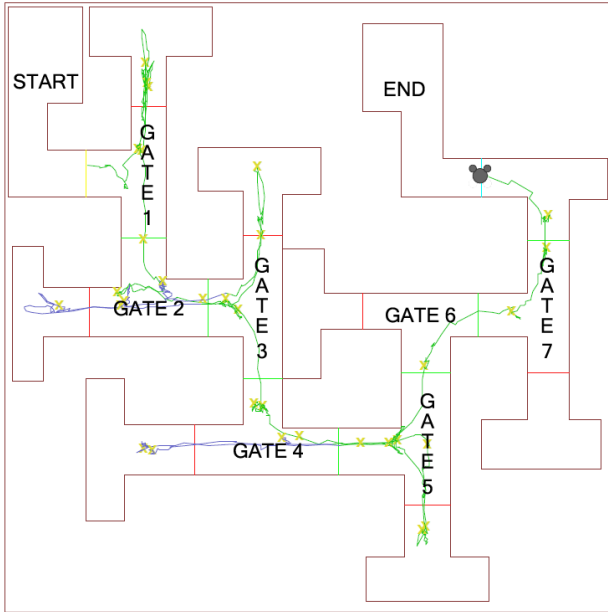


Figure 3: A screen capture of an analysis session using currently available views in the Coordinated Multiple View System 'ComVis'. The trajectory is displayed in green, with the sections where the mouse runs in the wrong directions in blue. The Yellow crosses are an indicator of time, marking the position of the mouse every 5 seconds.

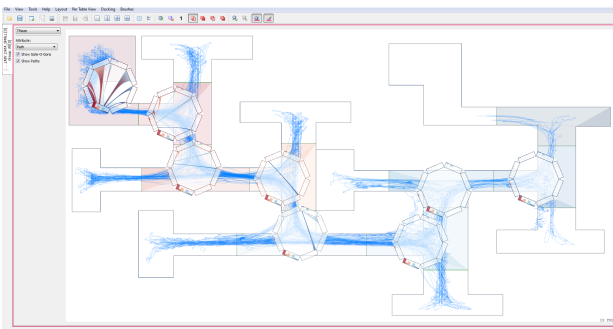


Figure 4: A screen capture of an analysis session using the Gate-O-Gon View. The T-Maze is visible in the back, each gate has a Gate-O-Gon depicting the distribution of time spent in the wrong direction for the respective gate.

ample, Figure 3 shows a trajectory partly green and purple. Green indicates the path running towards the end area, purple the wrong direction. Here the purple part show that the animal correctly passed gate five and returned to the start area. Here all times the animal spent in the gates four, three, two, one, and in the start area are computed.

Interactive Visual Analysis can help in answering the above questions by offering new possibilities to explore the whole trajectory ensembles at once. The data set which we analyze consists of one trajectory per run, with a total of 400 trajectories. In addition to the trajectory, each record has scalar attributes such as: Animal ID, total time,

total distance, number of gates passed correctly, number of gates not passed correctly, and many more. Finally, the record also has a time series attribute—the distance traveled as a function of time [7].

4 Design and Development of the Gate-O-Gon

The complex data set as introduced above, can be explored using the CMV-system as already shown for the open field data [5]. With this system, visualizing scalar values and interactively brushing and selecting subsets offers analysis possibilities which lead to better evaluation than conventional approaches. But there is still room for improvement. The system does not answer questions related to the overall direction of the animal, and times spent traveling in correct and wrong direction. These are specific questions for the T-Maze experiments. The next step is to visualize not only if an animal was heading into the wrong direction but also where it turned around and how far it had already come at any given time. How can the distances be compared, as there is a difference between an animal that came back to the start area from the gate 5 or from the gate 1. The solution is a new view which displays where an animal runs in the wrong way. Note that the animal can be running in the wrong way only in the start area and in the gates 1 to 6. The gate 7 is irrelevant as the experiment stops as the animal enter the end area, therefore the animal can never be in the area of the gate 7 while moving in the wrong direction.

Let n_i be the gates whereas $i \in \{0, 1, 2, \dots, 8\}$, and $i = 0$ and $i = 8$ for the Start and the End Area. The maximum reached gate is defined as $R \in \{1, 2, \dots, 7\}$. For each gate n_i where an animal is moving in the wrong direction we can identify R . "We are interested in the distribution of R for each gate. Note that we know that animals entered gate n_i from gate n_{i-1} or n_{i+1} , but we want to know how far the animal already had gone" [7].

The distribution of R is calculated from each trajectory. For this project the trajectories are already pre-evaluated and contain scalar values as well as the trajectory data. The trajectory data consists of coordinates. To compute the total distance traveled in the wrong direction each coordinate is evaluated separately. At any point of the evaluation the highest reached gate and the lowest retraced gate are known. Each coordinate is tested on it's position in the maze and assigned to the corresponding gate. Therefore the current direction the animal is running is known and whenever it changes directions to running the correct way, after retracing once, the total distance traveled in the wrong direction is known and saved in a distinct data-structure.

We propose a novel visualization of wrong direction distribution in respect to the gates. The newly introduced *Gate-O-Gon* depicts the wrong direction and the distribu-

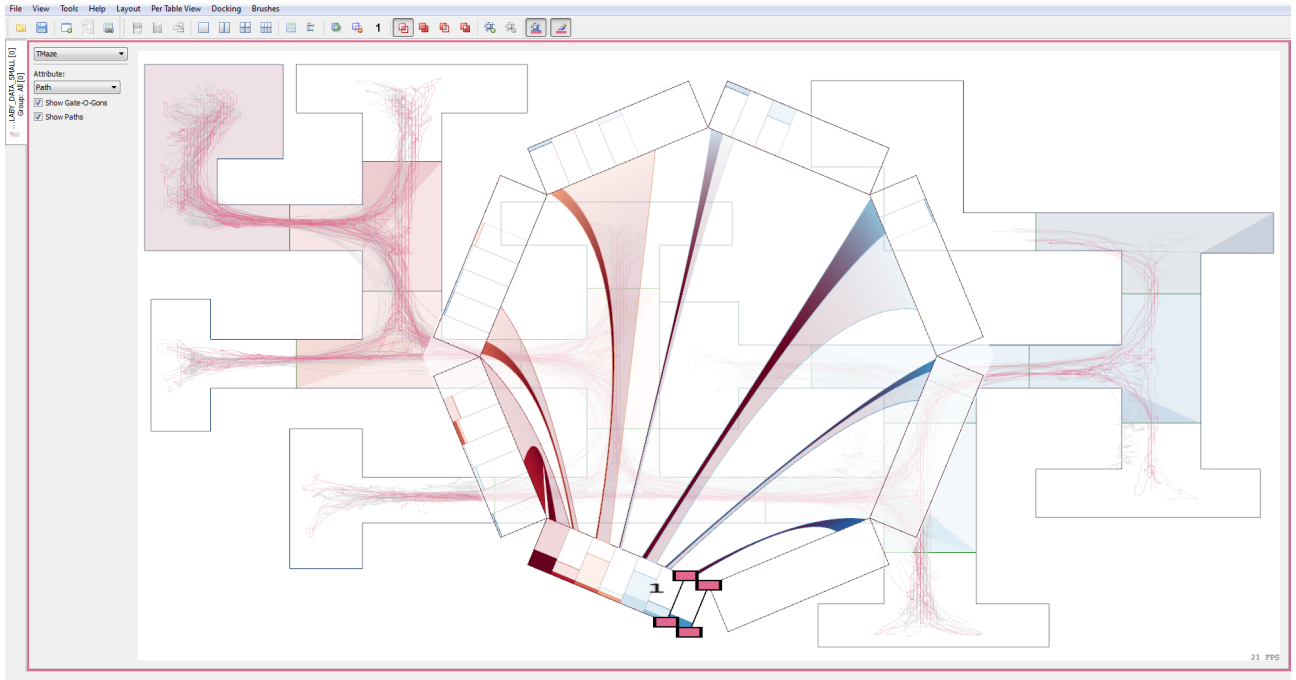


Figure 5: This screen capture shows an analysis session using the Gate-O-Gon view. The detailed view of the start area is selected. In this case a subset was selected as visible from the quadrangle in the histogram area. The subset contains all trajectories where the animal had passed the correct gate line of gate 7 but returned to the start area shortly after.

tion of R . It can be superimposed in the T-Maze itself. The set of trajectories can also be shown. As each gate has its own distribution, the *Gate-O-Gon* is placed on each gate. Figure 4 shows the current version of the *Gate-O-Gon* view. It gives a good overview of all wrong direction distributions, but is too small to get detailed information, hence a detailed view showing only the distribution of the selected gate, as shown in Figure 5, is also available.

The *Gate-O-Gon* (as depicted in Figure 5) is shaped like an octagon, each side representing a gate. The idea to use an octagonal shape was born from the wish to visualize a direct connection between two gates and that for all gates. The gates starting with the start area (gate 0) in the bottom left, followed by gate 1, 2, ..., 6 are shown in a clockwise order.

The edges of the octagon also show the distribution of R of all gates as a histogram, with the selected gate highlighted and the others faded. At gate 0 the values of R can go from 1 to 7, at gate 1 from 2 to 7 etc. At gate n_i the values of R can go from $i + 1$ to 7.

Additionally, the gate edges are connected by a strap also representing the distribution of R . At present the only distribution of the total times an animal was running in the wrong direction from gate n_j to gate n_i with $j > i$ is computed. Future design might contain other parameters such as correct time. For the selected gate i only connecting straps to gates $i + 1$ are available, since, as mentioned before, an animal cannot run in the wrong direction when coming from a lower gate h to gate i , with $h < i$. The thickness of the strap reflects the distribution of R and it shows

the same value as the histogram in the gate edges.

For easier distinction, each gate has a different color, blending from deep red to deep blue. For example, gate 0 is deep red, gate 4 is light blue and gate 7 is depicted in deep blue. The colors are reflected in the connecting strap and histogram, as well as in the T-Maze itself, where each gate is softly colored in the equivalent color (Figure 5). To emphasize the return direction from a higher gate connecting straps color is blending from the color of gate n_j to the color of the selected gate n_i . For example, in Figure 5 the connecting strap from the gate 5 to the gate 0 is blending from blue to deep red. These colors are also used to paint the background of the corresponding gates in the background of the MCV.

When selecting a subset of the data in a different view the *Gate-O-Gon* changes transparency, highlighting the brushed data set. At present brushing in the *Gate-O-Gon* is only possible in the detail view's histogram. Figure 5 shows brushing in the *Gate-O-Gon* histogram, returning animals from gate 7 to the start area are selected. The brushed subset is highlighted in the connecting straps as well as in the histogram. The whole ensemble set is hinted at by reduced opacity.

In the background of the *Gate-O-Gon-View* the used T-Maze is displayed. Inside the T-Maze the trajectories of the data set can be seen, using standard coloring of the ComVis MCV. Blue represents the whole, unbrushed data set, grey the non-selected subset, pink the selected subset.

5 Conclusions

This paper describes an idea for improving the analysis of animal trajectories ensembles. It introduces the *Gate-O-Gon*, a novel visualization which supports exploration and analysis of many trajectories at once. We will evaluate if the Gate-O-Gon can give answers to the important question of animal direction in particular gates and the previously achieved furthest point in case of the wrong direction.

Further improvement steps could include further development of the *Gate-O-Gon*, different approaches to display the distribution of the total traveled distance, or a heat map. We, moreover, plan a close cooperation with domain experts and conduct a comprehensive evaluation.

References

- [1] G. Andrienko and N. Andrienko. Coordinated multiple views: a critical view. In *Coordinated and Multiple Views in Exploratory Visualization, 2007. CMV '07. Fifth International Conference on*, pages 72–74, July 2007.
- [2] Gennady L. Andrienko, Natalia V. Andrienko, Peter Bak, Daniel A. Keim, and Stefan Wrobel. *Visual Analytics of Movement*. Springer, 2013.
- [3] H. Bubna-Littitz and J. Jahn. Psychometric testing in rats during normal ageing. procedures and results. *J Neural Transm Suppl*, 44:97–109, 1994.
- [4] Joern Kohlhammer, Daniel A. Keim, Giuseppe Santucci, Gennady Andrienko, and M. Pohl. Solving problems with visual analytics. In *The European Future Technologies Conference and Exhibition 2011*. Procedia Computer Science, 2011.
- [5] Kresimir Matkovic, Christiana Winding, Rainer Splechtna, and Michael Balka. Interactive Visual Analysis of Ethological Studies: Getting Insight from Large Ensembles of Animals’ Paths. In *EuroVA 2012: International Workshop on Visual Analytics*, pages 85–89, 2012.
- [6] Jonathan C. Roberts. State of the Art: Coordinated & Multiple Views in Exploratory Visualization. In *Proc. of the 5th International Conference on Coordinated & Multiple Views in Exploratory Visualization*. IEEE CS Press, 2007.
- [7] R. Splechtna, F. Bechthold, C. Winding, M. Balka, and K. Matković. Interactive visual analysis of animal trajectories in a t-maze, oct 2014.
- [8] J J Thomas and K A Cook. *Illuminating the path: The research and development agenda for visual analytics*. IEEE, 2005.

Image Processing & Vision

Recognition of Important Features of Triangulated Human Head Models

Kateřina Kubásková
Supervised by: Ivana Kolingerová

Department of Computer Science and Engineering
University of West Bohemia
Pilsen / Czech Republic

Abstract

Feature detection is often used in geoinformatics or computer graphics. A lot of feature detection methods have been developed. The goal of this work is to find methods suitable to detect features, implement and test them on the triangulated model of human head which is used to create an identikit.

Keywords: feature, curvature, thresholding, triangular model, morphological operators, MLS approximation

1 Introduction

The features of the model usually refer to a region of a part with some interesting geometric or topological properties. The detection of features is automatic and simple for human brain, but not for the computer. There have been developed many methods for recognition of features and they are used in various fields; CAD, NPR, computer vision, computer graphics, geoinformatics or cartography.

The aim of this work is to find methods suitable to detect features, implement and test them on a triangulated model of human head used to create a 3D identikit - a 3D portrait enabling to identify a person. The proposed method should be able to detect the features automatically or manually. Next step is to detect important points - the control points (the corners of eyes, lips, etc.). It is important to detect them, because they are used to deform a human head model or add a texture (see Figure 1). These points have been found manually and it was very time-consuming.

The rest of the paper is organized as follows. Section 2 gives an overview of the needed theory and existing methods. Section 3 presents the proposed methods to detect features on models of human head. Section 4 analyses results and experiments and Section 5 offers conclusions.

2 Theory and state of the art

In this section we briefly describe necessary background needed for the methods that can be used to recognize important features.

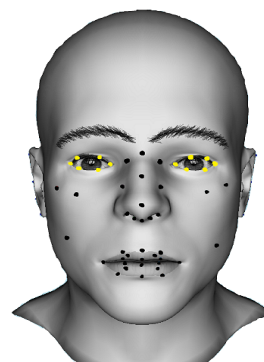


Figure 1: Identikit of a man seen together with control points.

2.1 Differential geometry

At the beginning it is necessary to define a regular surface. Let $D \subset \mathbb{R}^2$ denote an open subset. A smooth vector function $r : D \rightarrow \mathbb{R}^3$ of two variables is called parametrization for the surface $S \subset \mathbb{E}^3$ consisting of all points P with $\vec{OP} = r(u, v)$ with $(u, v) \in D$ if

1. r is one-to-one map,
2. the partial derivatives $r_u(u, v) = [x_u(u, v), y_u(u, v), z_u(u, v)]$ and $r_v(u, v) = [x_v(u, v), y_v(u, v), z_v(u, v)]$ are linearly independent at every point $(u, v) \in D$. A subset S that has a parametrization r as above is called a **regular surface** [12].

Next we define the **tangent plane** to the surface $S \subset \mathbb{E}^3$ at P as a plane that contains all tangent vectors to curves on S in a point P . The tangent plane is spanned by the tangent vectors to two parameter curves: $\rho(s, t) = r(u, v) + sr_u(u, v) + tr_v(u, v)$.

A vector $n \in \mathbb{R}^3$ is called a **normal vector** to S at P if n is perpendicular to all tangent vectors $v \in \rho(s, t)$. Let n be a unit normal vector, then $n(P) = n(u, v) = \frac{r_u(u, v) \times r_v(u, v)}{|r_u(u, v) \times r_v(u, v)|}$ [12].

An important property that characterizes a surface is curvature because it measures a local bending. The **normal curvature** ${}^n\kappa(i)$ is defined as the curvature of the curve that belongs both to the surface S and to the normal plane containing both n and unit tangent vector of the

curve. The normal curvature can be computed as ${}^n\kappa(i) = \ddot{r} \cdot n$, where \ddot{r} is the second derivative of a vector function r parametrized by the arc length. It can be negative, positive or zero [10].

At every point we define two types of directions - principal and asymptotic. Asymptotic directions are the directions with normal curvatures equal to zero. Principal directions are orthogonal directions with maximal or minimal normal curvatures. These two **principal curvatures** are called maximal curvature κ_{max} and minimal curvature κ_{min} .

The principal curvatures determine the range of the curvature of the surface in local neighborhood of a point. We can use the principal direction to compute other curvatures. The **Gaussian curvature** at a point $P \in S$ is obtained by the relation in Equation 1.

$$K(P) = \kappa_{max}(P) \cdot \kappa_{min}(P). \quad (1)$$

The average of principal curvatures is called the **mean curvature** (Equation 2).

$$H(P) = \frac{1}{2}(\kappa_{max}(P) + \kappa_{min}(P)). \quad (2)$$

More information can be found in [1, 10, 12].

2.2 Methods for feature recognition

There are several different representations for surfaces used in computer graphics but triangular meshes seem to be prevalent. Our head models are represented by 3D triangular meshes defined by a graph $G = (V, E, F)$, where V is a set of vertices of the mesh, E is a set of edges, F is a set of triangles. Due to popularity of triangular meshes representation there are many studies on feature detection on them.

There are some classical methods which somehow evaluate (classify) the vertices or edges and threshold them as belonging to a feature or not. Hubeli et al. in [3] describe some classification operators such as second order difference or best fit polynomial. Another classification operators can be the discrete Laplace-Beltrami operator from [5].

A range of robust methods is available to detect features directly from the meshes. Rössl et al. [4] compute curvature and use morphological operators to produce feature lines. Ohtake et al. [7] have developed a technique that is based on computing curvature tensors and their derivatives at each vertex by means of the projection and global approximation of an implicit surface. Their method achieves good results, but the detection process is time-consuming. Yoshizawa et al. [8] used local polynomial fitting of triangulated meshes to estimate curvature tensors and their derivatives and it reduces the computation time. Another reduction of time-complexity of estimation of the curvatures and their derivatives are addressed in [9] by applying the modified moving-least-squares (MLS) approximation directly to the mesh.

Kim et al. [11] have developed a technique based on voting tensor theory, which can handle n-dimensional triangular mesh. Karlíček in [2] tested and compared methods suitable for various classes of geometric objects, including triangulated head models.

3 Proposed method

In the previous section we described some methods that are used for detection of features. We selected and combined those methods that can handle sharp angles between neighbouring triangles and triangulated approximation of smooth surfaces and also work in optimal time-complexity. Our proposed method consists of the following parts:

- Vertex evaluation - for detection of features
- Evaluation thresholding
- Detection of important areas
- Detection of control points

3.1 Vertex evaluation

We experimented with three methods. The first one uses discrete curvature and was chosen because of good experience reported for head models in [2]. The second one uses a modified MLS approximation and the last one is Laplace-Beltrami operator.

3.1.1 Discrete curvature

The triangular model is a piecewise linear function and it is not possible to determine the derivation. We use approximation equations for curvature described in [10].

To compute the discrete curvature, a "mixed area" for each vertex, denoted A_{mixed} , is needed. It is based on Voronoi region defined for non-obtuse triangle. The area of the Voronoi region can be computed as

$$A_{Voronoi} = \frac{1}{8} \sum_{j \in N_i} (\cot \alpha_{ij} + \cot \beta_{ij}) \|v_i - v_j\|^2,$$

where N_i is the set of 1-ring neighbour vertices of vertex v_i , α_{ij} and β_{ij} are the two angles opposite to the edge in the two triangles sharing the edge (v_i, v_j) (see Figure 2a). This expression for the Voronoi area does not hold in case of obtuse angles. If there is an obtuse triangle among the 1-ring neighbours, the Voronoi region either extends beyond the 1-ring or is truncated compared to our area computation.

The mixed area A_{mixed} is computed as follows: for each non-obtuse triangle we use the Voronoi area, for each obtuse triangle we use the midpoint of the edge opposite to the obtuse angle and connect the midpoint to the centers of the adjacent edges. The mixed area is in Figure 2b.

Now when the mixed area is explained, we can express the curvature. Mean curvature is computed by Equation 3:

$$H(v_i) = \frac{1}{4A_{mixed}} \sum_{j \in N_i} (\cot \alpha_{ij} + \cot \beta_{ij}) \|v_i - v_j\|^2. \quad (3)$$

Gaussian curvature is given by Equation 4:

$$K(v_i) = (2\pi - \sum_{j=1}^{\#f} \theta_j) / A_{mixed}, \quad (4)$$

where θ_j is the angle of the j -th face at the vertex v_i , and $\#f$ denotes the number of faces around this vertex. This operator will return zero for any flat surface.

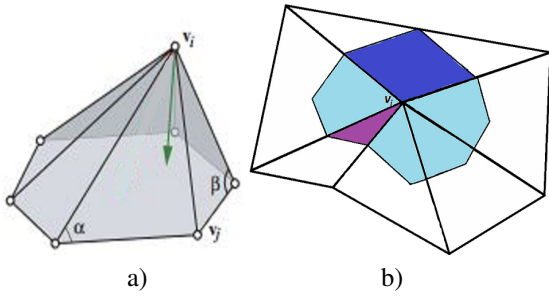


Figure 2: a) 1-ring neighbourhood of vertex v_i . The angles opposite to the edge (v_i, v_j) are α and β . Cotangent (green) Laplacian vectors for vertex v_i [5]. b) Mixed area at vertex v_i . Light blue are the areas in non-obtuse triangles, dark blue is in a triangle with an obtuse angle at the vertex v_i and the purple area in the triangle with an obtuse angle at one of the remaining vertices [2].

We have seen in Section 2.1 that the mean and Gaussian curvatures are easy to express in terms of the two principal curvatures ${}^n\kappa_{min}$ and ${}^n\kappa_{max}$. Therefore we can define the discrete principal curvatures by Equation 5.

$$\begin{aligned} {}^n\kappa_{max}(v_i) &= H(v_i) + \sqrt{H^2(v_i) - K(v_i)}, \\ {}^n\kappa_{min}(v_i) &= H(v_i) - \sqrt{H^2(v_i) - K(v_i)}. \end{aligned} \quad (5)$$

We must make sure that H^2 is always larger than K to avoid numerical problems. If it is not the case than set $\sqrt{H^2(v_i) - K(v_i)}$ to zero.

3.1.2 MLS approximation

In this section we use modified MLS approximation described in [9].

Given a vertex v_i , we first find a local reference plane H . We use a tangent plane orthogonal to a vertex normal at vertex v_i as a local reference plane. Estimation of the normal vector $n = (n_1, n_2, n_3)^T$ at each vertex is done by averaging the normals of a 1-neighbourhood of triangles and the vertex itself. The tangent plane in general form is obtained

$$H : x \cdot n_1 + y \cdot n_2 + z \cdot n_3 = d.$$

Let N_i^k be k -neighbourhood of vertices at vertex v_i , and let $X = \{x_i\}_{i \in N_i^k}$ be the orthogonal projections of the vertices N_i^k to H , represented in a specific orthonormal coordinate system defined on H , so that the origin is v_i . The size of the k -neighbourhood depends on the user. But 1 and 2-neighbourhood generates a poor MLS surface in which it is difficult to locate the principal directions. Adding more neighbors leads to an increased time-complexity. In our experiments we usually use 3-neighbourhood.

Next, we define a local approximation to the surface as the third-degree polynomial p that minimizes the weighted least-squares error given by

$$\mathcal{E} = \sum_{v_j \in N_i^k} (p(x_j) - f_j)^2 \theta(\|v_i - v_j\|),$$

$$\theta(\|v_i - v_j\|) = e^{-(\|v_i - v_j\|^2 / h^2)},$$

where the function θ is Gaussian non-negative weighting. The parameter h is the average of the lengths of the 1-neighborhood edges of v_i and $f_j = n^T v_j - d$ are the heights of the vertices v_j over H .

The error function can be rewritten as

$$\mathcal{E} = \sum_{v_j \in N_i^k} (b(x_j)^T c - f_j)^2 \theta(\|v_i - v_j\|).$$

where $p(x) = b(x)^T c$, $b(x)$ is the base vector of the polynomial and c is a vector of unknown coefficients. Then we put the partial derivations of the error function equal to zero to find the coefficients of the polynomial and get a linear system of equations given by Equation 6.

$$Ac = d, \quad (6)$$

where $A = \sum_{v_j \in N_i^k} 2b(x_j)b(x_j)^T \theta(\|v_i - v_j\|)$ and

$d = \sum_{v_j \in N_i^k} b(x_j) f_j \theta(\|v_i - v_j\|)$. The solution of the linear system is $c = A^{-1}d$.

Now we can estimate the principal curvatures at each vertex v_i . We convert the MLS polynomial $z = p(x_i)$ into the implicit surface $F = z - p(x_i)$.

For each vertex v_i we can estimate the unit normal vector at v_i as $n = \nabla F / (\|\nabla F\|)$. Next we can estimate the principal curvatures κ in the associated principal directions $t = (t_1, t_2, t_3)$ as follows

$$\kappa = \frac{F_{ij} t_i t_j}{\|\nabla F\|},$$

where F_{ij} denotes the second partial derivatives of F . Directions t_{max} and t_{min} are given by eigenvectors corresponding to the two non-zero eigenvalues of ∇n . The matrix ∇n is given by Equation 7.

$$\nabla n = \begin{bmatrix} \frac{\partial n_1}{\partial x} & \frac{\partial n_1}{\partial y} & \frac{\partial n_1}{\partial z} \\ \frac{\partial n_2}{\partial x} & \frac{\partial n_2}{\partial y} & \frac{\partial n_2}{\partial z} \\ \frac{\partial n_3}{\partial x} & \frac{\partial n_3}{\partial y} & \frac{\partial n_3}{\partial z} \end{bmatrix}. \quad (7)$$

Mean and Gaussian curvature can be then estimated using principal curvatures by expressions in Section 2.1. Unlike the method in Section 3.1.1, now we can get also negative values.

3.1.3 Laplace-Beltrami operator

The last method for classification of vertices is the Laplace-Beltrami operator [5, 6]. The discrete Laplace-Beltrami operator for a triangular mesh at the vertex v_i is defined in Equation 8.

$$\delta_i = \sum_{(v_i, v_j) \in \mathbf{E}} \omega_{ij}(v_j - v_i), \quad (8)$$

where $\sum_{(v_i, v_j) \in \mathbf{E}} \omega_{ij} = 1$ and the choice of weights defines the character of δ_i .

We choose the cotangent weights $\omega_{ij} = \frac{\cot \alpha + \cot \beta}{2}$, where α and β are the angles opposite to the edge (v_i, v_j) (see Figure 2a). The cotangent Laplacian is zero on planar 1-rings because of geometry-dependence. The cotangent Laplacian vector can be seen in Figure 2.

The cotangent Laplace-Beltrami operator is dependent on the size of triangles, therefore, for evaluation of vertices we have to use a modified cotangent Laplacian, which is defined in Equation 9.

$$\hat{\delta}_i = \frac{1}{d_i} \sum_{(v_i, v_j) \in \mathbf{E}} \omega_{ij}(v_j - v_i), \quad (9)$$

where $d_i = \frac{S_i}{3}$ and S_i is the area of the adjacent triangles at vertex v_i . The discrete Laplace-Beltrami operator is a vector, therefore, we use the size of the vector for evaluation of vertices.

It is also possible to use truncating of evaluation. A percentage of vertices with the highest evaluation are selected and they get a new value, the highest evaluation without already selected vertices.

3.2 Thresholding

After we have classified vertices using the described methods, we apply a standard thresholding to evaluate the vertices. The user specifies the threshold parameter as minimal weight that a vertex must have to be included into the subset of feature vertices. In case of the curvature computed using MLS approximate we have also negative weights; so in negative thresholding the user specifies a maximal weight.

We normalize the values of evaluation using the Equation 10.

$$w(v_i) = \frac{w(v_i)}{|w_{max}|} \cdot 100 \quad [\%], \quad (10)$$

where $w(v_i)$ is either the curvature from Section 3.1.1 or 3.1.2 or the size of cotangent Laplacian from Section 3.1.3

and $|w_{max}|$ is the absolute value of maximum evaluation of all vertices.

In agreement with recommendation in [2] we can also apply morphological operators on feature vertices to achieve better results. Morphological operators deal only with binary values. So we use a thresholding operation to determine the feature vector Ω :

$$\Omega_i = \begin{cases} 1 & \text{for } w_i \in [a, b] \\ 0 & \text{otherwise,} \end{cases}$$

where w_i is evaluation at vertex v_i and a, b are thresholding parameters. A set \mathcal{F}_s of significant vertices can be expressed as $\mathcal{F}_s := \{v_j \in \mathbf{F} \mid \Omega_j = 1\}$.

The morphological operators for triangulated meshes are defined as follows.

Dilation

Let $\mathcal{F}_s \subseteq \{1, \dots, N\}$. The dilation of \mathcal{F}_s by k -neighbourhood \mathbf{N}^k is defined as.

$$\text{dilate}^k(\mathcal{F}_s) := \{v_j \mid \exists v_i \in \mathcal{F}_s : v_j \in \mathbf{N}_i^k\}.$$

The dilation operator adds vertices to the feature. It can therefore be used to fill "holes" in the features.

Erosion

Let $\mathcal{F}_s \subseteq \{1, \dots, N\}$. The erosion of \mathcal{F}_s by k -neighbourhood \mathbf{N}^k is defined as

$$\text{erode}^k(\mathcal{F}_s) := \{v_j \mid \mathbf{N}_j^k \subseteq \mathcal{F}_s\}.$$

The erosion operator reverses the effect of dilation. It cuts off undesired branches.

Opening

The opening operator is defined as

$$\text{open}^k(\mathcal{F}_s) = \text{dilate}^k(\text{erode}^k(\mathcal{F}_s)).$$

The opening operator applies the erosion and after it applies the dilation. This application removes undesired artifacts, but the size of the feature is not preserved.

Closing

The closing operator is defined

$$\text{close}^k(\mathcal{F}_s) = \text{erode}^k(\text{dilate}^k(\mathcal{F}_s)).$$

In closing \mathcal{F} the feature is first grown and shrinked afterwards. It fills holes in the inner region of the feature and fills bays along the boundary.

3.3 Detection of important areas

Our aim is mainly to detect features automatically. In this case, we have to find a suitable threshold that gives the best result. One minimal threshold does not achieve a good

result on the whole model. Due to this we decided to detect important areas, which include the entire area of eyes, ears, nose and lips. Then we detect features and by applying different thresholds in different areas we get better results.

We based the detection of important areas on the simplest from the presented vertex evaluation approaches, i.e. the mean curvature from Section 3.1.1. The minimal threshold is selected as an average of evaluations. In Figure 3 we can see the resulting features. Then morphological operators are applied: closing operator, the opening operator, another closing operator and a dilation. The results of this algorithm are dependent on the choice of k-neighbourhood and the number of vertices N .

From experimental results we chose the following k-neighborhood:

- $N \leq 13000 \Rightarrow k = 5$,
- $13000 \leq N \leq 20000 \Rightarrow k = 10$,
- $N > 20000 \Rightarrow k = 15$.

If an automated processing is not required, the user may select whether the last operator, the dilation, is applied and chooses its k-neighborhood.

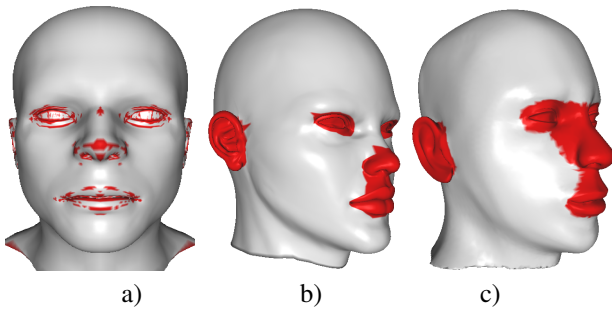


Figure 3: Detection of important areas: a) Features after evaluation by mean curvature and thresholding by average of evaluations. Detection of important areas with b) 15, c) 10 - neighbourhood.

Now the important areas on the head are detected. The area of ears is already separated but the rest is so far connected which is not what the user needs.

To separate these areas, two methods can be applied. The first method uses a simple distance based selection. The boundary between eyes and nose is in the half of distance between the point with maximal x-coordinate and the peak of the nose (maximal y-coordinate). The boundary between the nose and the lips is in the two thirds of the distance between the peak of the nose and the point with the minimal z-coordinate (see Figure 4).

The second method is based on the first, but the detection of the nose is modified. The area of the nose contains all vertices in the marked region in Figure 5. This region is formed by the nose root (the lowest point on the line of the nose), the boundary between the eyes and the nose and the width of the nose.

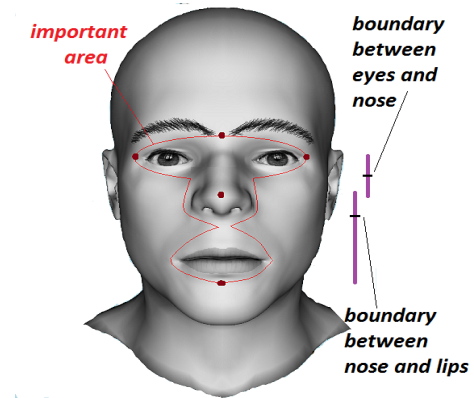


Figure 4: First method of separation of areas.

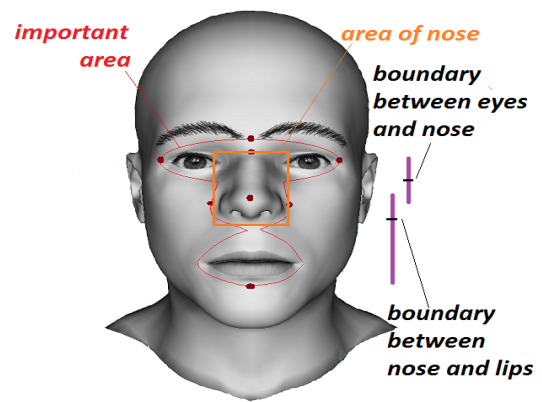


Figure 5: Second method of separation of areas.

3.4 Detection of control points

Next goal is to try to find the control points - the most important points of the feature. By the feature detection we get the border of the lips, eyes and ears, then detection of the control points is not difficult as described further.

Eyes On the eyes we need to detect the corners and 8 points. The corners of an eye have extreme x-coordinates (see Figure 6).

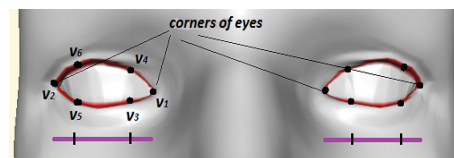


Figure 6: Detection of control points on eyes. Features (red) and control points (black).

Lips At first we detect the corners of lips (v_1, v_2) as extreme x-coordinates. Using these two points we detect other 12 points. Some of them are on the border of lips and some of them in the middle of the lips (see Figure 7).

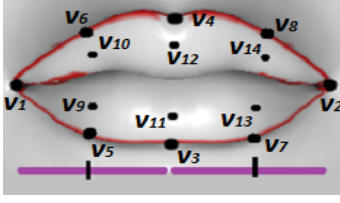


Figure 7: Detection of control points on lips. Features (red) and control points (black).

Ear We detect 5 points on the ears (see Figure 8a) First two points (v_1, v_2) have extreme z-coordinates. Other two points (v_3, v_4) lie in the first quarter of the distance from v_1 or v_2 and has the minimal y-coordinate. The last point v_5 lies in the middle of the ear.

Nose The detection on the nose is the most difficult task. Some of the points have to be detected outside of the set of features. The tip v_1 of the nose is found by the maximal y-coordinate, the nose root v_2 have minimal y-coordinate on the line of nose. The point v_3 lies in the middle between v_1 and v_2 . Other 5 points are detected in the features using the already known points, details see [1]. The detection can be seen in Figure 8b).

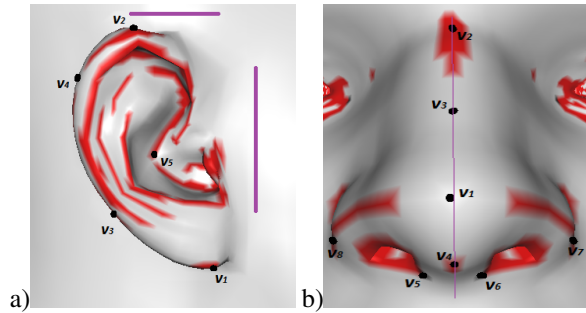


Figure 8: Detection of control points: a) on ears, b) on nose. Features (red) and control points (black).

4 Experiments and results

In this section we show results of the proposed methods. The method was implemented in Microsoft Visual Studio 2010 using the programming language C# and .Net Framework version 4. The program, where the method from Section 3.1.1 and morphological operators were implemented, was taken from [2]. We added other methods described in Section 3.1.2 and 3.1.3 and algorithms for automatic detection, detection of important areas and detection of control points.

We tested our proposed methods on 11 models. Each method works on various areas of the head differently. The choice of thresholds is important. First, we evaluate the results of curvature computation described in Section 3.1.1, then our implemented methods.

4.1 Discrete mean and maximal curvature

As minimal and Gaussian curvature do not achieve good results we present only results of mean and maximal curvature (see Figure 9).

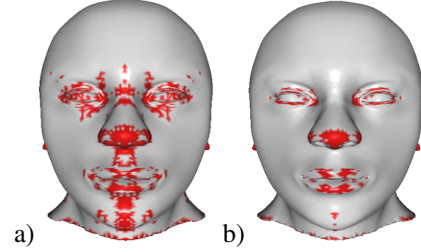


Figure 9: Detection of features using: a) minimal curvature from Section 3.1.1, b) Gaussian curvature from Section 3.1.2.

Mean curvature provides very good results on the area of eyes and ears. We get the border of eyes and ears. On the lips and nose the results of mean curvature are not very satisfactory. On a few models we get the border of the lips, on the nose we get only features around nostrils (see Figure 10 a).

Results of maximal curvature are similar to the mean curvature. On the eyes and ears we have also good results and even on the lips are the results quite good. With nose we have also problems and have only features around nostrils. The results can be seen in Figure 10 b).

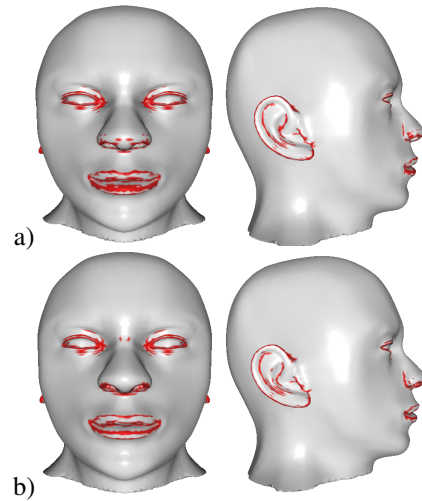


Figure 10: Detection of features: a) mean curvature, b) maximal curvature.

4.2 Mean and maximal curvature computed using MLS

Let us recall that evaluation computed by this method have also negative values, therefore, the color of features in figures is different than before (blue for negative and red for positive values).

The result of mean and maximal curvature are very satisfactory on the area of eyes and ears. On the lips the maximal curvature is better than the mean curvature. The area of the nose is complicated. Using maximal curvature one can detect the bridge of nose on some models. On other models again only the nostrils are detected. The results can be seen in Figure 11.

4.3 Cotangent Laplace-Beltrami operator

The definition of cotangent Laplace-Beltrami operator is very similar to the mean curvature in 3.1.1. Therefore, the results of these methods are not too different (see Figure 12).

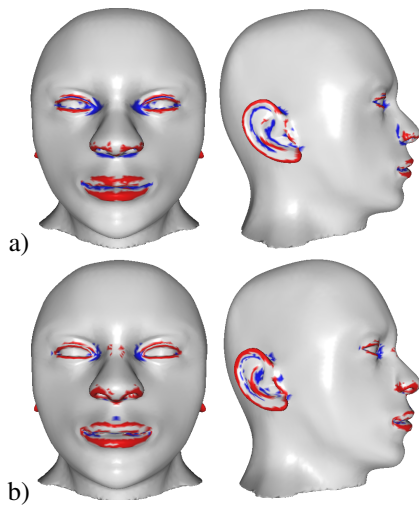


Figure 11: Detection of features: a) mean curvature computed using MLS, b) maximal curvature computed using MLS.

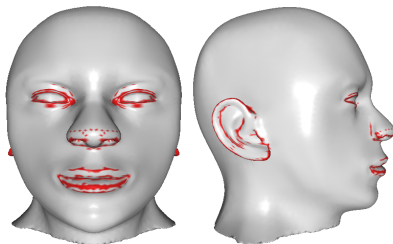


Figure 12: Features using Laplace-Beltrami operator.

4.4 Detection of important areas

The detection of important areas works on all models very well. In case that the important areas are too small for the user's needs, dilation can be applied (see Figure 13).

Next, we show the result of selection of important areas. We can see the difference of two methods in Figure 14. Using the first method, we do not get all necessary vertices in the area of nose (see Figure 14b). The second method fixes this failure and we get the whole area of nose in all models (see Figure 14c,d). It is possible to apply dilation

or erosion on each important area if the user is not satisfied with the result achieved so far.

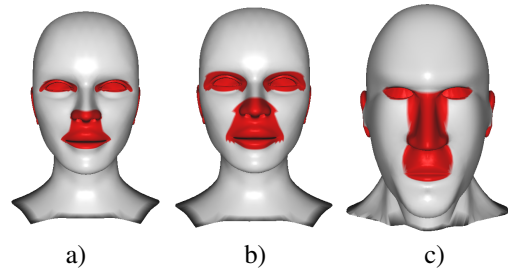


Figure 13: Detection of important areas: a,c) the dilation is not used, b) dilation in 3-neighborhood.

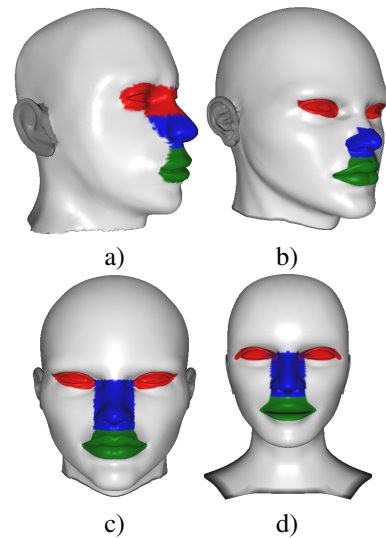


Figure 14: Result of selection of important areas: a,b) the first method, c,d) the second method.

4.5 Automatic detection

Our aim is the automatic detection of features, but for this we need to know a suitable configuration of parameters. It is impossible to determine an optimal threshold that will work on all heads. Therefore, for each area we chose the method that generally works well on this area for most tested models.

Our proposed automatic detection is following:

- Eyes: Maximal curvature from MLS approximate
- Ears: Mean curvature from MLS approximate
- Lips: Mean curvature with 5% values truncated
- Nose: Maximal curvature

The results of automatic detection of features are shown in Figure 15. The automatic detection is very good in 75 percent of models.

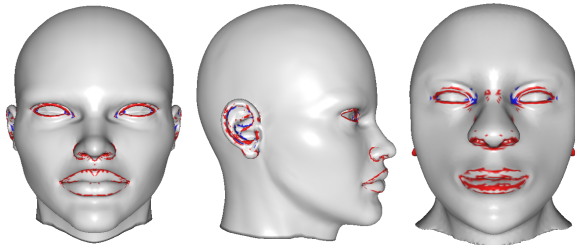


Figure 15: Automatic detection

4.6 Detection of control points

The last group are results of the detection of control points. Automatic detection of control points after the automatic detection of features is shown in Figure 16. These results are very good.

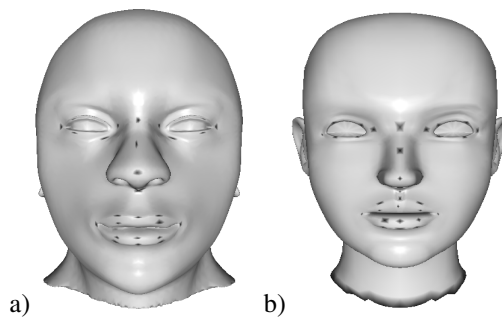


Figure 16: Detection of control points.

5 Conclusion

In this paper we have presented methods to detect important features on triangulated human head models. The aim of this work was to propose automatic detection and detection with manual choice of parameters.

We used three methods. We detected important areas using the method described in Section 3.1.1. To classification of vertices we then used Laplace-Beltrami operator (Section 3.1.3) and estimation curvature computed using MLS approximate (Section 3.1.2). Then we get the features by applying thresholding and we could use morphological operator to modify the features. We can also detect the features on each important area. In automatic detection we used methods which have the best results on the important area and the minimal threshold parameter is determined by an average of evaluation on this area.

The proposed method was tested on several models used for 3D identikits. The automatic method achieves a good result on most models. The worst results are in the area of the nose. Future development should consider other methods working well automatically also in the area of the nose. Detection of control points has satisfactory results, but there are based on results of the features.

6 Acknowledgement

I would like to thank Ivana Kolingerová for her support and excellent leadership in this project and to Petr Martínek for his help and consultations.

References

- [1] K. Kubásková, *Feature recognition on triangulated models of human head*. Pilsen, 2015. Bachelor thesis, Faculty of Applied Sciences, University of West Bohemia (In Czech).
- [2] L. Karlíček, *Feature recognition on triangulated models*. Pilsen, 2014. Master thesis, Faculty of Applied Sciences, University of West Bohemia (In Czech).
- [3] A. Hubeli, K. Meyer, M. Gross, *Mesh Edge Detection*. CS Technical Report. ETH: Institute of Scientific Computing, vcarso, 2000.
- [4] Ch. Rössl, L. Kobbelt, H.-P. Seidel, *Extraction of Feature Lines on Triangulated Surface Using Morphological Operators*. Smart Graphics, AAAI Technical Report SS-00-04, 2000.
- [5] A. Nealen, T. Igarishi, O. Sorkine, M. Alexa, *Laplacian Mesh Optimization*. GRAPHITE '06, p. 381-389, ISBN:1-59593-564-9, 2006.
- [6] O. Sorkine, *Laplacian Mesh Processing*. The Eurographics Association, p.53-70, 2005.
- [7] Y. Ohtake, E. Belyaev, H.-P. Seidel, *Ridge-Valley Lines on Meshes via Implicit Surface Fitting*. Journal ACM Transactions on Graphics, 2004, vol. 23, no. 3, p. 609-612.
- [8] S. Yoshizawa, A. Belyaev, H.-P. Seidel, *Fast and Robust Detection of Crest Lines on Meshes*. ACM Symposium on Solid and Physical Modeling, 2005, p.227-232, ISBN: 1-59593-015-9.
- [9] S.-K. Kim, Ch.-H. Kim, *Finding ridges and valleys in a discrete surface using a modified MLS approximation*. Computer-Aided Design, 2006, vol. 38, no. 2, p. 173-180.
- [10] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, *Discrete Differential Geometry Operators for Triangulated 2-Manifolds*. Visualization and Mathematics III, 2003, p. 35-57, ISBN: 978-3-662-05105-4.
- [11] H.S. Kim, H.K. Choi, K.H. Lee, *Feature Detection of Triangular Meshes Based on Tensor Voting Theory*. Computer Aided Design, 2009, vol. 41, no. 1, p. 47-58.
- [12] M. Raussen, *Elementary Differential Geometry: Curves and Surfaces*. Department of mathematical sciences, 2008, Aalborg University, Denmark.

Segmentation of Brain Tumors from Magnetic Resonance Images using Adaptive Thresholding and Graph Cut Algorithm.

Zuzana Bobotová*

Supervised by: Ing. Wanda Benešová, PhD.†

Institute of Applied Informatics
Faculty of Informatics and Information Technologies
Slovak University of Technology
Bratislava / Slovakia

Abstract

Development of methods for automatic brain tumor segmentation remains one of the most challenging tasks in processing of medical data. Exact segmentation could improve the diagnostics, as for example the time evaluation of the tumor volume. However, manual segmentation in magnetic resonance data is a time-consuming task. We present a method of automatic tumor segmentation in magnetic resonance images which consists of several steps. In the first step, a high intense cranium is removed from the image. In the next step, the histogram parameters of the image are analyzed using the method *Mixture of Gaussians*. These parameters control the morphological reconstruction (proposed by Luc Vincent 1993). The morphological reconstruction is followed by subtraction and thresholding. It produces a binary mask which is used in the last step of the segmentation: graph cut segmentation. First results of this method are presented in this paper.

Keywords: segmentation, brain tumor, Magnetic Resonance Imaging, morphological reconstruction, adaptive thresholding, graph cut algorithm, Mixture of Gaussians

1 Introduction

Medicine and diagnostics work with a large amount of visual data. Computer vision methods and image processing can help doctors with analysis. Hence, the doctors save their time and can focus on other important tasks.

Medical examination includes tests like MRI - Magnetic Resonance Imaging, CT - Computed Tomography, PET - Positron Emission Tomography, X-ray scans and other less known techniques. Test results can be represented by a single scan or by series of images. Then doctors analyze images and search for anomalies, damages or symptoms of the disease. The goal of the research is to replace a manual or semi-automatic analysis by the automatic processing using methods of computer vision. Nowadays, com-

puter vision segmentation methods are used in the analysis of subset of cells, organs or whole systems from the scans.

The aim of our work is to find an appropriate automatic method to segment brain tumors from magnetic resonance images (MRI). Output of a 3D MRI scan is a sequence of images called slices. These MRI data are stored in special medical formats such as *NIFTI* or *MHA*. Our goal is to segment the tumor from the 2D image (one slice) automatically. The presumption of the proposed method is that in the processed image of the brain is a tumor is included.

It is a very interesting area of research, because it is necessary to solve several problems. MR images are scanned with different contrasts characteristics. In addition to tissue density, tissue relaxation properties contribute to image contrast in MR images. Basic relaxations are T1 and T2. Next challenge is to deal with different sizes, shapes and intensity levels of tumors on the images. Intensity levels of the tumor depend on the aggressiveness of the tumor. Aggressive tumors are less intensive and they can blend with other brain material. The edges of such tumors are not clear.

2 Related works

Many computer vision segmentation methods have been developed during the last years. The article by Gordillo et al. [2] and also the article by Liu et al. [5] list the most suitable methods for medical imaging and brain tumor segmentation: global and local thresholding, region-based methods such as region-growing and watershed algorithm, pixel classification methods and clustering such as Fuzzy C-Means, k-means, Markov Random Fields, Bayes method and Artificial Neural Networks. Some algorithms implement these methods with various types of improvements.

Menzein et al. presented article [7] about Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). Twenty tumor segmentation algorithms were applied to a set of 65 multi-contrast MR images. In the research were implemented several segmentation methods and most of them were automatic. All methods were tested on the

*bobotova.zuzanka@gmail.com

†benesova@fiit.stuba.sk

same dataset. Results of their tests show that different algorithms worked best for different sub-regions. Successes of methods ranged between 74% and 85%. They found that no single method performed best for all regions.

Khotanolou et al. presented automatic segmentation algorithm [4] to detect brain tumor in 3D MRI data. In first phase, initial tumor segment is detected using histogram analysis, morphological operations and symmetry analysis. Then the tumor is detected using fuzzy classification and symmetry analysis again. Their results show that method is effective and suitable for brain tumor detection.

Prastawa et al. presented framework [9] for automatic brain tumor segmentation based on outlier detection. At first, abnormalities were detected using information about intensities. Secondary, tumor and edema presence is verified. Finally the spatial and geometric properties are used for determining proper sample locations. Method was tested on three datasets.

Havaei et al. presented article [3] about brain tumor segmentation method last year. They implemented deep neural networks with two different types of architectures. First type was two pathway architecture made from two streams. It allowed follow two aspects - visual details of the region around that pixel and where the patch is in the brain. Secondary three types of cascade architecture were implemented. Results of the methods are very promising.

Another work [8] from the last year published Prajapati and Jadhav. They utilized the following steps in their method for brain tumor segmentation from MR images: morphological operations, thresholding and region growing segmentation. Results of their tests show that region growing method is suitable for brain tumor detection.

3 Algorithm overview

MR images in our dataset differ in the space resolution and also in the intensity resolution. Hence, tumor segmentation must reckon with several problems. Intensities of tumors on the images are different according to tumor aggressiveness. Tumors have various shapes and localization and vary in sizes. Bigger tumors are not problematic for segmentation, but some tumors are very small and their intensities, sizes and shapes are very similar to other healthy brain parts. In our research, we develop an automatic method to solve the problems listed above.

The method consists of three steps as shown in Figure 1. First, the contrast is enhanced by image rescaling. In the second step, the cranium (skull) should be removed from the image. It means bones around the brain mass which protect the brain. The removing of cranium is important for the further processing, mainly in cases when it is of high intensity. In fact, the segmentation of a tumor could be confused by cranium, because a tumor has high intensity too. The final step is the tumor segmentation. The result is MR image with indicated boundaries of the tumor counted by two different algorithms.

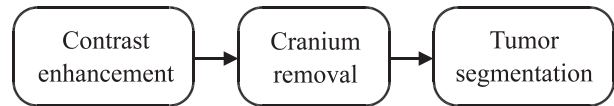


Figure 1: Steps of algorithm

4 Used methods

Several well-known computer vision methods are used in the proposed algorithm: *Mixture of Gaussians*, morphological operations and greyscale morphological reconstruction, thresholding and graph cut algorithm. Chapters 4.1 – 4.5 contain general explanation of these methods and chapters 5.1 – 5.3 explain the order, the reasons and the implementation of the methods.

4.1 Mixture of Gaussians

Gaussian mixture distribution is a multivariate distribution that consists of a mixture of one or more multivariate Gaussian distribution components. The number of components is fixed as input parameter. Each multivariate Gaussian component is defined by its mean and covariance, and the mixture is defined by a vector of mixing proportions.

4.2 Morphological operations

Morphology is the study of shape. Mathematical morphology mostly deals with the mathematical theory of describing shapes using sets. In image processing, mathematical morphology is used to investigate the interaction between an image and a certain chosen structuring element using the basic operations of erosion and dilation [6].

In our work, we use basic operations like open and close as well as more advanced morphological operation. Morphological reconstruction described in the next paragraph.

4.3 Greyscale morphological reconstruction

Greyscale morphological reconstruction is an iterative process. Input for the algorithm is mask image. Actually mask image is the processed image. Algorithm also needs marker image as shown in Figure 2. Greyscale morphological reconstruction is described in detail in the book by Šikudová et al. [12].

In basic morphological reconstruction binary dilation or erosion is applied for the marker image. Then the algorithm calculates the intersection with mask image. The processing continues until the mask image values stop changing.

Greyscale morphological reconstruction is based on similar principles. However, binary dilation or erosion is replaced with greyscale dilation or erosion and intersection is replaced with the selection of the minimum value among the sets of points.

Method is usually used for removing local maximas of the image. However, it is important to extract local maximas not remove them in some cases. Hence, reconstructed image is subtracted from the input image as shown in Figure 2.

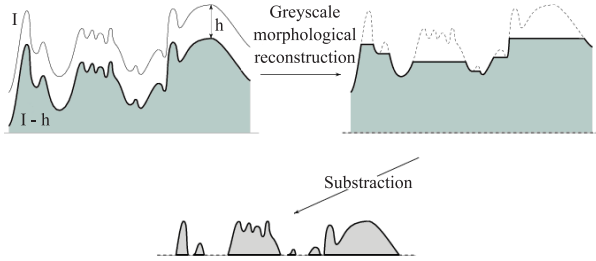


Figure 2: Greyscale morphological reconstruction followed by subtraction [12].

4.4 Thresholding

Thresholding is one of the simplest segmentation methods. Basic method exchanges each pixel $P_{i,j}$ in the image for black or white pixel according to intensity I of the pixel [11]. Thresholding input value is fixed constant called threshold. If $P'_{i,j}$ is thresholded version of $P_{i,j}$ according to intensity $I(P_{i,j})$ and T is threshold then:

$$P'_{i,j} = \begin{cases} 1 & \text{if } I(P_{i,j}) \geq T \\ 0 & \text{otherwise} \end{cases}$$

In the medicine, segmentation by thresholding often fails, because medical images have very complex distribution of intensities [1]. However, thresholding methods are often followed by other segmentation methods or combined with other methods. Threshold in our method has been derived using *Mixture of Gaussians* method.

4.5 Graph cut segmentation

Graph partitioning methods are efficient for the segmentation. They model the image like a weighted graph as explained in [13]. In this algorithm pixels are associated with nodes. Connections between them create weighted edges. Values of the weights depend on similarities or dissimilarities between neighboring pixels. The graph cut is a way how to partition one graph into two regions according to some characteristics. Edges created between two partitions of the graph are called cut edges. They have weights depending on the weight values of edges between pixels. Resulting weight of the cut is the sum of the weights of the cut edges. Finally, the result is a set of partitions and every partition is a segment of the image.

There are many partitioning methods. One of them is *GrabCut* algorithm from OpenCV library. It was designed by Rother et al. and described in the article [10].

Originally the algorithm needs user interaction to draw the input rectangle around the foreground region. The algorithm iteratively segments the foreground using Gaussian Mixture Model. The resulting distribution of pixels is used to build the graph. Nodes in the graph are pixels and two next nodes are added, source node as S and sink node as T . Each pixel in the foreground is connected to the S node and each pixel in background is connected to the T node. The weights of edges which connect pixels to the S or T node are defined by the probability that a pixel is in the foreground or in the background. The weights between neighboring pixels are defined by the pixel similarity. The min-cut algorithm is used to divide the graph. It finds the minimum cut of the weighted graph. Finally, pixels connected to the S node become foreground and pixels connected to the T node become background.

5 Implementation

5.1 Contrast enhancement

The input for the method is an image from magnetic resonance. Background of the image is typically black and tumors have high intensity. However, data are scanned with various settings which causes differences of the intensities. It means that on some images background is not black and tumors are not so intense. For this reason, rescaling is the first step of the proposed method. It is helpful for future processing, because images have similar characteristics. Hence, image is rescaled into the range from 0 to 255. Figure 3 shows input MRI image and the rescaled image.

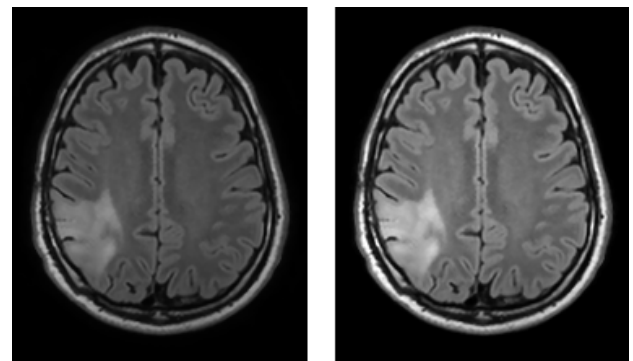


Figure 3: Input MR image (left) and the rescaled image (right).

The scaling results from the minimal and maximal value of the image. Actually the real minimal and maximal values of the image could be just some single casual pixels which are not relevant values for the rescaling. For this reason, a statistical method is used to count the minimal and maximal values used in the rescaling process. Hence, the values of percentile 35 and percentile 99.9 are calculated. This is because the typical large black background of MR images area covers at least 35% of the image and

the relevant high intensity area (cranium) covers at least 0.1% of the image.

5.2 Cranium removal

Before the segmentation, the cranium has to be removed from the image. Mainly in cases when cranium pixels have comparable values to the ones of the tumors. High intense cranium cause errors in the segmentation. Some examples are shown in Figure 4.

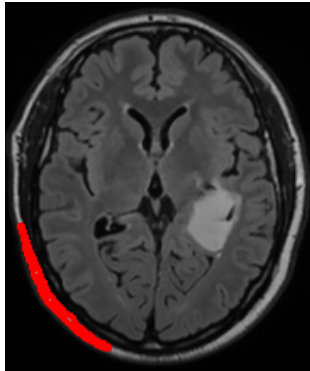


Figure 4: Example of the error segmentation caused by intensive cranium.

Cranium is removed depending on the mask created during the step. Mask creation is derived from the results of statistical method called *Mixture of Gaussians* and adaptive thresholding.

First, *Mixture of Gaussians* is done using distribution of three Gaussians. Three values are the result of the method. One represents black background and two following values represent brain pixels. Histogram of the image with Gaussian distribution are shown in Figure 5. Resulting values are used as input parameter for the next step binary thresholding.

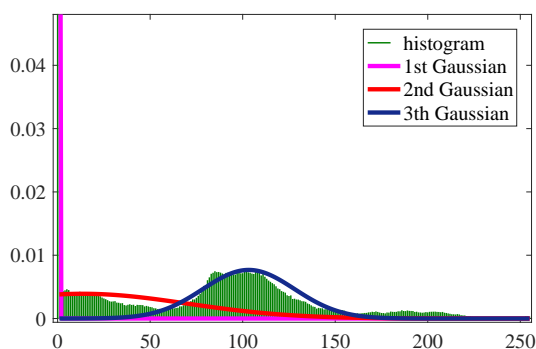


Figure 5: Histogram approximation using *Mixture of Gaussians*.

Parameter called threshold is usually fixed constant, but thanks to the results of *Mixture of Gaussians* method our thresholding is adaptive. The result of thresholding is a

mask which marks brain with cranium. It is shown in the Figure 6. This is an important step for the correct removal of the cranium. Morphological operations erode and dilate are used after the thresholding to remove small seeds from the mask. Results of morphological operation are visible in Figure 7.



Figure 6: Brain mask created after thresholding.

Next step of the algorithm is graph cut segmentation. The *GrabCut* method from the OpenCV library is used. The method needs input rectangle or a mask which represents foreground. The rest is background. In this step rectangle is used as foreground initialization to prevent removal of the brain mass parts (brain without the cranium). Initial rectangle is created depending on the previous thresholding as shown in Figure 7. *GrabCut* is explained in chapter 4.5. Result is contour which border brain and also cranium.



Figure 7: Initial rectangle for *GrabCut*.

Thickness of the contour is enlarged and used as mask for cranium removal. The mask is shown in Figure 8. The result is the image with removed cranium as shown in the Figure 9.

Our testing dataset contains various MR images. Actually some of them have low and other high intense cranium. Whole images are processed in cranium removal step. Sometimes cranium is not entirely removed and intensive remnants cause the problem for the segmentation. On the other side, parts of the brain are removed in some

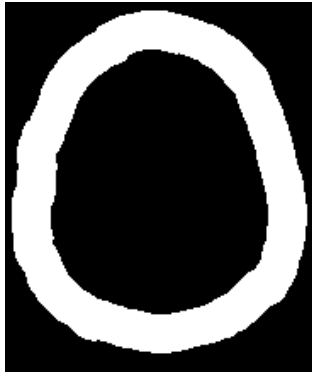


Figure 8: Mask for cranium removal.

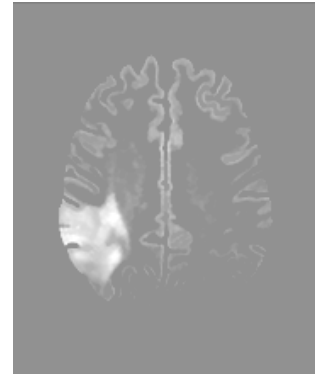


Figure 10: Result of the morphological reconstruction.

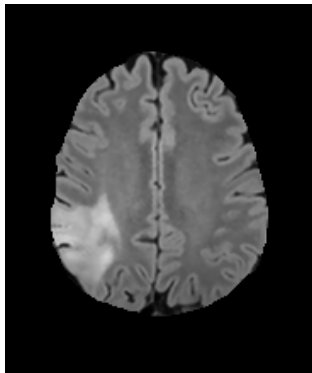


Figure 9: Brain with removed cranium.

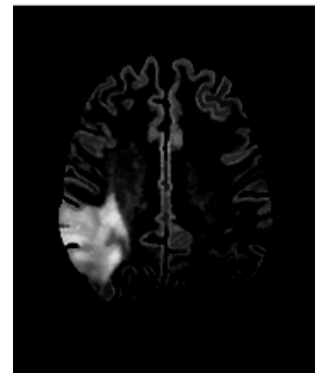


Figure 11: Result of the subtraction.

specific cases. However, the second problem does not cause errors in tumor segmentation, so it is not a priority to solve.

5.3 Tumor segmentation

Last step is the tumor segmentation. The same statistical method *Mixture of Gaussians* is done using distribution of three Gaussians again. From three resulting values, one represents black background, the second one represents high intensive brain parts and last one is the rest of the brain. Values are used as parameters for the future processing.

Next step is greyscale morphological reconstruction. It is implemented in OpenCV library. The method needs two parameters. Input image called mask image and a subtraction constant. In our algorithm constant depends on the results of *Mixture of Gaussians* method. Constant is subtracted from the input image and marker image is created. It is explained in the chapter 4.3 in detail. The method removes local maximas from the image. Result is shown in Figure 10. However, for our algorithm are important these maximal values. Hence, resulting image is subtracted from the input image. The result of the subtraction is shown in Figure 11.

The next step is binary adaptive thresholding. Constant for the method depends on the *Mixture of Gaussians* again.

The result is a mask which marks the most intense brain regions as shown in Figure 12. The biggest region is marked as tumor. If there are a lot of small regions then the most intensive is marked as tumor. Then morphological operations erode and dilate remove small seeds from the mask. The result is a mask (Figure 13) which limits tumor.



Figure 12: Mask of the most intense regions of the tumor.

Finally, graph cut algorithm detects boundaries of the tumor. In that case input for the *GrabCut* method is a mask. Mask marks foreground, probably background and background. Foreground is determined by the mask created after adaptive thresholding. It is because changes of the boundaries should not be large. Probably background



Figure 13: Mask of the tumor.

is a rectangle created from the same mask and the rest is background. Mask for *GrabCut* is shown in the Figure 14.



Figure 14: Input mask for the graph cut segmentation.

The last step in our method is the graph cut segmentation optional with the expectation to improve the quality of the segmentation. Visualization of the resulting contours with and without graph cut algorithm is shown in Figure 15. Another segmentations are shown in Figure 16.

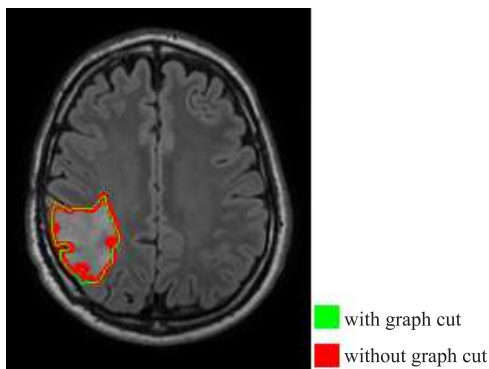


Figure 15: Vizualization of tumor segmentation.

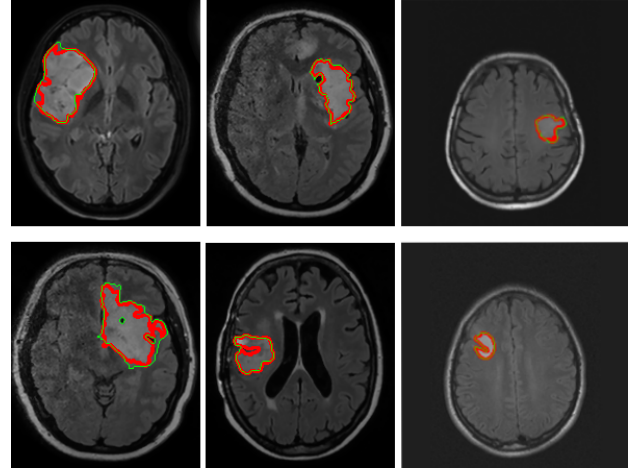


Figure 16: Examples of segmentation.

5.4 Implementation details

MATLAB is used to find the statistical minimum and maximum for contrast enhancement and also for the Mixture of Gaussians method. Output of the previous methods is a file with important statistical values used for the thresholding and morphological reconstruction. These two mentioned methods and the graph cut algorithm are implemented using the C++ programming language and the OpenCv library which includes all important methods of computer vision.

6 Results

Algorithm was tested on real MRI data gained of anonymous patients acquired in clinical practice. Magnetic resonance images came from various apparatus and were scanned with various settings, so they have different intensities.

The images for our dataset were selected from 3D MRI data witch were scanned with T1 relaxation. For the evaluation of our method, we have used 150 randomly selected 2D images with various measurements and intensities, which include tumors of different areas, shapes and locations.

Tumor segmentation was tested by two ways to detect advantages and disadvantages of proposed algorithm. First, it was tested with the algorithm which consists only of adaptive greyscale morphological reconstruction and adaptive thresholding without the graph cut algorithm. Second, it was tested also with the graph cut algorithm.

Algorithm results were compared with manual segmentations of tumors provided by experts. Verification was based on the per pixel comparison of the segmentation results and manual segmentations. Resulting segmentation was transformed to binary image. It is because manual segmentation was also saved as binary image.

Resulting tumor segmentation was divided on true pos-

itive (TP), true negative (TN), false positive (FP) and false negative (FN) regions. TP represents pixels where tumor was detected and should be. TN means that tumor was not detected and should not be. FP is when tumor was detected and should not be. Finally if tumor was not detected, but should be, it is FN. Figure 17 is visualization of pixel division. Statistical methods were used to evaluate results:

- true positive rate – sensitivity (TPR):

$$TPR = \frac{TP}{TP + FN}$$

- true negative rate – specificity (TNR):

$$TNR = \frac{TN}{TN + FP}$$

- predictive value positive – precision (PVP):

$$PVP = \frac{TP}{TP + FP}$$

- accuracy (A):

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

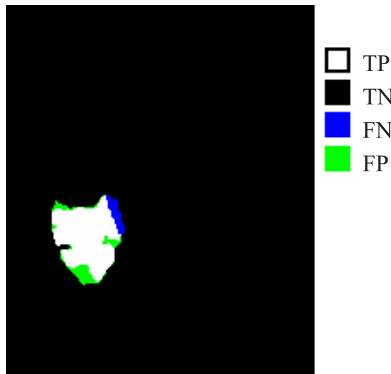


Figure 17: Vizualization of per-pixel division.

Success of the algorithm without graph cut segmentation is presented in the Table 1. In the Table 2 is presented success of algorithm with graph cut segmentation.

TPR	TNR	PVP	A
80.91%	99.75%	83.94%	99.28%

Table 1: Results of testing without graph cut algorithm.

Segmentation without graph cut algorithm reached 99.28% accuracy and true positive rate was 80.91%. Segmentation with graph cut algorithm failed in several cases. It means that no tumor boundaries have been found. It failed in 23 of 150 images what is 14.77%. Correctness

TPR	TNR	PVP	A
82.12%	99.63%	86.4%	99.24%

Graph cut failed in 14.77% of samples. Therefore, only the successful segmentations of the graph cut are presented in the table.

Table 2: Results of testing with graph cut algorithm.

and statistical results was evaluated only on the images where algorithm worked. Accuracy with graph cut algorithm was 99.24% and true positive rate was 82.12%.

Advantages of proposed algorithm lie in the ability to handle various data. It can evaluate MR image with various intensities using adaptive methods which depend on statistical intensity values of the image. Adaptive greyscale morphological reconstruction and adaptive thresholding are crucial for successful segmentation and correct localization of the tumor. Graph cut segmentation was also tested to segment the tumors. It increased the precision and accuracy of tumor detection in specific cases. However, the results using graph cut segmentation method are less successful comparing with the method which was done without the graph cut in summary. Boundaries of the tumor are not always clear. It is problematic for graph cut algorithm and causes that sizes of segmentations were bigger than real tumor.

In some specific cases, tumors were not located because they do not have the largest intensity so other more intensive areas were detected as tumor. The most errors were caused by the images which contained eyes or remnants of the cranium. Examples of the errors caused by the eye are shown in Figure 18. Solve those errors were not our priority so images with eyes were removed from our testing dataset. Necessarity of cranium removal is explained in chapter 5.2. Sometimes cranium is not removed and then it causes problems as is shown in Figure 4. Actually, small and very intensive areas should be filtered, but then small tumors can be lost.

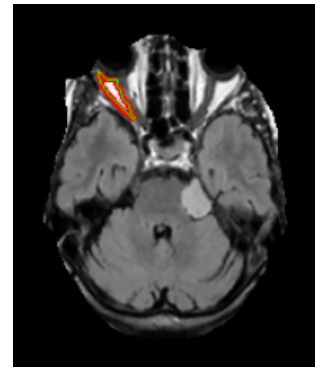


Figure 18: Example of the segmentation errors caused by the eyes.

As was mentioned, algorithm is divided into several

smaller steps and implemented in MATLAB and C++. Table 3 shows average time measurements of individual parts of code for one processed image. Time was counted in seconds.

Contrast enhancement		0.005 s
Cranium removal	Mixture of Gaussians	1.761 s
	Segmentation and removal	0.129 s
Tumor segmentation	Mixture of Gaussians	1.189 s
	Segmentation	0.125 s

Table 3: Average time measurements.

7 Conclusions and Future work

In this paper, we have presented automatic algorithm for the segmentation of brain tumors from magnetic resonance images. The main advantage of the presented algorithm is its robustness. It is designed with the goal to process images from various devices for the MRI data acquisition and with various intensities. It becomes possible using the adaptive thresholding and greyscale morphological reconstruction which get parameters according to the results of statistical method *Mixture of Gaussians*. It is followed by the graph cut algorithm. Adaptive thresholding and greyscale morphological reconstruction are crucial for the correct results. The graph cut algorithm increases the precision of the segmentation in some specific cases, but in summary the results using graph cut segmentation method are less successful comparing with the method which was done without the graph cut.

In future work we would like to solve problems with specific cases where sizes and intensities of tumors are problematic. We would like to extend method to segment tumor automatically from 3D MRI data not only 2D images. Then the algorithm will be tested on bigger dataset consists of many 3D MRI data of brains with tumors and compared with another methods.

Acknowledgement: This work was supported by the Grant VEGA 1/0625/14 and Siemens Healthcare Bratislava.

Source code is available: <http://vgg.fkit.stuba.sk/2016-04/segmentation-of-brain-tumors-from-mri-using-adaptive-thresholding-and-graph-cut-algorithm/>

References

- [1] M. A. Balafar, A. R. Ramli, M. I. Saripan, and S. Mashohor. Review of brain MRI image segmentation methods. *Artificial Intelligence Review*, 33(3):261–274, 2010.
- [2] N. Gordillo, E. Montseny, and P. Sobrevilla. State of the art survey on MRI brain tumor segmentation. *Magnetic Resonance Imaging*, 31(8):1426–1438, 2013.
- [3] M. Havaei, A. Davy, and D. Warde-Farley. Brain Tumor Segmentation with Deep Neural Networks. *arXiv*, page 13, 2015.
- [4] H. Khotanlou, O. Colliot, J. Atif, and I. Bloch. 3D brain tumor segmentation in MRI using fuzzy classification, symmetry analysis and spatially constrained deformable models. *Fuzzy Sets and Systems*, 160(10):1457–1473, 2009.
- [5] J. Liu, M. Li, J. Wang, F. Wu, T. Liu, and Y. Pan. A Survey of MRI-Based Brain Tumor Segmentation Methods. *Tsinghua Science and Technology*, 19(6):578–595, 2014.
- [6] T. Maintz. Digital and Medical Image Processing. page 341, 2005.
- [7] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-cramer, K. Farahani, J. Kirby, Y. Burren, et al. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE*, 34(10):1–32, 2014.
- [8] S. J. Prajapati and K. R. Jadhav. Brain Tumor Detection By Various Image Segmentation Techniques With Introduction To Non Negative Matrix Factorization. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(3):599–603, 2015.
- [9] M. Prastawa, E. Bullitt, S. Ho, and G. Gerig. A brain tumor segmentation framework based on outlier detection. *Medical image analysis*, 8:275–283, 2004.
- [10] C. Rother, V. Kolmogorov, and A. Blake. "Grab-Cut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.
- [11] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004.
- [12] E. Šikudová, Z. Černeková, W. Benešová, Z. Haladová, and J. Kučerová. *Počítačové videnie*. Wikina, Praha, 1 edition, 2011.
- [13] A. S. Torres and F. C. Monteiro. Image segmentation by graph partitioning. *IPB Campus Santa Apolónia*, 805:802–805, 2012.

Classification of built-up areas in LiDAR data based on second-generation connectivity filters

Robi Cvirn*

Supervised by: Domen Mongus[†]

University of Maribor / Slovenia

Abstract

In the recent years, Light Detection and Ranging (LiDAR) technology has become one of the prime technologies for spatial data acquisition. By rapid and accurate capturing of high-resolution 3D point-clouds, LiDAR has moved the focus of further research in Earth Observations towards data processing. In this paper, we propose a new approach for classification of built-up areas from LiDAR data. The methodology is based on a special class of morphological filters that rely on so-called second-generation connectivity. We first provide theoretical background on connected operators and explain how they can be applied for LiDAR data processing. Finally, we validate the proposed approach on several testcases.

Keywords: LiDAR, mathematical morphology, second-generation connectivity, object recognition, algorithms

1 Introduction

Light Detection and Ranging (LiDAR) technology has become one of the prime technologies for acquiring high-resolution spatial data. By rapidly capturing accurate 3D point-clouds of the Earth's surface, it allows for monitoring structures and processes with great precision over vast geographic areas. Over the past decade, a lot of research has been directed towards object recognition in LiDAR data and efficient methods for extracting ground [9, 10], buildings [8, 13, 20], vegetation [5], and even single tree-crowns [4, 11] have already been developed. The focus of further research is now shifting towards situation assessment, where recognized objects are taken into account in order to establish a wider sense of the current situation. A particularly important example of situation assessment is a classification of built-up areas, as it is critical for many studies of impacts that human developments have on the natural process [7].

In this paper, a new method for the classification of built-up areas is presented. In contrast to the related work, the proposed method has a different approach by exploiting the height information present within the LiDAR

data. This is achieved by connected operators from the framework of mathematical morphology that are based on second-generation connectivity.

The related work is described in Section 2. The relevant theoretical background is given in Section 3. Section 4 describes the data structuring needed in order to apply connected operators for LiDAR data processing together with the proposed method. The results are discussed in Section 5. Section 6 concludes the paper.

2 Related work

The classification of built-up areas has up to now been mostly achieved based on human intuition by considering the actual size of the area (and its population) in addition to the services that a given area offers (e.g. by the presence of education and medical institutions, train stops, or sports buildings) [1]. Nevertheless, several approaches have already been developed that rely on satellite images for delivering quantitative measurement of built-up areas for their classification. Pesaresi et. al [17] proposed a method for calculating built-up presence index from panchromatic satellite images. Tomowski et. al [25] developed a settlement area detection based on panchromatic and multispectral data, while Najab et. al [12] introduced a classification of settlements based on holistic feature extraction technique using high resolution satellite images. Van den Bergh [26] proposed a method for classification of settlements based in their illumination geometry in QuickBird images.

3 Theoretical background

This section describes theoretical basics of connected operators within the context of mathematical morphology. Let a universal nonempty set $E \subseteq \mathbb{R}^d$ (i.e. a definition domain), define a d -dimensional dataset (e.g. a binary 2D image, 3D voxel space, or any higher dimensional discrete set) is defined by the means of nonempty set $S \subseteq E$, where $s_i \in S$ is a d -dimensional point. A power-set (i.e. a set of all subsets of set S) is denoted as $\mathcal{P}(S)$, while the connectivity between the elements of S is given by the means of

*robi.cvirn@um.si

[†]domen.mongus@um.si

connectivity class $\mathcal{C} \subseteq \mathcal{P}(S)$ [15, 22, 23, 24]. For this purpose, \mathcal{C} has to satisfy the following two conditions[16]:

- $\emptyset \in \mathcal{C}$ and $\forall s_i \in S, s_i \in \mathcal{C}$ and
- $\forall \{C_i\} \subseteq \mathcal{C}, \bigcap_i C_i \neq \emptyset \Rightarrow \bigcup_i C_i \in \mathcal{C}$.

In simple terms, this means that an empty set, any given singleton, and any set of subsets $C_i \subseteq S$ with non-empty intersection is connected. Given $s_i \in S$, the corresponding connected component of a set S can be extracted by the connectivity opening Γ . Its formal definition is given as [16]:

$$\Gamma_{s_i}(S) = \bigcup_j \{C_j \in \mathcal{C} \mid s_i \in C_j \text{ and } C_j \subseteq S\}. \quad (1)$$

In order to simplify the notation, we refer to the connected component addressed by a point s_i as $C_i = \Gamma_{s_i}(S)$.

3.1 Connected operators

Connected operators within the framework of mathematical morphology are edge preserving operators that act directly on connected components [21]. Attribute filters are well-known examples of connected operators. They allow for filtering connected components according to their attributes. Let attribute function of a connected component Λ (e.g. its area, perimeter or width) and the corresponding attribute threshold λ , a binary attribute opening of an arbitrary set S is given as [14]:

$$\Gamma_{(\Lambda, \lambda)}(S) = \bigcup_{s_i \in S} \{\Gamma_{(\Lambda, \lambda)}(\Gamma_{s_i}(S))\}, \quad (2)$$

where attribute opening $\Gamma_{(\Lambda, \lambda)}(S)$ removes those connected components of S that do not satisfy a given attribute criteria $\Lambda(C_i) \geq \lambda$ (i.e. $\Gamma_{(\Lambda, \lambda)}(S) = \{C_i \mid \Lambda(C_i) \geq \lambda\}$).

3.2 Second-generation connectivity

Each connectivity class \mathcal{C} can be evolved to its child class with curtailed or augmented members. This concept is known as second-generation connectivity and it allows for manipulation over the connected components that can not be achieved by regular connected operators. Two types of second-generation connectivity exist, namely contraction-based connectivity that allows for braking the connected components and clustering-based connectivity that allows for merging them [3]. Due to the specifics of the proposed method, (see Section 4), only the latter is considered in this paper.

A cluster can be described as a set of connected components for which the mutual distances between them are smaller than a certain distance criteria. In order to achieve clustering, a clustering operator ψ needs to be applied that complies to the following restrictions [19, 23, 3]:

- ψ needs to be extensive and increasing,

- the resulting connectivity class has to be reduced or equal to the input connectivity class $\psi(\mathcal{C}) \subseteq \mathcal{C}$, and
- for all subsets of a set $\{S_i\} \in \mathcal{P}(S)$ it is required that $\forall i, \psi(S_i) \in \mathcal{C}$, and $\bigcap_i S_i \neq \emptyset \Rightarrow \psi(\bigcup_i S_i) \in \mathcal{C}$.

Let ψ be a clustering operator on $\mathcal{P}(S)$ and \mathcal{C} a connectivity class in $\mathcal{P}(S)$, a clustering-based connectivity class $\mathcal{C}^\psi \supseteq \mathcal{C}$ within the definition domain E can then be defined as:

$$\mathcal{C}^\psi = \{S \in \mathcal{P}(E) \mid \psi(S) \in \mathcal{C}\}. \quad (3)$$

By only redefining the elementary connectivity opening, this allows for applying second-generation connectivity together with any of the connected operators without changing their original definitions. A second-generation connectivity opening $\Gamma_{(\Lambda, \lambda)}^\psi(S)$, defined in regards to an arbitrary attribute function Λ with attribute threshold λ is given as [15]:

$$\Gamma_{(\Lambda, \lambda)}^\psi(S) = \Gamma_{(\Lambda, \lambda)}(\psi(S)) \cap S. \quad (4)$$

Finally, the results of the second-generation connectivity opening $\Gamma_{(\Lambda, \lambda)}^\psi(S)$ applied on a binary image S are shown in Fig. 1.

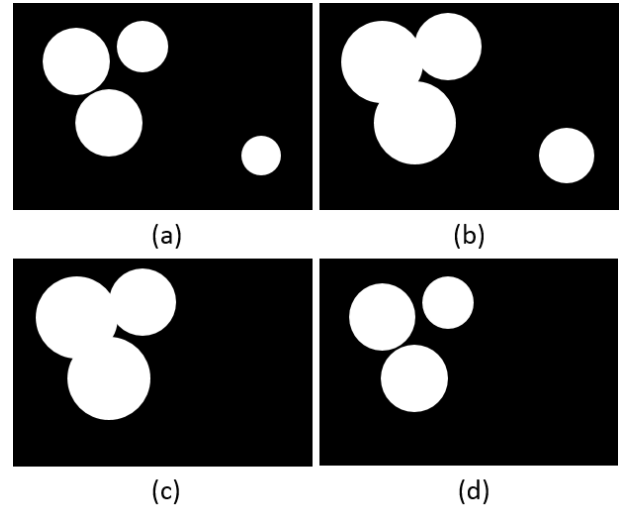


Figure 1: Second generation attribute opening, where (a) the original image S is (b) extended by a clustering operator $\psi(S)$ and (c) attribute opening is applied $\Gamma_{(\Lambda, \lambda)}(\psi(S))$ in order to obtain (d) the resulting set $\Gamma_{(\Lambda, \lambda)}^\psi(S)$.

4 Implementation of connected operators on LiDAR data

We denote an input LiDAR point-cloud as $P = \{P_i\}$ where p_i is an individual LiDAR point. Each p_i is associated with a coordinate triple, given as $x(p_i)$, $y(p_i)$, and $z(p_i)$. Note

that connected operators, as described in previous section, can not be directly applied on P , due to the lack of its topological structure. In order to overcome this issue, we construct a grid $G \subset \mathbb{R}^2$ of resolution r (usually $r = 0.5m$ is sufficient) and the extent that is equal to the bounding-box of P . A grid-cell is denoted as $g_i \in G$, while a set of LiDAR points that are closer to a given g_i than to any other $g_j \in G$ are denoted as $P_{g_i} \subseteq P$. In continuation, we demonstrate the efficiency of the proposed formulation for the classification of built-up areas.

4.1 Classification of built-up areas

The main objective of this method is to classify the built-up areas according to their sizes and heights of the contained buildings. The method uses preprocessed LiDAR data, where points belonging to buildings are already classified. In our case, were detected by extracting linearly distributed points using Locally Fitted Surfaces and measuring their geometric properties based on differential morphological profiles[8]. We can thus define a function $class : P \rightarrow [building, not\ building]$ that returns a classification of a LiDAR point. A set of building grid-cells $S \subseteq G$ is formally defined by Eq. 5, while its meaning is graphically explained in Fig. 2.

$$S = \{g_i \in G \mid \exists p_j \in P_{g_i} \text{ and } class(p_j) = building\}. \quad (5)$$

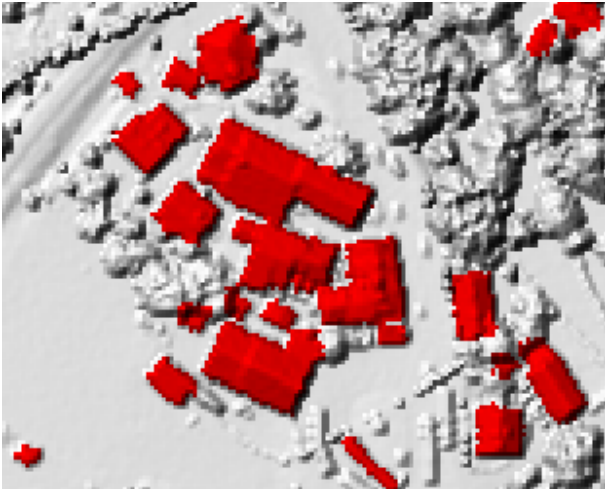


Figure 2: The definition of $S \subseteq G$, according to the buildings (colored red) contained within the input LiDAR dataset.

Each connected component represents a building and is a set of a 8-connected cells (i.e. $C_i \subset S$). In the next step, we apply a clustering-based second-generation connectivity on the set of connected components contained in S . Since spatial distance between buildings within the built-up areas may vary, spatially-variant clustering operator is an optimal choice [2]. As shown by the results (see

Section 5), sufficient accuracy is achieved by using a morphological dilation with the size of the structuring element that is directly proportional to the heights of the buildings contained within the corresponding connected components $C_i \in S$. Let $h : G \rightarrow \mathbb{R}$, be a height function, defined at a given $g_i \in G$ as:

$$h(g_i) = \bigvee_{g_j \in C_i} \{h(p_l) \mid p_l \in P_{g_j}\}, \quad (6)$$

where \bigvee denotes supremum (i.e. maximum). In addition, when $P_{g_i} = \emptyset$, $h = 0$. A clustering operator in a form of morphological dilation with variable window size is then denoted as $\delta^{t^S h}$, where t^S is a user defined parameter that defines the linear relationship between the heights of the buildings and the corresponding size of the structuring element. The results of $\delta^{t^S h}(S)$ are shown in Fig. 3.

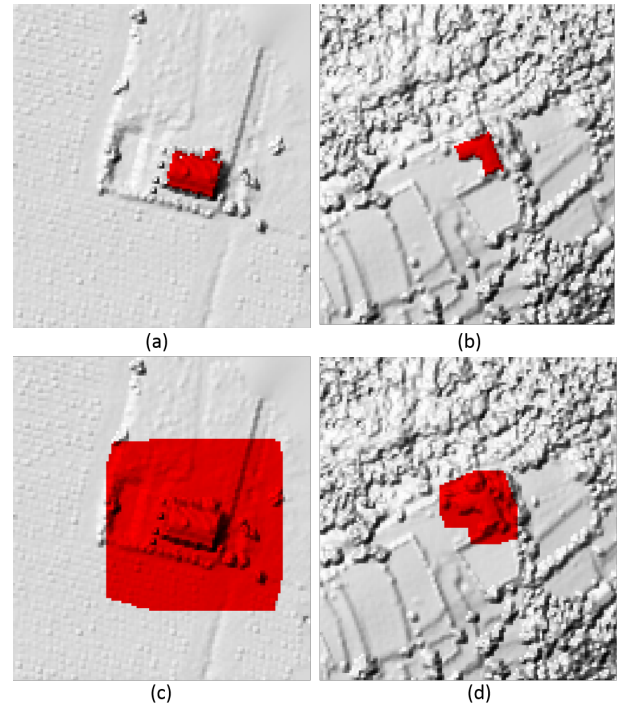


Figure 3: The spatially-variable dilation $\delta^{t^S h}$, applied on (a and b) two connected components of different heights (i.e. 13m and 6m, respectively) and (c and d) the obtained results.

According to Eq. 4, second-generation connectivity opening that removes those connected components C_i with areas $A(C_i) < a$ can now be given as $\Gamma_{(\Lambda, \lambda)}^{\delta^{t^S h}}$. According to the literature, built-up areas may be divided into hamlets, villages, towns and cities [1]. In order to identify the correct type, we apply a series of second-generation connectivity openings at an increasing scale and observe those connected components (i.e. buildings) removed at each of the scales. This concepts is also known as differential attribute profiles [14]. Let $a_1 = 50000m^2$, $a_2 = 1000000m^2$,

and $a_3 = 5000000m^2$ be a set of user defined area thresholds that correspond to the areas of particular built-up types (note that this particular values were intuitively defined, as there is no actual standard definition of the terms [1]). We may define a set of buildings belonging to each of them by the following rules:

- Hamlets $S^H = S \setminus \Gamma_{(A,a_1)}^{\delta^{r^S_h}}(S)$
- Villages $S^V = \Gamma_{(A,a_1)}^{\delta^{r^S_h}}(S) \setminus \Gamma_{(A,a_2)}^{\delta^{r^S_h}}(S)$
- Towns $S^T = \Gamma_{(A,a_2)}^{\delta^{r^S_h}}(S) \setminus \Gamma_{(A,a_3)}^{\delta^{r^S_h}}(S)$, and
- Cities $S^C = \Gamma_{(A,a_3)}^{\delta^{r^S_h}}(S)$.

The obtained results are shown in Fig. 4.

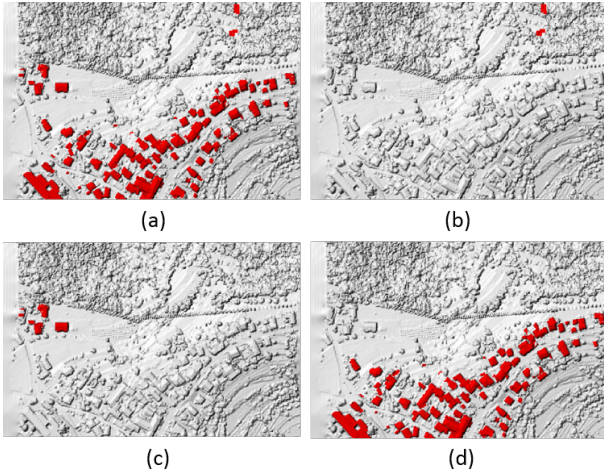


Figure 4: Differential attribute profiles on second-generation connectivity opening, where (a) the original S is divided into (b) hamlets S^H , (c) villages S^V , and (d) towns S^T .

5 Results

The proposed method for classification of built-up areas was tested on four different LiDAR datasets. Each of them contained different types of built-up areas as well as different terrain configurations and was acquired at different data densities. Details about particular test set are given in Table 1.

5.1 Validation procedure

To verify the quality of classification, the obtained results were compared with ground truth data. For this purpose, land cadastral data was rasterized and its actuality was checked by a domain expert. In this way, we were able

Table 1: Test datasets with description.

Dataset name	Description	Terrain type	Data density [points per m^2]
RA	Rural area with hamlets and villages	hilly	7.3
TS	Town and nearby small settlements	valley	8.7
CS	City and its suburb	flat	12.6
CC	Strict city center	flat	5.5

to perform a grid-cell to grid-cell comparison, where true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) were measured for each of the classes. Accordingly, the following measurements of quality were used for validation [18, 6]:

- Completeness, describing the rate of correctly recognized classes, is defined as:

$$comp = \frac{TP}{TP + FN}, \quad (7)$$

- Correctness, describing the rate of correct detection, is defined as:

$$corr = \frac{TP}{TP + FP}, \text{ and} \quad (8)$$

- F1-score that is harmonic mean of completeness and correctness is given by:

$$F1 = 2 * \frac{comp \cdot corr}{comp + corr}. \quad (9)$$

5.2 Sensitivity analysis

As explained in Section 4, the proposed method uses one input parameter t^S that defines the linear relation between the heights of the buildings and the size of the structuring element in spatially-variant dilation. In order to provide its optimal definition, we were progressively increasing the value of t^S , while measuring the success rate of the method. The obtained results are shown in Table 2.

Table 2: Sensitivity analysis of the method in regards to the parameter t^S , presented by F1-score.

Dataset	t^S				
	0.5	1.0	1.5	2.0	3.0
RA	50.5	86.3	96.7	35.9	36.3
TS	10.9	18.8	91.4	96.1	94.5
CS	17.7	33.7	43.4	93.2	54.9
CC	73.4	91.2	94.3	94.2	95.2

As shown by Table 2, the proposed method is sensitive to the value of t^S due to significantly different spacing of buildings within the different types of built-up areas. This indicates that, by defining the size of the clustering operator according to the correspondent height of the building, the issue can only be mitigated but it cannot be solved. Still, the proposed method managed to achieve accuracy of over 93% in all of the cases.

5.3 Validation

A closer look at Table 2 reveals that low t^S values are more appropriate for classifying datasets containing rural or suburb areas, while large t^S values are more suitable for classifying datasets containing city areas. The main reason for this is that hamlets do not require any clustering in order to be successfully recognized, while high buildings in city areas are often widely spaced with green areas or parking lots between them. A more detailed analysis of the method's quality and accuracy was therefore performed using optimal definition of t^S for each particular test dataset. The obtained results are shown in Table 3 and Fig. 5.

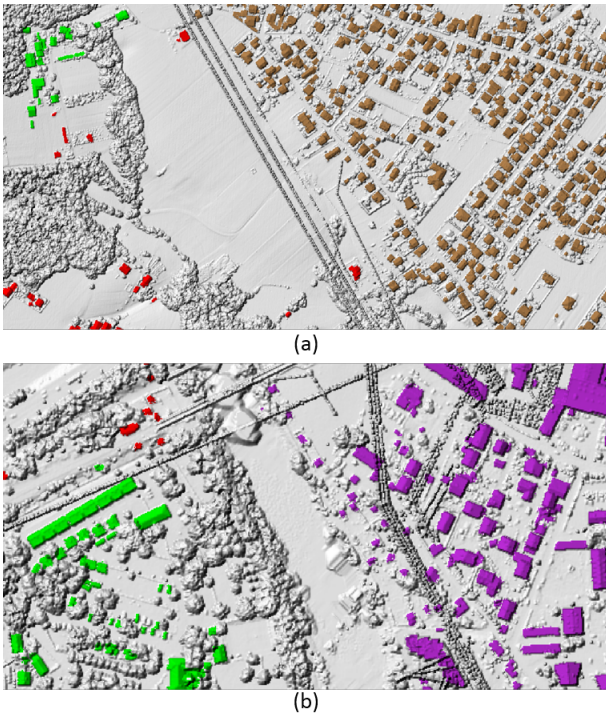


Figure 5: Examples of classified datasets containing (a) a city (colored brown), nearby village (colored green), and hamlets (colored red) and (b) a town (colored purple) with several nearby hamlets.

As shown by Table 3, the proposed method achieves average F1-score of over 87% per all classes, with average completeness as well as correctness close to 90%.

However, large deviations in the results can be noticed when considering hamlets or villages. The main reason for this is that these are relatively small types of settlements, where error in the classification of a single building significantly affects the overall accuracy. Nevertheless, most of the inaccuracies are caused by two characteristic errors. Namely, isolated low buildings within towns (see an example in Fig. 6) and groups of hamlets containing relative tall buildings that become clustered and recognized as a town or a village (see example in Fig. 7). Still as shown in Table 4, the classification of the built-up areas is significantly more efficient than the traditional approach, where building heights are not considered in a clustering criterion. In latter case, the used clustering criterion was based on the area attribute of the regions, while the same tuning procedure as described in Section 5.2 was used. Although the proposed approach achieves higher accuracy, the compared method can also be applied on other types of data, where object heights are not explicitly known (e.g. satellite images [17, 12] or digital orthophoto [26]).

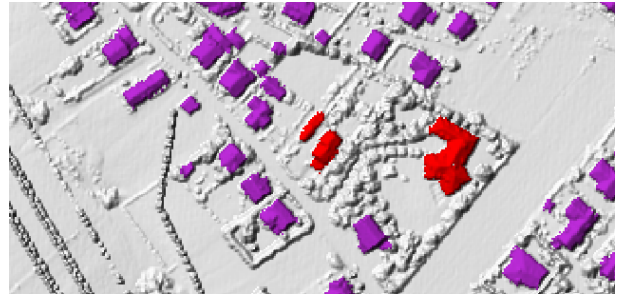


Figure 6: Missclassified low buildings (colored red) within a town (colored purple).

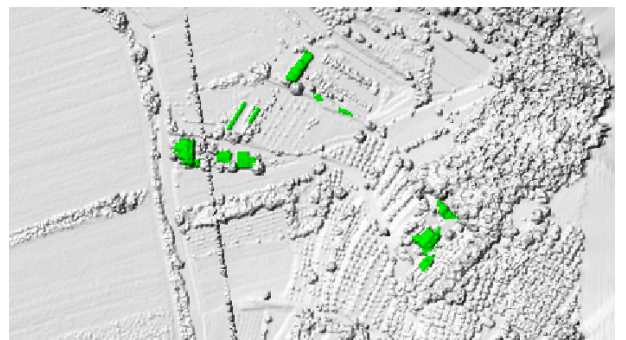


Figure 7: Missclassified hamlets, clustered into a village due to the contained high buildings.

6 Conclusions

This paper proposes a new method for the classification of built-up areas in LiDAR data. The method is based on

Table 3: Validation results of built-up area classification.

Dataset	t^S	Metric	Built-up area types			
			S^H	S^V	S^T	S^C
RA	1.5	comp	100	96.1	not contained	not contained
		corr	82.5	100	not contained	not contained
		F1	90.4	98.0	not contained	not contained
TS	2.0	comp	64.5	97.9	100	not contained
		corr	95.2	88.7	95.0	not contained
		F1	76.9	93.1	97.5	not contained
CS	2.0	comp	96.5	77.8	not contained	98.9
		corr	86.2	87.5	not contained	100
		F1	91.1	82.4	not contained	99.4
CC	3.0	comp	not contained	100	not contained	96.9
		corr	not contained	0	not contained	100
		F1	not contained	0	not contained	98.4
Average	-	comp	87.0	93.0	100	97.9
		corr	88.0	69.1	95.0	100
		F1	86.1	68.4	97.5	98.9

Table 4: Analysis of the method with clustering criterion based on the area attribute of the regions, presented by F1-score.

Dataset	t^S				
	0.01	0.03	0.05	0.10	1.00
RA	26.9	78.8	93.4	32.5	0.0
TS	17.7	86.9	76.6	58.9	0.0
CS	11.0	93.0	92.4	91.8	85.6
CC	87.8	91.6	93.0	94.1	92.0

mathematical morphology, where connected operators of second-generation are used for clustering buildings into groups of built-up areas. As the method uses only one user defined parameter, its performance can simply be optimized, although the method is relatively sensitive to it. Nevertheless, with the average F1-score of 93%, the proposed method proved to be accurate.

References

- [1] M. Aston. *Interpreting the landscape*. B.T.Batsford Ltd, 1985.
- [2] N. Bouaynaya, M. Charif-Chefchaoui, and D. Schonfeld. Theoretical foundations of spatially-variant mathematical morphology part i: Binary images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(5):823–836, May 2008.
- [3] U. Braga-Neto and J. Goutsias. Connectivity on complete lattices: New results. *Computer Vision and Image Understanding*, 85:22–53, 2002.
- [4] L. Eysn, M. Hollaus, E. Lindberg, F. Berger, J. M. Monnet, M. Dalponte, M. Kobal, M. Pellegrini, E. Lingua, D. Mongus, and N. Pfeifer. A benchmark of lidar-based single tree detection methods using heterogeneous forest data from the alpine space. *Forests*, 6(5):1721–1747, May 2015.
- [5] S. Hancocka, J. Armston, Z. Li, R. Gaulton, P. Lewis, M. Disney, F. M. Danson, A. Strahler, C. Schaaf, K. Anderson, and K. J. Gaston. Waveform lidar over vegetation: An evaluation of inversion methods for estimating return energy. *Remote Sensing of Environment*, 164:208–224, July 2015.
- [6] C. Heipke, H. Mayer, and C. Wiedemann. Evaluation of automatic road extraction. *3d Reconstruction and Modeling of Topographic Objects*, 32:151–160, September 1997.
- [7] A. Mabogunje. *The development process: A spatial perspective*. Routledge, 2015.
- [8] D. Mongus, N. Lukac, and B. Zalík. Ground and building extraction from lidar data based on differential morphological profiles and locally fitted surfaces. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:145–156, July 2014.
- [9] D. Mongus and B. Zalík. Parameter-free ground filtering of lidar data for automatic dtm generation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:1–12, January 2012.
- [10] D. Mongus and B. Zalík. Computationally efficient method for the generation of a digital terrain model from airborne lidar data using connected operators. *Selected Topics in Applied Earth Observations*

- and Remote Sensing, *IEEE Journal of*, 7(1):340–351, January 2014.
- [11] D. Mongus and B. Zalik. An efficient approach to 3d single tree-crown delineation in lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 108:219–233, October 2015.
 - [12] A. Najab, I. Khan, M. Arshad, and F. Ahmad. Classification of settlements in satellite images using holistic feature extraction. In *12th International Conference on Computer Modelling and Simulation*, 2010.
 - [13] J. Niemeyer, F. Rottensteiner, and U. Soergel. Contextual classification of lidar data and building object detection in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87:152–165, December 2014.
 - [14] G. K. Ouzounis, M. Pesaresi, and P. Soille. Differential area profiles: Decomposition properties and efficient computation. *IEEE Transactions on pattern analysis and machine intelligence*, 34(8):1533–1548, August 2012.
 - [15] G. K. Ouzounis and M.H.F. Wilkinson. Mask-based second-generation connectivity and attribute filters. *IEEE Transactions on pattern analysis and machine intelligence*, 29(6):990 – 1004, June 2007.
 - [16] G. K. Ouzounis and M.H.F. Wilkinson. Hyperconnected attribute filters based on k-flat zones. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):224–239, February 2011.
 - [17] M. Pesaresi, A. Gerhardinger, and F. Kayitakire. A robust built-up area presence index by anisotropic rotation-invariant textural measure. *IEEE Journal of selected topics in applied earth observations and remote sensing*, 1(3):180–192, September 2008.
 - [18] D. M. W. Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. Technical report, School of Informatics and Engineering, Flinders University, Adelaide, Australia, December 2007.
 - [19] C. Ronse. Set-theoretical algebraic approaches to connectivity in continuous or digital spaces. *Journal of Mathematical Imaging and Vision*, 8:41–58, 1998.
 - [20] F. Rottensteiner. Automation of object extraction from lidar in urban areas. In *Geoscience and Remote Sensing Symposium, 2010 IEEE International*, pages 1343 – 1346. IEEE, July 2010.
 - [21] P. Salembier and M.H.F. Wilkinson. Connected operators. *IEEE Signal processing magazine*, 26(6):136–157, November 2009.
 - [22] J. Serra. *Image Analysis and Mathematical Morphology. 2: Theoretical Advances*. Academic Press, 1988.
 - [23] J. Serra. Connectivity on complete lattices. *Journal of Mathematical Imaging and Vision*, 9:231–251, 1998.
 - [24] J. Serra. Connections for sets and functions. *Fundamenta Informaticae*, 41:147–186, 2000.
 - [25] D. Tomowski, M. Ehlers, U. Michel, and G. Bohmann. Decision based data fusion techniques for settlement area detection from multisensor remote sensing. In *1st EARSeL Workshop of the SIG Urban Remote Sensing Humboldt-Universitt zu Berlin*, 2006.
 - [26] F. van den Bergh. The effects of viewing- and illumination geometry on settlement type classification of quickbird images. In *Geoscience and Remote Sensing Symposium (IGARSS), 2011 IEEE International*, pages 1425 – 1428, July 2011.

Wavelet-based hierarchical heightmap compression method

Michal Lařan

Supervised by: Martin Kahoun

Charles University in Prague
Faculty of Mathematics and Physics
Prague / Czech Republic

Abstract

In this paper, we present a wavelet-based method for lossy compression of heightmap terrain data. It keeps the reconstructed data within a certain absolute per-sample error bound adjustable by the user. This method accepts blocks of float samples of dimensions $2^n \times 2^n$ at the input, for which it can perform progressive mip-maps decomposition. The compression of a 256×256 block takes about 30 ms and the decompression about 1 ms. Thanks to these attributes, the method can be used in a real-time planet renderer. It is able to achieve the compression ratio of 37:1 on the whole Earth 90m/sample terrain dataset transformed and separated into square blocks, while respecting the maximum error of 5m.

Keywords: heightmap, lossy compression, wavelet, lifting, guaranteed maximum error bound, real-time planet rendering, mip-map, progressive decompression, transformation, quantization, filtering

1 Introduction

Real-time rendering of the whole Earth requires working with large terrain data, their storage and distribution. A multiresolution (LOD-ing¹) approach is essential in order to reach reasonable frame rates. In literature, a survey paper summarizing the most common multiresolution rendering methods exists [7]. Some of them also contain data compression.

For example, in C-BDAM [5] and P-BDAM [2], the compression takes place in the refinement of a node of the LOD hierarchy. The values inside a child node are predicted from the parent node as accurately as possible. The differences between these predictions and the real values are called residuals. These residuals are then quantized and compressed by an entropy codec - thus, the compression is lossy. Both these methods use a slight modification of the wavelet lifting scheme, ensuring that the error of the reconstructed data is kept within a maximum error bound adjustable by the user [8].

¹LOD is the abbreviation of level of detail - degradation of quality of the displayed data with the growing distance in order to optimize the rendering

Another paper [6] describes a method based on the same principle - the residuals required to refine a square node of the terrain hierarchy are compressed. The computation of the residuals is based on the JPEG2000 standard, which is again a wavelet scheme. However, this method does not support an arbitrary maximum error adjustable by the user and its rendering pipeline does not handle visual artifacts between adjacent nodes of different LODs.

In practice, many applications handle the real-time rendering well with LOD schemes tailored to their needs. In such cases, a compression method tied to a concrete LOD scheme (which is the case of the mentioned methods) is not feasible. This method handles only the compression, so it can be used as a plug & play component in an existing real-time renderer. Its only job is to compress a block of terrain height samples sized $2^n \times 2^n$ and to provide fast progressive decompression of its mip-maps, while respecting the maximum error bound at every mip-map. The source code of the method is written modularly, so that any representation of the height samples can be compressed - doubles, floats or even arbitrary structures. It is inspired by C-BDAM - the compression method is extracted from the LOD scheme and simplified.

As a case study we have implemented this method as a plugin into an application, which transforms the heights on the planet surface into 256×256 blocks of 32-bit float samples in the unit of meters, which are then stored separately and during the run fetched into a quadtree-based LOD hierarchy. The mip-maps of the blocks are used while looking at them from a side.

This approach introduces heavy redundancy of the data - a block corresponding to a certain quadtree node contains simplified blocks of its children and all these blocks are stored separately. To the contrary, in C-BDAM only the residuals needed to reconstruct the children from the parent node are stored.² However, the reason why this approach is used is that the user can navigate to any area almost immediately - only the data needed for the scene has to be fetched, without having to reconstruct it by traversing from the root. Moreover, this approach enables the user to flexibly extend the terrain data by high-resolution insets. The mentioned redundancy of the data emphasizes the need for an efficient compression method as possi-

²The LOD structure in C-BDAM is not a quadtree, though

ble, doing only what is required - providing the mip-maps while respecting the maximum-error bound of the samples inside each one of them.

In Section 2, we briefly describe the basic theory of wavelets and link C-BDAM and this method to it, in Section 3, we briefly describe the basic outline of the method. In Section 4, we describe the details of the method. In Section 5, we compare the core algorithm of this method to the algorithm of C-BDAM. We present the results in Section 6 and then discuss them in Section 7.

2 The introduction to wavelets

Basically, two generations of wavelets exist - the first one which is based on computation with dilated and translated wavelet function [1] and the second one which is based on filter banks - high-pass and low-pass filtering [3]. It has been proven that these two approaches are computationally equivalent [4].

We will describe the second generation of discrete wavelet transform which is relevant for this work. The basic step of such transform is called lifting - a decomposition of the input signal samples the count of which is a power of two into two equally sized parts - low-pass (the low frequency information) and high-pass (the high frequency information). This step is then recursively applied to the produced low-pass part until its length is 1. The opposite of lifting is reconstruction - enriching low-pass samples by high-pass information to obtain twice as detailed set of samples. It is the exact inverse of lifting. This transform is widely used for compression of data which can be achieved by quantizing the produced high-pass parts (often called residuals).

The lifting is performed in the following way: the input samples x_k are split into the even ones: $x_{2k} = x_e$ and the odd ones: $x_{2k+1} = x_o$. Then two operators are introduced: the prediction operator P which is used to produce the final high-pass part d (residuals) from x_o and the update operator U which is used to produce the final low-pass part s from x_e .

The prediction-first methods firstly apply the prediction operator and then the update operator:

$$\begin{aligned} d &= x_o - P(x_e) \\ s &= x_e + U(d) \end{aligned}$$

The reconstruction is then the exact inverse:

$$\begin{aligned} x_e &= s - U(d) \\ x_o &= d + P(x_e) \end{aligned}$$

To the contrary, the update-first methods firstly apply the update operator and then the prediction operator:

$$\begin{aligned} s &= x_e + U(x_o) \\ d &= x_o - P(s) \end{aligned}$$

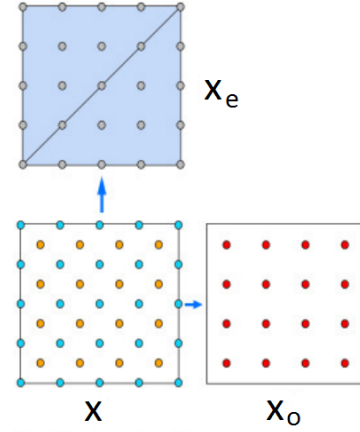


Figure 1: Lifting in C-BDAM - the samples x are separated into the even ones x_e which will become s - low-pass and the odd ones x_o which will become d - high-pass.

Source: C-BDAM [5] (edited)

The reconstruction then looks like this:

$$\begin{aligned} x_o &= d + P(s) \\ x_e &= s - U(x_o) \end{aligned}$$

C-BDAM uses a slight variation of update-first lifting when constructing the coarser LOD s from the finer LOD x . It uses not only x_o , but the whole x as the input to the first update. Moreover, the computation of s cannot be written as the summation of the product of U and x_e anymore, because x_e is multiplied inside $U(x)$:

$$\begin{aligned} s &= U(x) \\ d &= x_o - P(s) \end{aligned}$$

The corresponding reconstruction is:

$$\begin{aligned} x_o &= d + P(s) \\ x_e &= U^{-1}(x) \end{aligned}$$

Besides, the samples x are regularly distributed in the plane, so the decomposition into x_e and x_o depends on the position of the samples, no longer on their indices (Fig. 1). However, the count of x_e is still half the count of x . Note that if the residuals d were simply quantized after lifting, each step of the reconstruction would make the maximum absolute deviation from the original values larger. To ensure the maximum-error bound at each level, the residuals computed during the lifting are corrected according to the actual values in another top-bottom pass which then turns out to be identical to the reconstruction (decompression).

The proposed method uses the same main lifting principle as C-BDAM (update-first and use the whole x as the input of U), but introduces several differences: the count of x_e and thus s is not half the count of x , but one fourth of

it, as each four neighboring pixels inside x are collapsed into one inside s (Fig. 2). In addition, the lifting is not complete, as no prediction and computation of residuals is performed there. These are let to be performed in the second top-bottom pass where also the maximum-error bound is ensured. Just like in C-BDAM, this pass is identical to the reconstruction of the data. The prediction operator is applied more times in the reconstruction which is explained in Section 4. The reasons for all these differences are explained in Sec. 5.

3 The outline of the method

Here is how the compression works. We perform two passes on the input heightmap. In the first bottom-top pass, we compute the target mip-maps - from the largest one to the smallest one. In the second top-bottom pass, we construct the compressed mip-maps from the smallest one to the largest one with respect to the target mip-maps in order to preserve the maximum-error bound. For each constructed mip-map, we store the residuals needed to produce it from the previous decompressed mip-map.

In more detail, given the input square block of float height samples L_n sized $2^n \times 2^n$, n mip-maps $L_{o..n-1}$ are constructed from it. The dimension of L_i is half the dimension of L_{i+1} and L_i is computed from L_{i+1} by averaging of pixels - see the details in the next section.

In the second top-bottom pass, with $L_{o..n}$ at hand, we compute $L_{o..n}^\bullet$ - the final decompressed mip-maps. L_i and L_i^\bullet are of the same size and the maximum absolute deviation between their corresponding samples is not greater than D - the parameter set by the user. We will denote this by:

$$\maxdev(L_i, L_i^\bullet) \leq D,$$

where

$$\maxdev(A, B) = \arg \max_{x,y} |A[x][y] - B[x][y]|$$

We construct these mip-maps with the help of a uniform quantizer Q_D respecting this error bound:

$$\maxdev(Q_D(x), x) \leq D,$$

where x is an arbitrary float sample or block of samples.

L_o^\bullet contains just the quantized sole sample of L_o

$$L_o^\bullet = Q_D(L_o),$$

and L_{i+1}^\bullet is computed from L_i^\bullet by quantizing the differences between the target values L_{i+1} and the predictions from L_i^\bullet :

$$\begin{aligned} E_{i+1} &= L_{i+1} - P(L_i^\bullet) \\ E_{i+1}^\bullet &= Q_D(E_{i+1}) \\ L_{i+1}^\bullet &= P(L_i^\bullet) + E_{i+1}^\bullet \end{aligned} \quad (1)$$

where P is a prediction operator, E_{i+1} are the differences and E_{i+1}^\bullet are the quantized differences. Note that thanks to the fact that the quantizer keeps the maximum absolute error within the bound D and the residuals E_{i+1} are computed with respect to the target mip-map L_{i+1} , $\maxdev(L_{i+1}^\bullet, L_{i+1}) \leq D$, no matter what values are in L_i^\bullet and what is the form of the prediction operator P . All the quantized residuals $E_{o..n}^\bullet$ are then compressed with the help of an entropy codec (Zlib) and saved ($E_o^\bullet = L_o^\bullet$), so the more accurate P is, the better the compression ratio is. The details of the prediction operator used in this method are described in the next section.

During the decompression, the quantized residuals are read and used to progressively reconstruct the mip-map levels $L_{o..n}^\bullet$ (eq. 1).

4 Details of the method

In this section, we describe the details of the method, namely how the target mip-maps are constructed and what the prediction operator P looks like.

The down-sampling of the mip-maps can be performed by any form of averaging. As we saw in the previous chapter, the maximum absolute error does not depend on how the mip-maps look, as long as they contain valid values. However, the way the mip-maps are constructed affects the compression ratio. Moreover, various mip-map constructions produce different visual artifacts. In terms of the visual artifacts, the best way to down-sample a mip-map is to just average the four neighboring pixels $[2n, 2n+1]$, $[2n, 2n+1]$ at L_{i+1} into $[n][n]$ at L_i .

In the previous section, we made a simplification claiming that a decompressed mip-map L_{i+1}^\bullet is constructed from the previous L_i^\bullet in just one step (eq. 1). We did that in order to emphasize the fact, that $\maxdev(L_{i+1}^\bullet, L_{i+1}) < D$. In fact, three such steps happen. Nevertheless, the residuals are checked after each of these steps and all the predictions are made from the decompressed values, so the maximum error bound is still kept. So, when we construct the following decompressed mip-map, every pixel p from L_i^\bullet is substituted by four pixels in L_{i+1}^\bullet as shown in Fig. 2.

The first one of them, labeled a , is predicted directly from L_i^\bullet by a simple prediction operator $P_a(L_i^\bullet) = p$. Following this, the residuals E_a and E_a^\bullet are computed according to the target value a_t in L_{i+1} and a is assigned the final value a^\bullet (eq. 2). It holds that $\maxdev(a^\bullet, a_t) \leq D$.

$$\begin{aligned} E_a &= a_t - p \\ E_a^\bullet &= Q_D(E_a) \\ a^\bullet &= p + E_a^\bullet \end{aligned} \quad (2)$$

The second one of them, labeled b , is predicted from the pixels a^\bullet in L_{i+1}^\bullet by the straight-oriented order 2

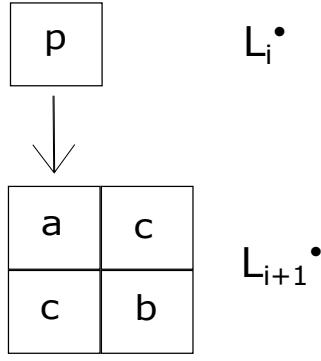


Figure 2: Substitution of a pixel p from $L_i^•$ by four children in $L_{i+1}^•$

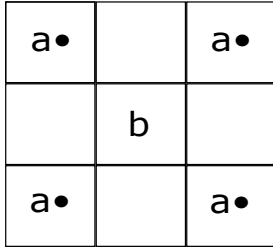


Figure 3: The prediction of b - $P_b(L_{i+1}^•)$ - is the average of all the displayed $a•$

Neville interpolating filter (fig. 3), just like the border values are predicted in C-BDAM. A similar computation of residuals E_b and $E_b^•$ then follows according to the target value b_t from L_{i+1} and b is assigned the final value $b•$ (eq. 3).

$$\begin{aligned} E_b &= b_t - P_b(L_{i+1}^•) \\ E_b^• &= Q_D(E_b) \\ b• &= P_b(L_{i+1}^•) + E_b^• \end{aligned} \quad (3)$$

The cases when the filter comes out of the image are handled by a specific mirror extension (fig. 4). Unlike C-BDAM, where the order 4 Neville filter is used for the interior values, in this method, the order 2 filter is used even for the interior values in order to increase the speed. As an additional optimization, the interpolation with the order 2 filter can be easily cached during horizontal traversal. Moreover, using the order 4 filter made the compression ratio slightly better - probably because it predicts hills and valleys more accurately, but made the quality of the reconstructed heightmap worse - it produced sharper artifacts on the borders of smooth gradient terrain blocks (Fig. 5) and near sharp terrain changes (Fig. 6).

The reason for these artifacts is that while the predictions are close enough to the real terrain (their quantized residuals are zeroes), the reconstructed values might be

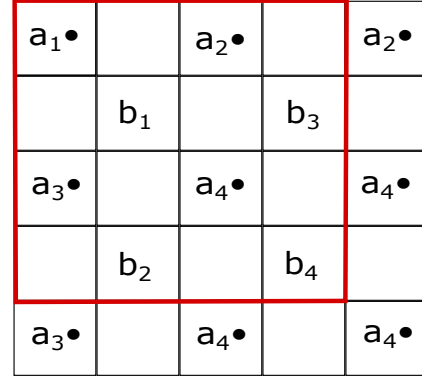


Figure 4: Handling of border cases in the computation of $P_b(L_{i+1}^•)$ - the red line represents the border.

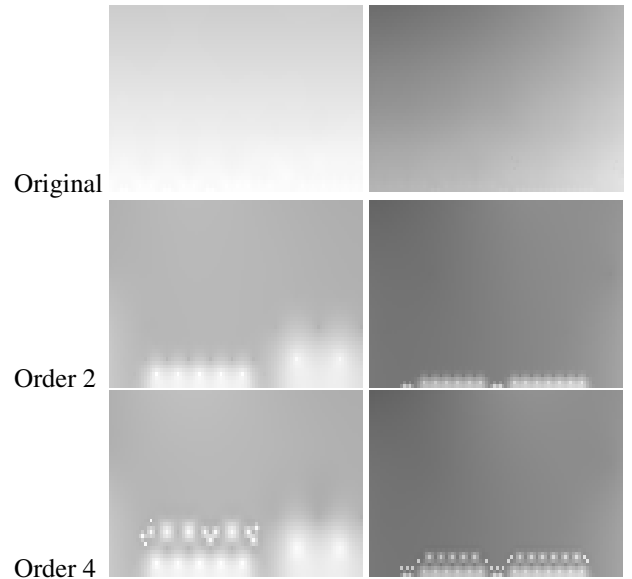


Figure 5: Two examples of different artifacts caused by order 2 and order 4 filters at the border of smooth gradient terrain - the first row shows the target heightmaps, the second row shows the same heightmaps compressed with the order 2 filter, the third row with the order 4 filter.

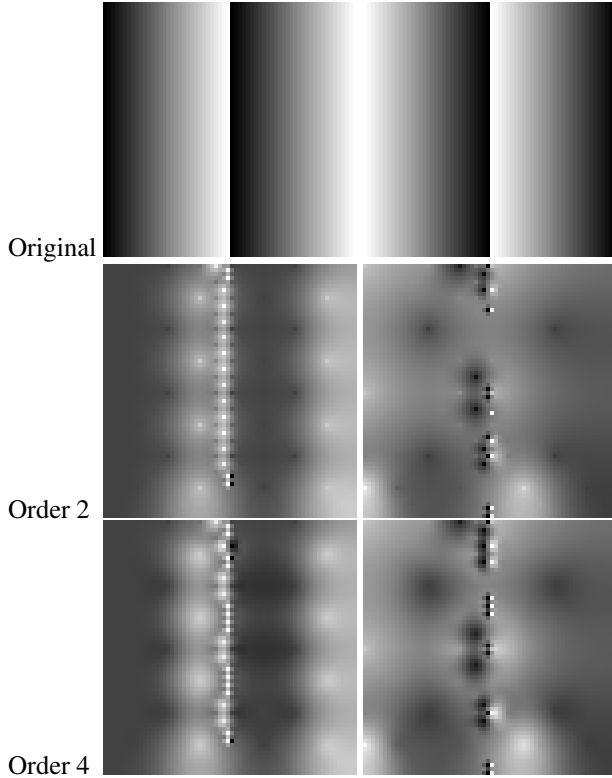


Figure 6: Two examples of different artifacts caused by order 2 and order 4 filters at a sharp terrain change - the first row shows the target heightmaps, the second row shows the same heightmaps compressed with the order 2 filter, the third row with the order 4 filter. The values in the original images range from 0 to 16 and the maximum deviation of the compression is 9.

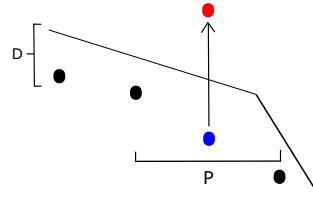


Figure 7: The illustration of how an artifact occurs - the black predictions are within the maximum-error bound D , so they are equal to the reconstructed values, but the blue one is not. Because a uniform quantizer is used, the blue prediction is shifted by $2D - 1$ to the top, creating an artifact.

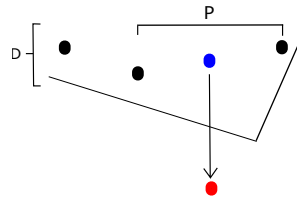


Figure 8: Another illustration of an artifact - the black predictions are within the maximum-error bound D , but the blue one is not. The blue prediction is shifted by $2D - 1$ to the bottom, creating an artifact.

systematically above/under the terrain. But as soon as one prediction is a bit further from the terrain than those at the adjacent pixels, its residual is quantized to a non-zero value and the reconstructed value might flip to the other side of the terrain, producing a visual artifact. This often happens when smooth terrain is followed by a sharp change. The prediction operator might then predict different values near this change, as it reaches out to the area behind the change (Fig. 7, 8). This spike is then propagated to the next levels, but still within the maximum error bound. The mirroring at the borders produces such sharp changes, too, in a different, more complex way. However, some better form of mirroring might sort this out.

While the prediction of the order 2 filter is the average of the four neighboring values, the prediction of the order 4 filter tends to differ from the neighboring values more, so this filter has the tendency to produce more disturbing artifacts.

The remaining pixels labeled c are predicted from the pixels a and b in \mathbf{L}_{i+1}^\bullet by the diagonally-oriented order 2 Neville interpolating filter (fig. 9). The computation of residuals E_c and E_c^\bullet then follows according to the target value c_t from \mathbf{L}_{i+1} and c is assigned the final value c (eq. 3).

$$E_c = c_t - P_c(\mathbf{L}_{i+1}^\bullet)$$

$$E_c^\bullet = Q_D(E_c)$$

	•	
•	c	•
	•	

Figure 9: The prediction of $c - P_c(L_{i+1}^\bullet)$ - is the average of all the pixels marked with a dot - •.

		b_1^\bullet		b_2^\bullet	
	a_1^\bullet		a_2^\bullet		a_2^\bullet
b_1^\bullet		b_1^\bullet		b_2^\bullet	
	a_3^\bullet		a_4^\bullet		a_4^\bullet
		b_1^\bullet		b_2^\bullet	

Figure 10: Handling of border cases in the computation of $P_c(L_{i+1}^\bullet)$ - the red line represents the border.

$$c^\bullet = P_c(L_{i+1}^\bullet) + E_c^\bullet \quad (4)$$

The cases when the filter comes out of the image are handled by a specific mirror extension (fig. 10). For the same reasons as in the prediction of b pixels, the order 2 filter is used for the prediction of all c pixels - both interior and exterior ones. Similarly, the interpolation with such filter can be cached during diagonal traversal.

The residuals E_a^\bullet , E_b^\bullet and E_c^\bullet are then encoded by an entropy codec and stored. The decompression is done in a similar manner with the only difference that the residuals are not computed anymore, but just decoded and read. So, we substitute every pixel from L_i^\bullet by four pixels in L_{i+1}^\bullet , the value of which is computed in three passes of prediction followed by adding the read residual (the last lines of eq. 2, 3, 4).

5 Functional comparison to C-BDAM and wavelets

As we already mentioned, C-BDAM contains the whole rendering pipeline, whereas our method does not. However, it can be compared to C-BDAM in terms of how lifting is performed. As we already mentioned in the end of Section 2, C-BDAM omits a half of the samples while con-

structing a coarser LOD, whereas our method omits three fourths of the samples. This is spatially equivalent to two steps of lifting in C-BDAM (Fig. 1). The first step removes the pixels b and the second step removes the pixels c as seen in Fig. 2. Nevertheless, this equality is only spatial.

In our method, an analogy of the update operator of lifting is used to construct L_i from L_{i+1} (the averaging of four neighboring pixels - Sec. 4). However, the lifting is not complete in our method as it does not contain the prediction operator - no residuals are computed there yet. In C-BDAM, also a prediction operator is used in the lifting to produce intermediate residuals. However, using just these residuals would not guarantee any maximum error bound, so C-BDAM makes another top-bottom pass to correct the residuals against the real values of samples produced in the first bottom-top pass. To make this correction fit into the original wavelet framework, several intricate computations need to be performed, including division, which is quite a large performance hit.

Our vision was that once it is needed to perform an extra top-bottom pass to correct the residuals so that the maximum error bound is guaranteed, it is not necessary to compute any temporary values of the residuals during the lifting steps (the construction of the LOD pyramid). This is why we perform just an analogy of the update (the averaging of pixels) in the update-first scheme and let the following top-bottom pass compute suitable values of residuals. This is obviously a major deviation from the wavelet scheme. In the top-bottom pass, we just predict the values in the finer LOD as accurately as possible, but these predictions have no linkage to the previous bottom-top pass, as they have not been used there at all. Then we directly compute the residuals with respect to the original values computed in the first bottom-top pass at the corresponding levels.

All in all, it can be said that the way the residuals are computed in this method is an extreme simplification of the way they are computed in C-BDAM. This way of computation does not even conform to the second-generation wavelet scheme - the lifting is not complete and the reconstruction is not the inverse of lifting. We think that without the residuals quantization or the per-level correction of residuals, respecting the wavelet scheme makes sense, as it ensures computational equivalency with the first-generation wavelets. However, in case the residuals need to be corrected at each level, we think that conforming to a wavelet scheme makes no sense, because this correction immediately destroys the mentioned equivalence - once a residual is cropped in order to get the resulting value closer to the actual data, it cannot be said that any of the following reconstruction is the inverse to the lifting performed before. Moreover, thanks to the mentioned deviation of C-BDAM from the classical update-first second-generation wavelet discussed in Section 2, we question its computational equivalency with the first-generation wavelet even with no residuals quantization or cropping performed. Because of this, we think that the computa-

tions made in the second top-bottom pass can be optimized this way without any cost. Thus, this method would probably better be called wavelet-inspired than wavelet-based.

6 Results

This method has been applied in the real-time planet renderer mentioned in the introduction on height data of the whole Earth with the resolution of 90m (SRTM). Due to the redundancy of data in the applied LOD hierarchy, the size of the original data was 260GB. With the maximum error bound set to 5m, the size of the compressed data is 7GB, which yields the compression ratio of 37:1.

For a comparison, C-BDAM reached the compression ratio of 64:1 on the same dataset, but with the maximum error bound set to 16m. Thanks to the fact that the LOD hierarchy of C-BDAM contains no redundancy, the size of the original data was just 29GB and the size of the compressed data just 870MB. Under such circumstances, only the comparison in terms of the compression ratio is relevant.

In Fig. 11, you can see a part of a heightmap compressed by this method, together with the differences from the original.

7 Conclusions

In this paper, we described a heightmap compression method designed to be a plugin into an existing real-time planet renderer with its own rendering pipeline. The method proved to be convenient for the purpose, providing fast decompression (only about 1ms per block of data). Its compression ratio is comparable to C-BDAM, which is the method with the best compression ratio among the methods for the terrain compression, which guarantee a maximum error bound adjustable by the user.

References

- [1] P. M. Bentley and J. T. E. McDonnell. Wavelet transforms: an introduction. *Electronics Communication Engineering Journal*, 6(4):175–186, Aug 1994.
- [2] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. Planet-sized batched dynamic adaptive meshes (p-bdam). In *Proceedings IEEE Visualization*, pages 147–155, Conference held in Seattle, WA, USA, October 2003. IEEE Computer Society Press.
- [3] Roger L. Claypoole, Geoffrey M. Davis, Wim Sweldens, and Richard G. Baraniuk. Nonlinear wavelet transforms for image coding via lifting. *IEEE Trans. Image Processing*, 12:1449–1459, 2003.

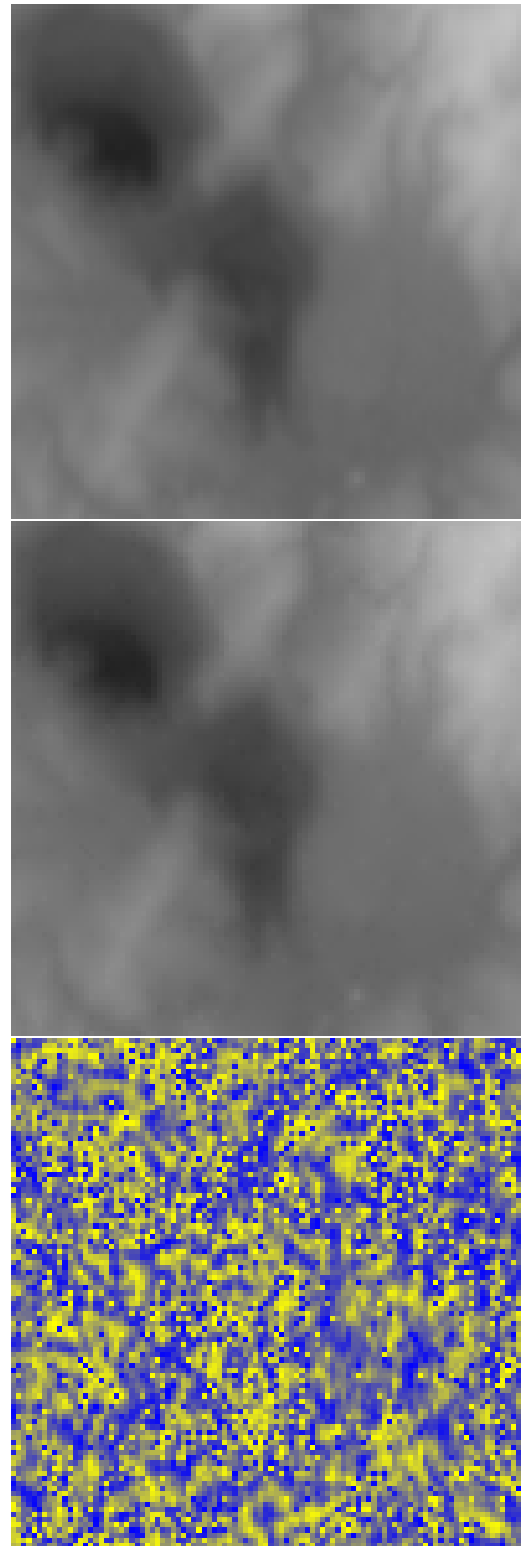


Figure 11: From the top to the bottom - the original terrain, the same terrain compressed with the maximum deviation of 5m, the difference between these two. The brighter the color, the greater the value. In the difference image, the yellow color means 4.5m, whereas the blue color means -4.5m.

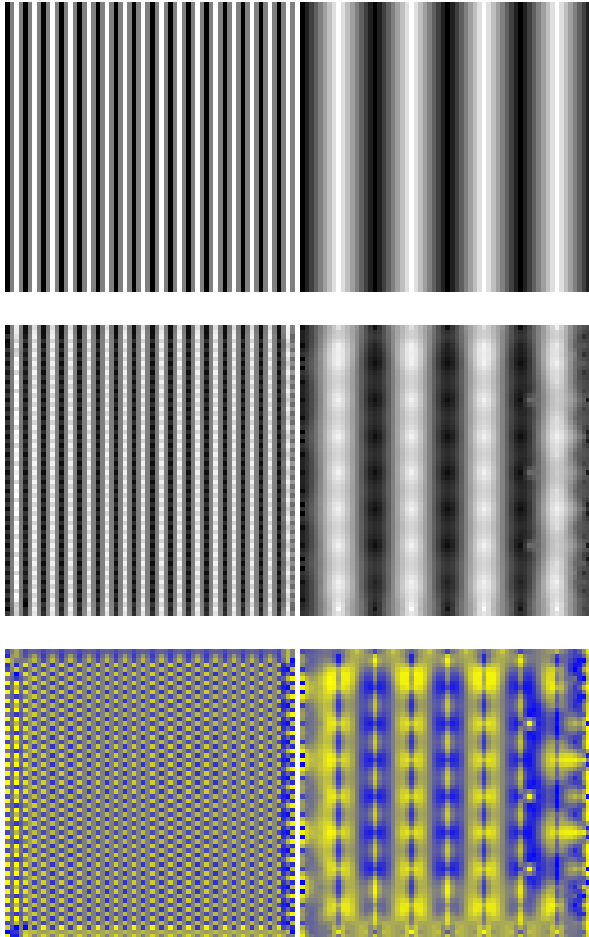


Figure 12: Two synthetic test images of size 64x64, each one containing spiky terrain with the heights ranging from -16 to 16. On the left, the longitude of spikes is 4, on the right, it is 16. From the top to the bottom - the original, compressed with the maximum deviation of 5, the difference between these two. The brighter the color, the greater the value. In the difference image, the yellow color means 4.5, whereas the blue color means -4.5.

- [4] Ingrid Daubechies and Wim Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4:247–269, 1998.
- [5] Enrico Gobbetti, Fabio Marton, Paolo Cignoni, Marco Di Benedetto, and Fabio Ganovelli. C-BDAM – compressed batched dynamic adaptive meshes for terrain rendering. *Computer Graphics Forum*, 25(3):333–342, September 2006. Proc. Eurographics 2006.
- [6] Ricardo Olanda, Mariano Perez, Juan Manuel Orduna, and Silvia Rueda. Terrain data compression using wavelet-tiled pyramids for online 3d terrain visualization. *Int. J. Geogr. Inf. Sci.*, 28(2):407–425, February 2014.
- [7] Renato Pajarola and Enrico Gobbetti. Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, 23(8):583–605, 2007.
- [8] Sehoon Yea and W.A. Pearlman. A wavelet-based two-stage near-lossless coder. *Image Processing, IEEE Transactions on*, 15(11):3488–3500, Nov 2006.

Augmented Reality & Interaction

Foreground Detection and Prototyping of Photographic Composition on Android

Marek Salat*

Supervised by: Adam Herout†

Faculty of Information Technology
Brno University of Technology / Czech Republic

Abstract

The goal of the project is to create a new image prototyping application. The application enables users to capture scenes and enhance reality in an innovative way using an ordinary smartphone. They can replace background of the captured scene and create collages or new original images. The creation image can be created and shared within seconds with minimal interaction. Proper foreground/background detection (image matting) is a vital process. The solution I am suggesting uses appropriate computer vision and image processing algorithms, namely *Global Sampling Matting*. The application is built for the Android platform and uses SDK together with NDK. Sections of the core algorithm are accelerated in the GPU via an OpenGL ES 3.1 compute shader. One part of my work focuses on optimizing algorithms and effective image processing on Android devices. Another part of the work aims to create an intuitive user interface that requires minimal interaction. At the moment, the application is in a pre-release state (open alpha testing) published on Google Play – *ViralCam*¹.

Keywords: Image composition, Scene prototyping, Foreground detection, Background extraction, Image matting, Global sampling matting, Matting, Trimap, Alpha mask, Android, Compute shader, GPGPU

1 Introduction

Users often have an idea on how to incorporate elements of one picture into another picture (see Fig. 1, 2 and 4). However, in many cases, they are not familiar with image editors (Photoshop, Gimp etc.). Most of the times, the programs are either too complicated or available for a price that is prohibitive to this kind of audience. They are not able to create what they want and they must rely on others (Fig. 2 and 1). Even if they try to solve the task using traditional tools it can be an unpleasant task (for example *magic lasso tool* in Photoshop). The task gets even more



Figure 1: The example of the image composition. The iconic iron throne from the *Game of Thrones* with a person sitting on it.

difficult when the user has to use a combination of techniques to adjust areas containing hair and semitransparent objects to achieve a satisfactory result.

The inability to produce own composition or the frustration of doing so are the main issues which led to the project. With just a little help, these frustrated users themselves will be able to produce more creative work. That is the main reason for building *ViralCam*. It allows the user to see in real-time what is being captured and how this scene fits to the other picture. Combining the images together requires minimum interaction; the user just selects foreground and places it over a camera preview. The *ViralCam* is created for Android platform (Section 4).

Selecting the foreground or background (foreground or background detection) is a vital task for this project. The task is more commonly called *image or alpha matting* and is described in the following Section 2.

The project uses known image matting algorithms such as *Global Sampling Method for Alpha Matting* [6] covered in Section 3. The algorithm has been implemented in Android NDK. The nature of the algorithm, and the fact that some newer Android devices are equipped with *OpenGL ES 3.1 Compute Shader*, allowed that it has been massively parallelized and run on GPU (described at Section 4.2). The GPU parallelization sped up the whole process from 4 – 7 seconds to less than a second which could brought

*salat.marek42@gmail

†herout@fit.vutbr.cz

¹Available for Android 4.1 – 6.0 <https://play.google.com/apps/testing/com.salat.viralcam.app>



Figure 2: Image shows that it does not pay off to ask for image editing on the Internet. The scene is simple, the user had a photo and wanted to put his own figure over the first photo. Request from *CollegeHumor* [4].

the project closer to a nice-to-have feature *real-time scene visualization*.

2 Alpha Matting

Throughout this paper, it is assumed that a color image \mathbf{I} consists of a 3D discrete array of pixels (red, green, blue). The Alpha matting or the digital matting is associated with the problem of softly separating an image into a foreground image \mathbf{F} and into a background image \mathbf{B} from a single input image \mathbf{I} along with its opacity mask α . It means that \mathbf{I} is formed by linear blending of \mathbf{F} and \mathbf{B} using α . These three images relate by matting Equation (1). Image matting is used in interactive image editing, video seg-

mentation and also in film making.

$$\mathbf{I} = \mathbf{F}\alpha + \mathbf{B}(1 - \alpha) \quad (1)$$

The matting problem cannot be solved uniquely since there are many possible foreground and background explanations for the observed colors [10]. Equation (1) shows that there are seven unknowns on one hand but only three equations (three unknowns for foreground color, three for background color and one for alpha, the only known is image color) solving them.

Despite the fact that the problem is inherently under-constrained, it could be solved by adding more information about the image. The additional information could take form of a scribble-set or trimaps (see Figure 3). Such information labels pixels into two groups: the first group defines pixels which are definitely foreground, the second group labels pixel which are definitely background. The remaining pixels are marked as unknown. The alpha value α is then calculated for unknown pixels only.

Even with a known alpha value and these constraints, the problem is still ill-posed (the alpha value may be estimated incorrectly in favour of foreground or background). Therefore, several solutions proposed other additional constraints [15].

2.1 Trimap as a User Input

As mentioned before, the input can be in the form of scribbles [9] or trimaps [8]. I have found scribbles to be unintuitive and most of the times, it was difficult to predict the result without knowing the principles of the underlying algorithm. Very often, the resulting image was completely different from the expectations.

In this project, I chose trimap as the input constraint. Trimap segments the image into three regions: *definite foreground*, *definite background*, and *unknown*. Alpha values are calculated only for unknown pixels using knowledge from other regions since their alpha values are known.

The trimap can be easily drawn and with a little help. A user is then able to create it on a first try in few seconds [8]. More about constructing trimap in the Anroid application can be found in Section 4.

The trimap quality plays a significant role in the precision of the resulting alpha mask (or image). Very good trimap can reduce the number of unknown variables that imply fewer variables to estimate. The thickness of the unknown region creates a considerable factor of a good trimap [13].

The project aims at effective trimap creation requiring minimal interaction and, at the same time it must be intuitive. Quality of the computed alpha mask and time to compute trimap are also taken in account.



Figure 3: Example of the matting problem. From the left *first*: input image. Second: user-defined trimap (blue for background, red for foreground and green as an unknown region). Third: computed alpha mask. Fourth: original image without background (foreground area and unknown region are merged). Last: background is replaced. Figure is borrowed from book *Image and Video Matting: A survey* [13].

3 A Global Sampling Method for Alpha Matting

The algorithm is inspired by *Robust matting* [12], *Knock-out* [3], *Shared matting*[5]. All these methods collect nearby (in a certain metric, e.g., Euclidean/geodesic distance, or nearest on a ray) samples. He et al. [6] proposes use a global sample set that contains all available samples. The spatial distance and the color fitness are then considered simultaneously for selecting the good samples from this set [6].

The goal is to select a good pair of samples (foreground, background) for any unknown pixel from all candidate pairs. The algorithm comprises following steps (the following steps are brief summary of the algorithm, more information can be found in the seminal work [6]):

Create global sample set – foreground (background) sample set consists of all known foreground (background) pixels on the unknown region’s boundary. Global samples are denoted as **FB** search space.

Extend global sample set – Add random pixel to global set. I have chosen to add the same amount of pixels as the total number of pixels in the global set.

Initialize samples – Each of the unknown pixel has its own sample. The sample consist of foreground pixel, background pixel, closest distance to foreground and background boundary, cost value and alpha. Sample initialization assigns random boundary pixel from the global set, cost value is set to infinity. The closest distance is found by iterating over global set and comparing distances.

Apply SampleMatch algorithm – All pixel from global set are sorted by intensity (actual sorting criteria does not matter [6]). The goal is to find a pair (foreground, background) of points in the **FB** search space for each unknown pixel which has the (approximately) smallest cost. The method iterates over *propagation* and *random search* stages. As He et al. [6] claims, ten iterations are sufficient.

Propagation – For the unknown pixel being scanned, its cost is updated by considering the current optimal sample pairs of its neighbouring pixels. I have chosen to search in 8-neighbourhood.

Random search – Intuitively, the random search step tests a sequence of random points in the neighborhood (in the **FB** space) of the current optimal point. The neighborhood radius decreases exponentially (in each iteration).

4 Android Application

The most important part of the project is to create the application that will respect UX and design principles. For this part I decided to implement an application for Android. According to Gartner, [14] Android shares more than 84% of the market at the moment. The second place belongs to iOS.

4.1 Android NDK Utilization

Implementing a real-time image processing application in interpreted language (Android main programming language is Java) can be challenging. Fortunately, Android offers capabilities of native code via Android NDK. Programmer can choose from C or C++. All devices are equipped with standard libraries such as STD for C++. There are some limitations, but for most cases, it is not an issue.

Programming in NDK brings significant performance improvement. The programmer may also target specific architecture and can compile optimized code for multiple CPU architectures (x86-x32, arm7, arm8). The current implementation of the *A Global Sampling Method for Alpha Matting* for images 800x600 on Nexus 5X takes 4–7 seconds depending on the size of the unknown region.

From my experience, the bottleneck on the Android platform is usually the memory. It is not recommended to use more than three images of the same size as the device screen. Older platforms allow allocating only 16 MB per application [1]. Furthermore, allocation of more continuous memory may be an issue; e.g. in case of a bitmap with dimension of the size of the screen. Other problems could be caused by reading randomly from a bitmap due to skipping large chunks of memory which are not cached. All these things must be considered when porting code to

android NDK.

My application assumes that users will create photos quickly, spontaneously and the results will be shared on social media. The typical image size on Facebook, Twitter, or 9gag is around one mega pixel. On the grounds of performance reasons and the fact that the image does not have to be in full resolution I have decided to scale input images to a lower resolution, somewhere around 0.5 to 1 mega pixel, typically 800×600 .

4.2 GPGPU via OpenGL ES 3.1 Compute Shader

Some older Android devices supported OpenCL. It was a great tool for massive parallelization. Unfortunately, Google dropped the support and introduced their own parallelization library called RenderScript. From my point of view, the tool is badly documented and the programmer does not have a good control of whether the code ends up running on the GPU or just on the CPU. Android 5.0 introduced OpenGL ES 3.1 with compute shaders. At the moment, compute shaders are supported at least on 7.1 % devices [2]. It needs to be mentioned that a device running Android 5.0 or higher *may or may not* support the feature.

Programmers might take advantage of this technology. However, compute shaders are still bound to rendering pipeline and cannot be used without rendering on the screen (*GLSurfaceView* and *GLSurfaceView.Renderer* must be used²).

4.3 Implementation of the Compute Shader

The matting algorithm comprises of the following steps:

1. **Find the boundary for foreground and unknown region and boundary for background and unknown region.** Boundary is found by checking neighbourhood each pixel. If the pixel represents the foreground and at least one neighbour is labelled unknown, the pixel is assigned to the foreground set, similarly for the background pixel. The step is independent on other pixels. However writing to the global set must be synchronized in the way that no value is rewritten or maximal size is preserved. In the compute shader implementation atomic counters are used to prevent both issues.
2. **Extend the boundary by adding random pixels to the global set.** The step assigns random pixel from foreground region to the foreground boundary and analogous to the background pixel. The pixel is not assigned to the global set if global set is contains

²You can download the fragment I use at GitHub repository. You can also find there few examples showing how to compile the shader, bind buffers, textures, pass data through shader and also how to run them. Please visit <https://github.com/MarekSalat> for more information.

more than twice the amount of the original boundary size. The step is also independent on other pixels and can be run in parallel.

3. **Initialize samples for each unknown pixel.** First, the initialization, finds the distance to the nearest foreground and background pixel from the global set (measured by euclidean distance). Secondly, it assigns random foreground and background pixel from global set. The cost value is set to infinity, the alpha value is set to zero. The initialization is also independent on other samples.
4. **Iterate over propagation and random search.** The propagation searches in neighbourhood for better sample selection. Random search follows immediately afterwards. It selects random pixel from the global set by decreasing radius exponentially. Both propagation and random search update the cost value and the alpha value. As mentioned above in Section 3, the step is run for each sample (unknown pixel) and each sample processing is independent from others.
5. **Update alpha mask.** At this point the algorithm calculated all alpha values for unknown pixels. The step fills whole alpha mask by corresponding alpha values (255 for definite background, 0 for definite foreground and sample alpha value for its unknown pixel).

Each step must be completed before the next step can be run; the same applies to the iteration over *propagation* and *random search*. Thus, each step represents one shader (kernel) which is dispatched only if previous step has finished. The CPU implementation loops over all pixels in the image for in each step; each thread computes one pixel. The dimension of the workgroup is 32×32 and the whole problem has the dimension of the image (rounded and divided by size of the workgroup).

Better performance can be achieved by using texture units for reading from images. The texture unit caches images very well for access in neighbourhood around pixel unlike access in buffer.

Computing alpha mask has been sped up from 4-7 second to 300-500 milliseconds. This has been measured on Nexus 5X.

4.4 User interface

Figure 4 shows application capabilities. When the application is started, background is selected first. The Application offers several predefined images. The user can also choose pictures from an image library.

Scene capturing and visualization – At the moment, for the sake of MVP (minimal viable product) simplicity, I have chosen to simply overlap the camera output by semi-transparent background. It turns out that this solution is sufficient enough for visualizing the scene. The user can

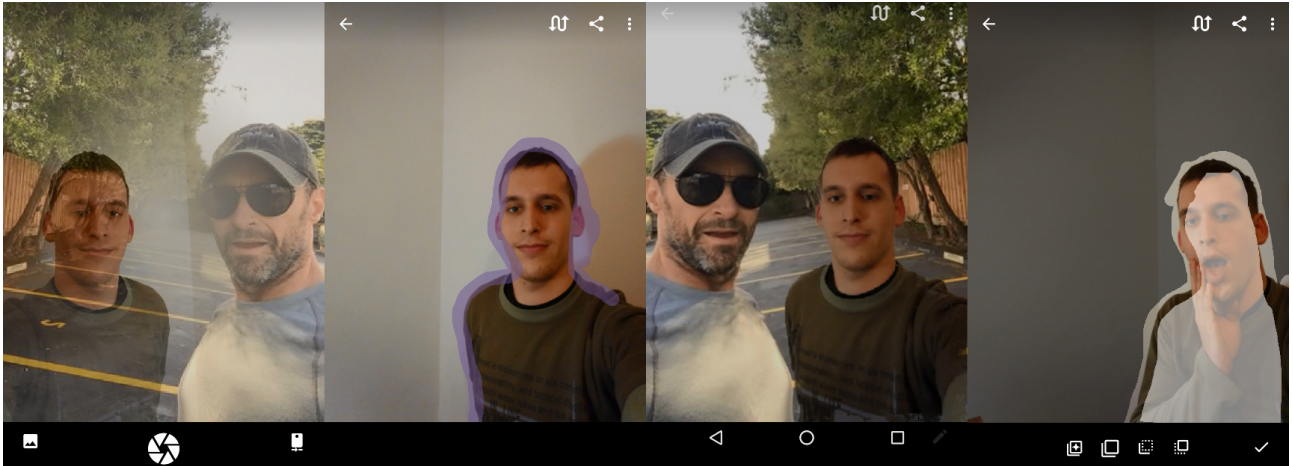


Figure 4: The first image shows the visualization of scene capturing. The second image shows trimap initialization. The third image is the result. Last picture contains trimap editing. .

swipe over the image to increase or decrease background transparency.

Trimap initialization – User roughly marks the object edge. This does not have to be precise to the pixel. The inner area is automatically marked as definite a foreground. Outer area is marked as definite a background. If the image contains larger transparent areas, such as hair or spikes, the trimap can be edited and these areas can be marked as an unknown region.

Show result – After trimap initialization, the application displays the result. At this step, the user can pinch to zoom to validate all the problematic parts of the image.

Edit trimap – User can always edit the trimap by simply drawing a definite foreground, background or unknown region (application user brush named *Clear*). In most of the applications, this step requires switching between different brushes by the user. To improve the user experience, I introduced a smarter brush which selects the brush based on the starting point of the user's swipe motion. In other words, if the user starts drawing from the background, the background brush is selected, other brushes are initiated by drawing from their respective areas. The final step is to share the result on social media.

5 Conclusions

At the moment, the published alpha application is using only the NDK implementation. The compute shader variant is not ready for production. The performance and quality has been tested on *LG Nexus 5X* with a standard dataset [11]. Provided dataset contains various images with trimaps with respective true alpha mattes. The dataset is composed of images from non-transparent to semi-transparent or even fully transparent images, also images with short or long hair.

Measured average time per unknown pixel is 0.11 milliseconds which is 7.2 seconds per image (roughly half

megapixel). The 7-seconds processing time it is not close to the *real-time preview* goal. However, the compute shaders seem to be promising. For *real-time preview* the quality may be lower and the whole process may be sped up by scaling the image to a lower resolution. On the other hand, a significant drawback for the compute shader is lower support on Android devices (less than 7%).

The average MSE error without pre-processing or post-processing is 353. For comparison, the *Robust matting* [12] MSE is 350 on the same data set and the method is ranked on *alphamatting.com* as 36th. The quality could be better and it will be addressed in future releases, still it is sufficient enough for *MVP (minimal viable product)*. The general image quality and composition perception will be a part of the future user testing evaluation.

There are several ways to increase the matte quality. First way is trimap pre-processing where colors closer to the unknown region boundary with similar color properties (color and spatial distance) are considered to be known depending on other regions. Such a pre-processing reduces the number of unknown variables and increases overall matte quality. The other method is the post-processing as He et al. [6] proposed. They used *Fast Guided Filter* [7] which ran 0.3 second per mega pixel, but it could be estimated to run slower on regular mobile device.

The application is published as an open alpha version on Google Play – *ViralCam*¹ for download.. Within few weeks, it is going to be published to the production. The most common issue so far was a lack of a help which was added in version 1.3. Other important issue coming from users were problems with camera focus and, on some devices, also image rotation after capturing a scene. All these issues have been addressed in last the update. However, the camera rotation issues still persists on *Sony Xperi E4g*. For the production version, *Google Analytics* will be integrated to gather more precise data about user acquisition and behaviour within the application.

References

- [1] Android compatibility downloads. <http://source.android.com/compatibility/downloads.html>, 2016.
- [2] Dashboards. <http://developer.android.com/about/dashboards/index.html>, 2016.
- [3] A. Berman, A. Dadourian, and P. Vlahos. Method for removing from an image the background surrounding a selected object, October 17 2000. US Patent 6,134,346.
- [4] CollegeHumor. 12 people who had photoshop requests and got just perfect results. <http://www.collegehumor.com/post/6997451/12-people-who-had-photoshop-requests-and-got-just-perfect-results>, 2014.
- [5] Eduardo S. L. Gastal and Manuel M. Oliveira. Shared sampling for real-time alpha matting. *Computer Graphics Forum*, 29(2):575–584, May 2010. Proceedings of Eurographics.
- [6] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun. A global sampling method for alpha matting. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2049–2056, June 2011.
- [7] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, June 2013.
- [8] C. L. Hsieh and M. S. Lee. Automatic trimap generation for digital image matting. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, pages 1–5, Oct 2013.
- [9] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 61–68, June 2006.
- [10] Richard J. Radke. *Computer Vision for Visual Effects*. Cambridge University Press, New York, NY, USA, 2012.
- [11] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. A perceptually motivated online benchmark for image matting. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1826–1833, June 2009.
- [12] J. Wang and M. F. Cohen. Optimized color sampling for robust matting. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.
- [13] Jue Wang and Michael F. Cohen. Image and video matting: A survey. *Found. Trends. Comput. Graph. Vis.*, 3(2):97–175, January 2007.
- [14] Meulen R. Woods V. Gartner says emerging markets drove worldwide smartphone sales to 15.5 percent growth in third. <http://www.gartner.com/newsroom/id/3169417>, 2015.
- [15] Y. Zheng and C. Kambhamettu. Learning based digital matting. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 889–896, Sept 2009.

Natural interaction with small 3D objects in virtual environments

Irfan Prazina*

Supervised by: dr. Selma Rizvić†

Faculty of Electrical Engineering Sarajevo
University of Sarajevo / Bosnia and Herzegovina

Abstract

This paper presents a novel solution for natural interaction with small 3D objects in virtual environments, using Leap Motion as sensor and WebGL for presenting a virtual hand and 3D objects. Most current solutions for this type of interaction use gestures as a mean of control. The interaction will emulate a natural grab-and-move action of a human hand. The virtual hand will be displayed alongside with a 3D object and will be able to grab and move the object using Cannon.js physics engine. The virtual hand which emulates real hand should minimize the user's learning time.

Keywords: natural interaction, virtual reality, WebGL, user immersion

1 Introduction

Many cultural monuments and artifacts disappear or get destroyed. One way to preserve them for next generations is through technology. Preservation is not enough, as even the best technology if not being used gets forgotten and obsolete. To keep people interested in virtual heritage the new ways of immersive interaction must be developed.

Once we have geometric data of virtual objects and environments, and different ways of presenting them, we must find means of navigation which will help a user to get the most information and best experience. There are different ways of interaction with virtual objects and environments. Users can be interested in a new technology and technique, but that interest will be short lived if it is not easy to use. Our virtual presentations are web based because we would like to present the selected cultural heritage objects to everyone, not just the people who come to visit museums or archaeological sites. Our experience shows that online presentations also attract people to physical locations of monuments.

We will present a way of interaction with virtual objects using only a hand and Leap Motion. This type of interaction will mimic the movements and dynamics of real hand. Users will be able to see a virtual representation of their hand and with very little delay see all the movements they make.

The paper is organized in the following way: Section 2 presents related work, in Section 3 is description of our natural interaction's concept, Section 4 presents a case study and the interaction implementation, Section 5 presents our user evaluation and a conclusion is given in Section 6.

2 Related work

Papers [7], [8] and [5] describe the use of the Leap Motion for a gesture recognition.

In [7] comparison of Kinect and Leap Motion is given. The authors state that Kinect gives more detailed data, but is less accurate, while Leap Motion gives high level data as a set of relevant hand points and pose features.

In [8] Leap Motion is used to recognize sign language gestures. The Sign language gestures recognition can be problematic when one or more fingers are not in line of sight.

In [5] the authors use the Leap Motion API to detect key presses and to create a keyboard like musical instrument. These works show advantages of Leap Motion over Kinect, like well documented API and user friendly data structure. As drawback Leap Motion tends to lose track of some fingers if line of sight is blocked.

In [4] a detailed analysis of the precision and reliability of Leap Motion is given. The distance from the hand to the sensor is a parameter that significantly affects Leap Motion's consistency and performance. Leap Motion's accuracy significantly drops if the hand is more than 250mm above the controller. Inconsistent sampling frequency is stated as an important limitation of the controller performance.

In [6] Kinect is used to detect human gestures for interaction with 3D objects like moving, opening car doors, selecting a seat, etc.

Common use of Leap Motion is in recognition of gestures and using them as triggers for an action. In our paper hand movements are directly mapped to movement of the virtual hand and the virtual hand interaction with other objects is simulated using physics engine. In [9] the autor describes how a physics engine can be implemented in JavaScript.

*iprazina1@etf.unsa.ba

†srizvic@etf.unsa.ba

3 Our concepts of natural interaction

Natural interaction is often realized with excessive use of gestures and gesture recognition. In this work we tried to make interaction with a virtual object as close to the interaction with a real object as possible. Users need to learn very little before interaction. We achieved this by simulating behavior of a real hand in 3D virtual environment using physics engine for simulating physics and web browser as an execution environment. Web technologies are used because everyone has a web browser, and those who don't have Leap Motion can use a mouse for the interaction with objects.

To simulate the hand behaviour in a virtual environment we used:

- Three.js – JavaScript library for creating and displaying 3D graphics [3], it is based on WebGL,
- Cannon.js – JavaScript library used as physics engine [1] and
- Leap.js – Leap Motion's JavaScript library for acquisition of a Leap Motion data [2].

In this type of interaction physics engine has a pivotal role. If we do not set collisions right, the hand will go through objects or it will push objects before we even get close. Cannon.js is chosen because of its simplicity and speed. Lack of feeling of touch was the problem we anticipated before making this work. When one is interacting in a way described here, grabbing movement can be tiresome, because in some cases one will grab too fast, too strong or too weak and the object will slip. To solve this problem we used Cannon.js lock constraint force. The constraint mimics glue effects, it makes one object stuck to another. If the user puts his palm close enough to an object, the object will stick to the palm and will not slip in case of fast grabbing, very strong or very weak grip. When an object is on a palm the user can rotate it by rotating the palm, as one would rotate it in real life. To make a virtual hand able to hold an object we need to add collision boxes to the hand and to the object. The hand is made of boxes, where one box represents one bone of the hand.

Each finger consists of 4 bones (Figure 1) where the first bone is in the palm and last three are movable finger's bones. Each bone in the hand has a mesh and a body related to it. The mesh is used for 3D presentation, and the body is used for physics simulation. The shape of the mesh is the same as the shape of the body so objects do not go through visual representation of the fingers or avoid them at distance (Figure 2). Leap Motion detects bones positions and orientation data. On each Leap Motion's loop iteration we get new data which are used to move bone bodies; these two steps are done using Cannon.js and Leap.js. After bone body has been moved, the mesh position must be updated; this is done in Cannon.js loop. If some bone

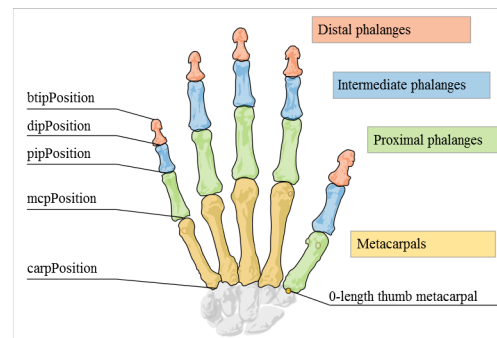


Figure 1: Structure of the virtual hand, picture is from developer.leapmotion.com

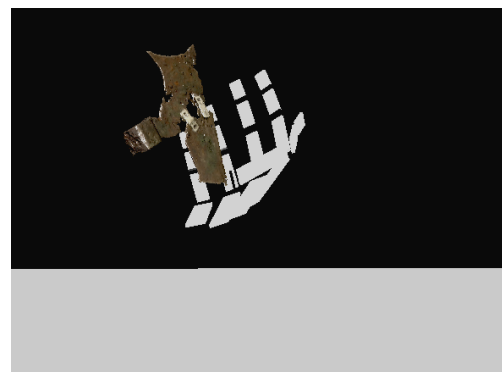


Figure 2: Display of the virtual hand

has hit an object Cannon.js will calculate its velocity and object's position will be updated accordingly. The final step in the iteration of Leap Motion's loop is rendering changes in the scene (Figure 3).

One of the challenges was to synchronize all coordinate systems. Each library has its own coordinate systems for bone positions (Leap.js), body positions (Cannon.js) and mesh positions (Three.js). Details on coordinate systems synchronizations will be explained in Section 4.2.

4 Case study

4.1 The White Bastion project

The fortification known as "White Bastion" is one of the most impressive and important historical sites in Sarajevo. It is located on the south-east outskirts of the city, with an overview of the city valley. Through the history it had a very significant and strategic position. The fortification is a part of the dominant defense walls that were surrounding the old city of "Vratnik". The value of the historical site presents the various strata, starting from medieval until the present time. During the archaeological excavations remains from medieval fortification from 14th century were found, Ottomans period (17th century) when the fortification was expanded and some new objects were built. During Austro-Hungarian rule the part of the fortification and

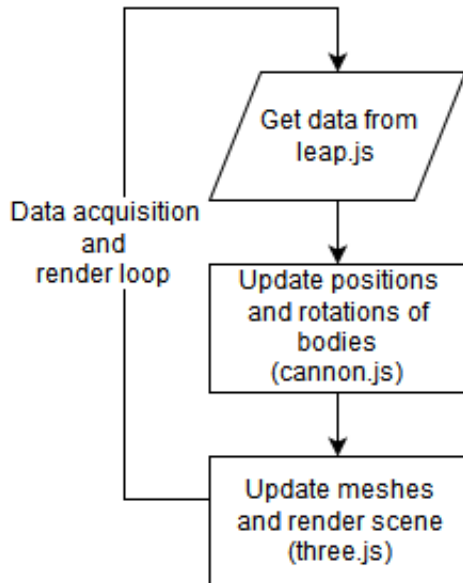


Figure 3: The algorithm diagram

the object inside the walls were demolished and destroyed and a new group of objects was built. During the early excavation, a significant number of artifacts was found, registered and conserved for the purpose of the exhibition hosted in Museum of Sarajevo. 4D Virtual presentation of White Bastion aims to present the historical development of this cultural heritage object through digital stories combined with interactive 3D models of the Bastion in various time periods. These models contain digitized findings from the site and their 3D reconstructions. Structure of the project is displayed in Figure 4.

4.2 Implementation details

The implementation of the idea required the use of three libraries: one for physics simulation, one for display and one for acquiring data. The main challenge of this implementation was to synchronize all the position and orientation data, and all the coordinate systems. Leap Motion uses the three dimensional right-handed Cartesian coordinate system which has millimeters as units. The x axis is placed on the longer side of the device. Leap Motion uses a rotation matrix to represent bone's orientation. Cannon.js will calculate data in any coordinate system, but one coordinate system must be used consistently. Body's orientation is represented in the form of the quaternions. Three.js uses the right-hand coordinate system where z axis is perpendicular to the screen oriented toward a user. Leap Motion coordinate system is used as a reference, mesh and body positions are set accordingly. Leap Motion uses loop iterations to update bone position and orientation. We must update body and mesh positions on each loop iteration. In the first loop iteration we set mesh position directly to be equal to bone positions. This is the initial-

ization phase in which each mesh gets the position of its corresponding bone.

```

mesh
  .position
  .set(
    bone.center()[0],
    bone.center()[1],
    bone.center()[2]
  );

mesh.setRotationFromMatrix(
  (new THREE.Matrix4)
    .fromArray(bone.matrix())
);

```

In the following iterations a mesh position is set using the body position. The body position is set in a same way as mesh position with exception of orientation. The orientation of the body is set using the quaternion helper object. Helper object is initialized using bone rotation matrix, and its orientation is adjusted to a base bone rotation. Base bone rotation is set to Euler angles $(\frac{\pi}{2}, 0, 0)$.

```

quat = new THREE.Quaternion();
quat
  .setFromRotationMatrix(
    (new THREE.Matrix4)
      .fromArray(bone.matrix())
  );
quat.multiply(baseBoneRotation);
body.quaternion.set(
  quat.x,
  quat.y,
  quat.z,
  quat.w
);

```

In this way the virtual hand's movement is solved. Setting the positions of meshes according to the positions of the bodies ensures that objects underlie the physics simulation. To solve afore mentioned problems of lack of tactile feedback and problem of "slippery" objects we added lock constraint force which activates when the palm is at a distance less than 50 units. In that case a new constraint object is created between small 3D object and palm bone. The small 3D object will still move inside of the hand and it will appear as it is stuck with one side to the palm so it will not slip away if the grip is fast, too strong or too weak. The constraint is deactivated when user moves away his hand.

There are two events that must be handled: when a hand is placed above the sensor and when a hand leaves sensor's space. On first event bodies and meshes are created and the Leap Motion loop starts updating bone positions. On second event all bodies and meshes must be destroyed.

The described interaction is added to the White Bastion project. In the White Bastion project there is a web page for each of the digitalized archaeological findings. The

web page's layout is made of two Three.js scenes; in the first scene 3D model of the excavated object is presented and in the second scene the reconstruction of the object is displayed. In the initial version of the project objects can be viewed from all sides using mouse for movement and rotation. In the new version the Leap Motion interaction is added for the museum setup. An example web page is presented in Figure 5.

5 Evaluation

The Leap Motion interaction is evaluated using qualitative user experience analysis. Participants in the analysis have different educational backgrounds, age and gender. Ten participants have been interviewed. The interview has been structured in such a way that participants can share their impressions on the Leap Motion interaction implementation and the interaction using a computer mouse. Users had enough time to experience both interactions. They were asked to look the object using mouse, and after they explored the object from all sides they were introduced to Leap Motion. In the Leap Motion interaction users were asked to look at same object, and how hard was for them to see all the sides and details using the new interaction. Hypotheses that are being checked in the interview are presented in Table 1. The interview is structured in

H1	Leap Motion interaction is easier to use than mouse
H2	Leap Motion interaction is more engaging
H3	Leap Motion can make people interested in virtual heritage

Table 1: List of hypotheses

form of open questions which participants used as guidelines. This type of interview is used to give participants opportunity to express their opinions. Questions are formulated in a way that one question can give information about one or more afore mentioned hypotheses. List of questions is given in Tables 2 and 3. In the interview

Q1	What is your opinion on interaction with a virtual object using mouse? (What are advantages and disadvantages?)
Q2	What is your opinion on interaction with a virtual object using Leap Motion?(What are advantages and disadvantages?)
Q3	What is easier to use and why?

Table 2: Questions for hypotheses H1 and H2

Q4	What is your experience with virtual scenes and virtual cultural heritage?
Q5	What do you think about virtual heritage and this type of projects?

Table 3: Questions for hypothesis H3

most of the subjects had similar experience while using the interaction with Leap Motion. All of them spent more time using Leap Motion than the mouse interaction, but

they found mouse interaction easier to use so H1 has not been confirmed. Even though participants found mouse easier to use, they also found the Leap Motion interaction more interesting and immersive, which proves H2. Some of them made statements like: "Leap Motion is interesting, has potential, but less precise", "When using the Leap Motion all focus is on the 3D object", "Leap Motion can make people interested in viewing all these objects", etc. (H2) All participants claimed that mouse interaction is more precise and reliable but dull and less immersive. Participants had little experience with virtual cultural heritage and they stated that projects like this make them interested in history and cultural heritage, thereby confirming H3.

6 Conclusion

In this paper we presented a novel solution for natural interaction with small 3D objects in virtual environments, using Leap Motion as sensor and WebGL for presenting virtual hand and 3D objects. In the qualitative user experience analysis of this method the participants noticed that the Leap Motion interaction has precision and reliability issues in comparison with the traditional mouse interaction. However, they spent more time using the Leap Motion interaction, and they found that interaction immersive and interesting. Participants stated that they had more fun and found new experience more engaging. This type of interaction could be viable if precision and speed of the Leap Motion controller is improved. In the current state Leap Motion can still be used as presented because of its immersive potential, but improvements are needed.

References

- [1] Cannon.js. <http://www.cannonjs.org>. Accessed: 2016-02-14.
- [2] Leap motion. <http://developer.leapmotion.org>. Accessed: 2016-02-14.
- [3] Three.js. <http://threejs.org>. Accessed: 2016-02-14.
- [4] Jože Guna, Grega Jakus, and Matevž Pogačnik. An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors*, 2014.
- [5] Jihyun Han and Nicolas Gold. Lessons learned in exploring the leap motion sensor for gesture-based instrument design. *International Conference on New Interfaces for Musical Expression*, 2014.
- [6] Jong-Oh Kim, Mihye Kim, and Kwan-Hee Yoo. Real-time hand gesture-based interaction with objects in 3d virtual environments. *International Journal of Multimedia and Ubiquitous Engineering*, 2013.

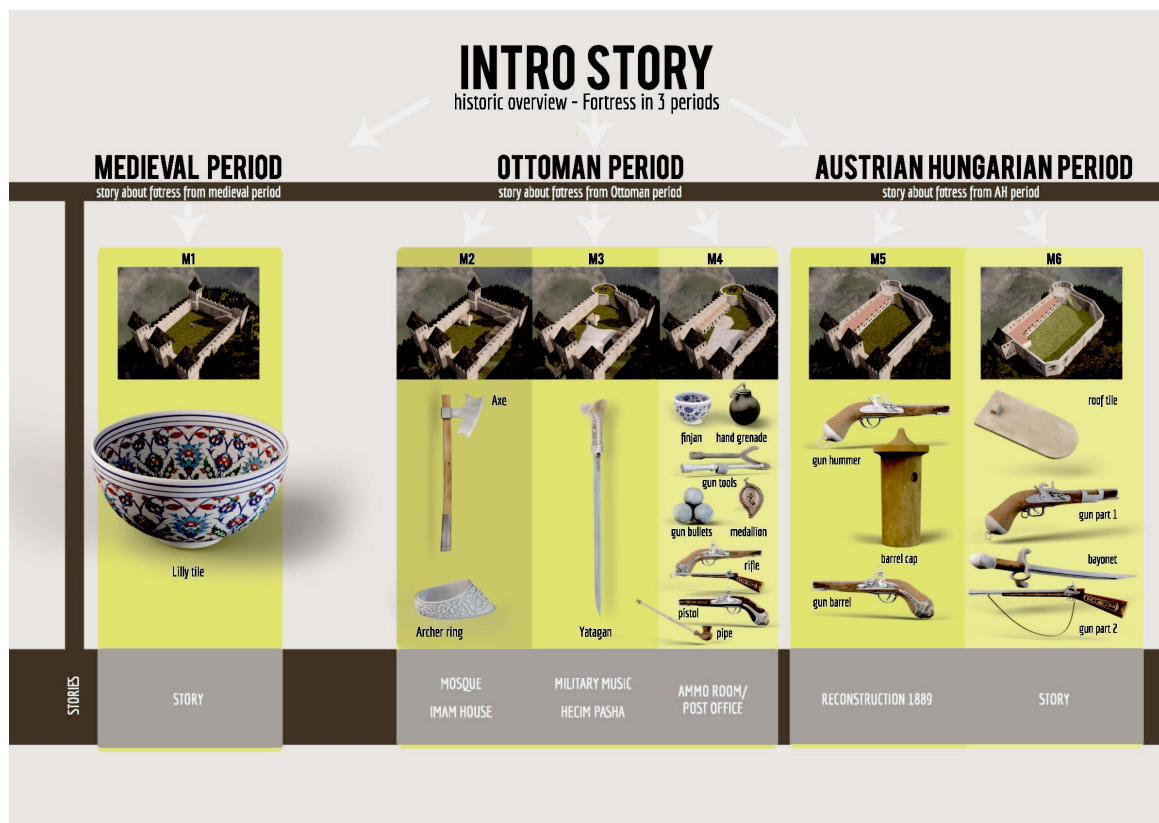


Figure 4: Project structure



Figure 5: A 3D object web page from White Bastion project

- [7] Giulio Martins, Fabio Dominio, and Pietro Zanuttigh. Hand gesture recognition with leap motion and kinect devices. ICIP, 2014.
- [8] Leigh Ellen Potter, Jake Araullo, and Lewis Carter. The leap motion controller: A view on sign language. 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration, 2013.
- [9] Adam Ranfelt. Build a simple 2d physics engine for javascript games. IBM developerWork, 2012.

Generation of lecture notes as images from recorded whiteboard and blackboard based presentations

Ondrej Jaribka*

Marek Šuppa†

Supervised by: Zuzana Černeková‡

Faculty of Mathematics, Physics and Informatics
Comenius University
Bratislava / Slovakia

Abstract

With raising amount of e-learning materials such as lecture videos or on-line video courses, we decided to develop an application, which can help students or content creators in their effort to prepare study materials. Main goal of our application is to create slides from given video depicting a black or white board without any occluding objects such as lecturer standing in front of this board. Slides will contain valid information from key frames of the given lecture video. Based on the assumption that the board is static in the video, this is done by extracting the board from video frames, which is then segmented into equally sized rectangular cells. These cells are stored and the change of information inside them is tracked. Afterwards, the final image is created from saved cells when all cells are sufficiently stable.

Keywords: whiteboard, blackboards, lecture, generation of images, slides, e-learning, video presentation, perceptual hashes, board extraction, key frame generation, slide generation

1 Introduction

With increasing amount of lectures, courses and other e-learning materials available on-line, it is becoming more and more apparent that students as the primary consumers of such content lack tools necessary for its usage in an effective fashion. Given the rise of massive open on-line courses [9] a strong push for creation of instructional videos can also be seen in academic environments. As stated in [12]: “fast expansion of the Internet and related technological advancements, in conjunction with limited budgets and social demands for improved access to higher education, has produced a substantial incentive for universities to introduce e-learning courses”. In [11] the authors also claim that “many users stop their on-line learn-

ing after their initial experience”. They further state that “instructor attitude toward e-Learning, e-Learning course flexibility, e-Learning course quality, *perceived usefulness*, *perceived ease of use*, and diversity in assessments are the critical factors affecting learners’ perceived satisfaction”.

It is not difficult to find inefficiencies in the ways instructional video content is most often consumed. For example it is very impractical to always rewind on-line lecture to get to the exact point where specific detail was discussed, pause the video, essentially “copy” the information from the paused video frame and then continue watching. Not only does it break the user’s focus but it also requires additional interaction with the video content, that might result in undesired increase of the user’s frustration, which is certainly not desired.

For this reason we decided to develop a tool that could help potential viewers extract information from these presentations or lectures into a more practical format. This could also help content creators to prepare and create content which can be then perceived more easily. As stated in [11] *technology* is one of the main factors affecting user satisfaction.

In our work, we focus on whiteboard and blackboard based lectures or presentations. The methods we present select key frames, from which a slide containing the information previously shown on board is created. Another requirement we enforce is that these slides should contain only information that is on the board, and therefore any occluding objects ought to be removed from the scene and information behind them should also be shown on our final slide.

This paper is structured as follows: in the *Related work* section we present an overview of academic literature, that touches the problems our work is designed to solve. Then we proceed to describe methods for *Board extraction*, *Removal of occluding objects* and *Change detection*. In the end, we present the results obtained thanks to our implementation of aforementioned methods in the *Results* section and finally conclude with a short summary of the most important results and possible directions for future work in

*o.jariabka@gmail.com

†marek@suppa.sk

‡zuzana.cerneкова@fmph.uniba.sk

2 Related work

While our work tries to solve a very specific problem, there are a number of similar projects inspired by the increasing amount of readily available video-based lecture material.

Visual Transcripts [10] aims to create lecture notes from blackboard-style videos. Unlike our work, their system assumes that the video only has a blackboard in it, and that the video is static, except for content which is continually added and a mouse pointer which is used to highlight certain parts of the blackboard. The same type of video is used in another related work [8], in which the authors summarize the input video in form of a single image. Parts of the image function as links to positions in the input video and make it very easy to jump to the precise moment where a specific concept was first introduced.

The focus of many authors is mainly on one or few methods for specific subtasks, that represent the respective parts of our system. For example most papers on removing occlusion events from videos focus on 3D objects or use multiple cameras to generate the final image, as described in [4] and [3]. These methods are quite efficient, especially when implemented on GPUs. They might also be called accurate but they are not quite suited for our purpose.

On the other hand, many approaches are focused on creating fast and efficient methods for recognition of change in images and search for similar or unique objects in images. Various methods were designed in order to solve these tasks, such as using image features to detect similar objects in images or using perceptual hashes [13] to detect significant change. In [2], the authors developed a new method for image hashing, which can be used for fast look-ups of similar images. The thesis of Christoph Zauer [14] focuses on implementation and benchmarking of various image hashing algorithms and methods. While the primary aim of these methods was different, they can be easily adapted for the purpose of searching for change in information in series of frames generated from a video sequence.

Algorithms for detecting a change in scenes of video sequences were also proposed in [6] and [7]. These algorithms search for “drastical” points of change, such as hard cuts or special editing effects¹. Even though these methods were quite useful as a model, we could not adapt them because our work is focused more on fine grained change between multiple frames and longer lasting sequences, where change is being slowly added through the span of multiple sets of frames.

¹ An example of these effects are dissolve or wipe transitions.

3 Board extraction

Creators of instructional videos strive to give their content (be it on black or white board) the majority of visual space the video provides. Despite their efforts, often there are parts of video frames one would not expect to find in a “presentation slide”. Moreover, variability in these parts of video frames might cause issues in further stages of the processing pipeline, as it might be mistaken for the actual content. It follows, that in order to create a “presentation slide” from a set of video frames, only the significant parts of the frame need to be considered.

In this section, we describe main methods used for extracting board from video sequences. In our research we focused mainly on videos with one board present in the video or multiple boards not separated by a bigger gap in between them (Figure 1a). Before we detect the main region of interest where the board is located, we have to decide if we are looking at a white or black board. A simple preprocessing method ran on every input frame is used. This method first converts RGB image into grayscale and then computes a histogram. Since we focus only on white or black boards, the dominant colour from these is selected under the assumption that the area of a board occupies vast majority of given frame. This colour is then accentuated by thresholding the image to only extract colours that are close to our chosen colour. Thresholding value is different for every channel and it is approximated to previously found dominant color of our image. The mask obtained by this method is then used to compute bitwise AND in separated colour channels in order to boost our dominant colour (Figure 1b). After this process one of the proposed method is used to obtain main region of a board.

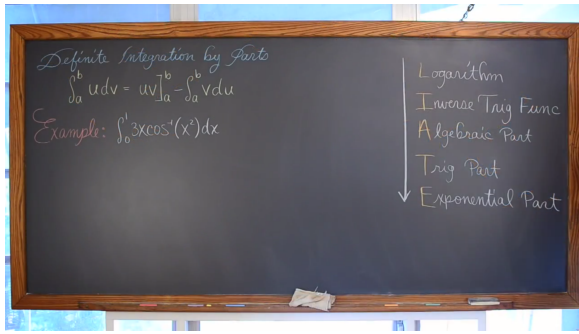
3.1 Histogram method

This method at first reduces colour spectrum of the image by thresholding the dominant colour one more time. In the next step, all the values in every row and column are summed up separately into two arrays. The first derivation is then computed on the acquired arrays to identify spikes in colour change. Extremes of these arrays then specify corners of the board. Finally, a bounding box around board is formed from the obtained points.

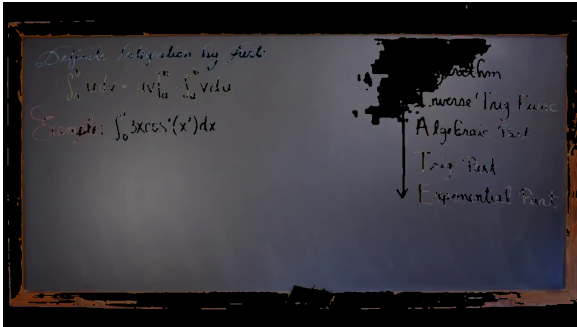
3.2 Region growing method

This method is based on a simple region growing algorithm. At first we have to select a seed for our algorithm. This is done by thresholding the main colour with almost maximum value found from the histogram computed in advance, as part of the preprocessing step. This highlights “blobs” of colour with highest values, so that we can be

²We used the video titled “Definite Integration by Parts” by Rob Tarrou which can be found at <https://www.youtube.com/watch?v=6rWG5WPysgE>



(a) Original image frame from the video.



(b) Output of the preprocessing function applied on the original frame. Note that green colour is accentuated while white or highly illuminated sections are attenuated.

Figure 1: Images of original video² frame and the resulting frame from the preprocessing function

almost sure that we are starting somewhere in the region of a board and not for example on the person standing in front of it. Then, we randomly pick one region where we place our seed. A simple region growing follows, in which we compare 4 neighbouring pixels to check if they fit our colour threshold (if they are part of the board). Once this is done, we obtain mask of a board. We search for contours and extract the biggest one. This contour is then returned as the bounding box of our board.

3.3 Processing of output from board extraction models

Output of every model is then submitted to the last test which checks if area occupied by the the bounding box is at least one third of a given image and if its shape is rectangular. Finally, the image is cropped. If a video or an image is taken from a slight angle, perspective of cropped board is then slightly shifted to compensate for this, so that the resulting slide would look more like an observer is standing in front of a board.

4 Removal of occluding objects

This section describes algorithms that we design in order to remove occluding objects from lecture videos. We consider as occluding object anything that is bigger than at least one third of a board and performs some sort of a move. An example of such an object can be a student or a lecturer. The main idea is to segment the board into smaller sections called cells and then keep track of how information changes in these cells by keeping a simple count of on how many frames we saw individual cell. This process can be described as a sequence of the following steps:

- Divide board into smaller cells
- Initialize each cell
- Compute mask of occluding objects
- Check each cell to see whether it is occluded by another object
- Change “seen” counter for each cell based on occlusion events
- Stitch individual cell into final slide

4.1 Initialization

We divide our cropped image of the extracted board into equally sized rectangular cells based on parameters of the input image. Individual cells are overlapping by values spanning between 10 to 20 pixels based on the size of a given frame. Each cell is then initialized by setting “seen” counter to 0 and its value is stored in an array. When cells are initialized, then we evaluate each of them to check if it is partly or fully occluded by another object³. This is done by computing bitwise AND between section and occlusion mask (see Section 4.2 and Section 4.3). If there is overlap detected the counter is not increased, otherwise it is incremented. For obtaining occlusion mask, we proposed two methods (Figure 2a).

4.2 Region growing based method

The First method is the same as our region growing algorithms since this method only considers pixels that fit our board condition. Pixels representing skin, person or other objects are omitted. This can then specify any objects that are in front of the board. If region growing algorithm was previously used for board extraction, mask produced by this method is then considered as mask of the board. This mask is then inverted and series of dilations and erosions is performed to remove error areas and to accentuate edges. Finally, we search for contours and filter out those, that are smaller than one third of the image.

³Note that if a cell is occluded throughout the whole video we believe that it is save to assume that there is no valid information behind it. Our software provides a way to leave out or add such a cell based on user input.

4.3 Absolute difference based method

The second one uses the last slide which was saved and therefore we assume, that this frame contains valid information and it is without any occlusion events. First, we calculate the absolute difference between current frame and this frame. This generates mask of events that changed from slide to slide. Then, we threshold this mask to remove pixels that arose from slight light changes for example person casting a shadow onto a background. Finally, similarly to the previous method we search for contours and filter out those, that are not at least one third of given image.

4.4 Final slide generation

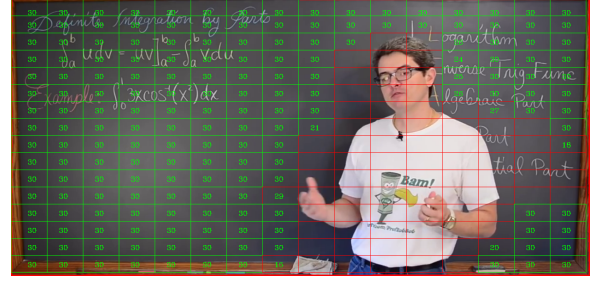
The final slide, shown in Figure 2b, is then “stitched” together from the last saved slide and currently stored sections where we threshold each section’s counter with two values. If the counter value is higher then upper threshold we saw that section enough times to be sowed into the image. If the value is under lower threshold section is rejected and part of the old image is used. If the value is between these two thresholds one of the methods discussed in the section 5 is used to detected how much old and new section differ. If this similarity measure is higher then our threshold then section is rejected because sections where too similar and old part of the slide is used. If it is lower, the section changed enough so it can be sowed into the image.

5 Change detection

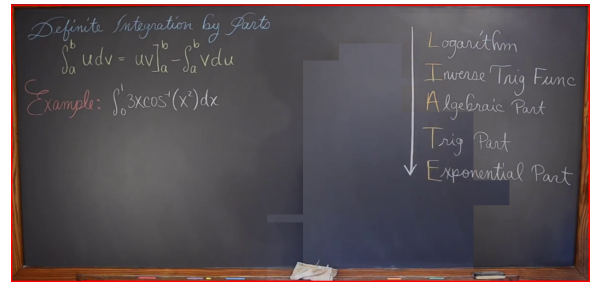
This section focuses on how are we actually going to choose key frames in the lecture or how we detect how much information in processed slides differ. The Main problem, that we faced was how we can choose key frame in the video or presentation. When we can say that enough information was added or subtracted from a board, so we can create slide of given board. For this we developed two main approaches: first one uses feature detection and second computes perceptual hash with one of the specified methods.

5.1 Feature detection

To detect change with this method we used ORB detector for feature detection. First step is to detect features in both last saved slide and current slide. Next, brute force matching, between features of these two image, is performed. Number of matched features is then compared to maximum of found matches. This value is then returned to be latter compared as our similarity measure for further tests.



(a) Input image segmented into grid of rectangular cells. Note that overlap of the individual cells is not shown on the image.



(b) Output of “stitching” algorithm without any further post-processing

Figure 2: This images show main steps in objects removal processing pipeline process.

5.2 Perceptual hashes

Another approach to detect change in our slides is through perceptual hashes. We implemented and compared probably three most known functions.

Average hash is a hashing function which computes hash of a file by firstly converting given image to grayscale. Then reducing a size of image to small square usually of size 8x8 to remove high frequencies. Hash is then created by computing mean value of pixels in transformed image which is then plugged into the thresholding function 1.

$$f(x,y) = \begin{cases} 1 & \text{if } f(x,y) \text{ is } > \text{mean} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Difference hash or dHash, similarly to average hash, computes its value by firstly reducing a size of given image and converting into grayscale, then it calculates difference between adjacent pixels. This identifies relative gradient direction in the image. After this, it determines resulting value by using thresholding function.

$$f(x,y) = \begin{cases} 1 & \text{if } f(x,y) \text{ is brighter than } f(x-1,y) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

While the first two algorithms are quick and easy they might be too rigid in comparison. For instance, it can generate false-misses if there is gamma correction or colour

histogram applied to a image. To reduce this effect we use perceptual hash.

Main idea behind perceptual hash, or more commonly know as pHash, is that it uses discrete cosine transform (DCT) to reduce high frequencies in the image. Same as previous hashes initial step is to reduce size of given image and convert it into grayscale. Then it computes DCT on given image and subsequently reduces it to keep just lower frequencies of the picture. Next step is to calculate the mean DCT value while excluding the first term. This leaves out completely flat image information from being included. Finally, it further reduces DCT and computes resulting hash values based on the thresholding function similar to average hash.

$$f(x,y) = \begin{cases} 1 & \text{if } f(x,y) \text{ is } > \text{mean DCT} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

These hash functions are used to compute hashes of the last saved slide and the currently processed slide. Hamming distance is then computed between these hashes. To unify the output from these functions and feature detection function we compare length of last hash and hamming distance.

Output from these similarity functions is then compared to our similarity threshold. If our measure is lower than this threshold, slides differ and information on the board have to be different, so slide is created. If it is greater, then slides are similar. Current slide is thrown away and all counters are reset.

This section summarizes the results of the proposed methods. We created a dataset of videos and presentations from various on-line lectures and courses. We chose to implement our methods in Python programming language using an Open source computer vision library - OpenCV [1]. The tests run on single desktop workstation, using the following hardware: NVIDIA Corporation GeForce GT 650M, Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz. Preprocessing of every frame in a video, took on average with 300 method calls - 7.02 ms (+- 0.31 ms). Every method was tested for performance from 300 method calls by measuring its individual speed in milliseconds in our processing pipeline.

5.3 Board extraction results

In this section, we present performance values as well as precision and recall values for individual methods.

Method name	speed in ms	sdv
Histogram	4.75	0.65
Region growing	2.03	0.57

Table 1: Average performance values in milliseconds with standard deviation for individual methods

As we can see in Table 1, the Histogram algorithm was on average about two times slower than the region growing

algorithm. This is understandable as the histogram algorithm is much more complex and needs to perform more operations than a simple region growing algorithm.

While this comparison might be interesting it only shows how fast the respective methods are. In order to evaluate the system as a whole we are more interested in their performance: essentially the answer to the question how well were these two methods able to extract the board from input images.

In order to answer that question, we used a dataset of 19110 images. For these images the bounding box of the board was marked by a human expert. This should serve as the “ground truth” in our experiments [5].

To compare these two methods we compute *precision* and *recall* for both of them. We define *precision* as:

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (4)$$

and recall as:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad (5)$$

In both of these equations *true positives* are defined as the area of the image which was marked by the method as a board and the “ground truth” agrees with that. *False positives* is defined as the area that was marked by the method as a board but the “ground truth” did not mark it as a board and *false negatives* is the area which was marked by “ground truth” as a board but the method does not agree with that.

These two values can be put together into a single metric that is called *F1 score* that is defined as the harmonic mean of *precision* and *recall*:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (6)$$

Note that while these methods are traditionally more often used in information extraction, they are also considered a well defined metric in computer vision and image processing [5].

Method name	precision	recall	F1 score
Histogram	0.4245	0.9828	0.5929
Reg. growing	0.9899	0.9356	0.9620

Table 2: Precision, recall and F1 score values of board extraction methods.

Given these metrics, we can observe some interesting statistics about the proposed methods in Table 2. We can see that while the histogram algorithm has a very high recall and therefore produced quite few false negatives its precision is quite low on the other hand. This suggests that it produced quite a lot of false positives which might not be desired for the final processing pipeline.

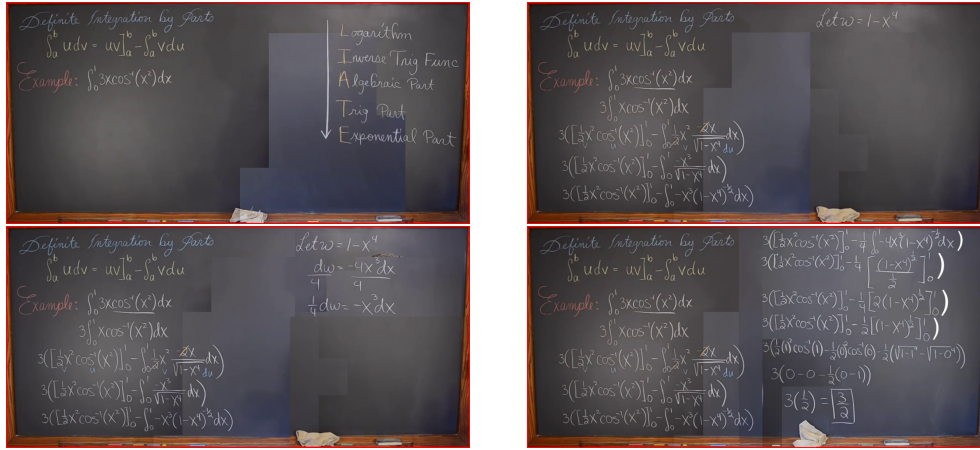


Figure 3: Output images obtained by running the final processing pipeline on an example video.

The region growing algorithm on the other hand shows balanced values for both precision and recall. This suggests that it is a robust method, even though it produced an increased amount of false negatives which might not be desirable, since it would mean that parts of the board would be lost.

As we can see from the comparison of F1 scores, the region growing algorithm seems to be a more robust method and therefore we can conclude that it might be a better choice than the histogram based method.

5.4 Object removal results

While in the previous section the metric for performance comparison of methods was quite straightforward, finding one for object removal has proven difficult. In the end we used a hand-curated dataset of 150 images for which the ground truth (object to be removed) was provided by a human expert.

Method name	precision	recall	F1 score
Reg. growing-based	0.2531	0.5595	0.3485
Abs diff-based	0.4767	0.9877	0.6430

Table 3: Precision, recall and F1 score values of board extraction methods.

The precision, recall and F1 score value for both of the suggested methods can be found in table 3. As we can see, the scores of the absolute difference-based method are better. This can be explained by the nature of the region growing-based method: given its randomized seeding, it might easily start growing the board's region from a point in the middle of the object, that should be removed. This essentially means, that it does the inverse task and produces incorrect result. Looking at the performance values in table 4 we can see that while region growing-based method is faster, the difference is not substantial. Based on this analysis we chose the absolute difference-based method to be used in the final pipeline.

Name	speed in ms	sdv
Reg. growing-based	160.47	0.91
Abs diff-based	195.68	0.84

Table 4: Average performance values in milliseconds with standard deviation for individual methods

5.5 Change detection results

It was difficult to design a measure that would express how well is function performing in terms of selecting key frames in video. This was due to the fact, that even though we can have individual frames in sequence tagged the resulting value, if the selected key frame is in the right place, is very perception dependent. This is why we chose to only measure how many frames will a method create and how long will it take the whole pipeline (with a given measure) to do so.

Name	speed in ms	sdv
ORB algorithm	13.13	0.85
pHash algorithm	7.52	0.76
dHash algorithm	5.32	0.43
aHash algorithm	3.23	0.59

Table 5: Average performance values in milliseconds with standard deviation for individual methods

As we can see in Table 5, the ORB algorithm was the slowest which is understandable as it is also the most complex algorithm. Performance of hashing methods was comparable with fastest one being the aHash algorithm⁴. This can be also closely related to values in Table 6 where, aHash not only has the best performance values but it also managed to create most slides. This can be associated with aHash being one of the the simplest methods that is very vulnerable to even slight light intensity change in the

⁴Which is not very surprising since it performs simple mean on given images.

scene. In a similar fashion, dHash suffers from the same problem, as it also created the same number of slides.

Name	speed (sec)	# of slides
ORB algorithm	1783	7
pHash algorithm	1117	6
dHash algorithm	1115	8
aHash algorithm	1112	8

Table 6: Average performance values in seconds for individual runs of entire pipeline with given method on the whole input video and the number of slides created using these methods.

From values in tables 5 and 6 we can conclude that pHash is the best choice for detecting change in our video sequences. It provides the best ratio between speed of individual methods, speed of the entire run of a pipeline and number of created slides. While feature detection did reasonably well in comparing images, its performance could not compare to hashing algorithms.

6 Conclusion and future work

In our work, we present and compare methods for detecting and extracting boards from white or black board video based presentation. This setup can later be abstracted to detecting and extracting large scale object in image or video that matches some sort of a predicate. In our case, it was colour and shape of an object. We got sufficient results with regards to board extraction with the best method being region growing.

We also tested and evaluated functions for comparing how similar are two images in order to generate key frames in the video sequence. Finally, we proposed methods for extracting information from given presentation even with occluding events present in the sequence. In this category, we also managed to achieve sufficient results with hashing functions with the best one being the pHash algorithm.

We proposed two methods for removing objects in front of our board. The performance of the absolute difference-based one was found to be better overall and chosen for the final pipeline. The slides created by our final processing pipeline can be seen in 3.

Many of these algorithms can be improved. For example after function that creates final slide, we can use post processing method to smooth transition between sections of the old image and the new one. Or it might be possible to further shift perspective in order to better fit the current slide if for instance camera moved during the presentation. Support for multiple boards with bigger gaps between them can be added. Also based on colour of a board, text on slide can be further enhanced for latter used in OCR algorithm.

Given our focus on simplicity, performance and speed, we also believe that the proposed algorithms might serve

as a basis for a system, that would produce slides as images from white and black board based videos in real time.

References

- [1] Gary Bradski et al. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.
- [2] Cédric De Roover, Christophe De Vleeschouwer, Frédéric Lefèvre, and Benoit Macq. Robust image hashing based on radial variance of pixels. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–77. IEEE, 2005.
- [3] Takahide Hosokawa, Songkran Jarusirisawad, and Hideo Saito. Online video synthesis for removing occluding objects using multiple uncalibrated cameras via plane sweep algorithm. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pages 1–8. IEEE, 2009.
- [4] Byung-Gook Lee, Ho-Hyun Kang, and Eun-Soo Kim. Occlusion removal method of partially occluded object using variance in computational integral imaging. *3D Research*, 1(2):6–10, 2010.
- [5] Vladimir Y Mariano, Junghye Min, Jin-Hyeong Park, Rangachar Kasturi, David Mihalcik, Huiping Li, David Doermann, and Thomas Drayer. Performance evaluation of object detection algorithms. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 965–969. IEEE, 2002.
- [6] Jianhao Meng, Yujen Juan, and Shih-Fu Chang. Scene change detection in an mpeg-compressed video sequence. In *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, pages 14–25. International Society for Optics and Photonics, 1995.
- [7] Takafumi Miyatake, Satoshi Yoshizawa, and Hiro-tada Ueda. Method for detecting change points in motion picture images, January 28 1992. US Patent 5,083,860.
- [8] Toni-Jan Keith Palma Monserrat, Shengdong Zhao, Kevin McGee, and Anshul Vikram Pandey. Notevideo: facilitating navigation of blackboard-style lecture videos. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1139–1148. ACM, 2013.
- [9] Laura Pappano. The year of the mooc. *The New York Times*, 2(12):2012, 2012.
- [10] Hijung Valentina Shin, Floraine Berthouzoz, Wilmot Li, and Frédo Durand. Visual transcripts: lecture notes from blackboard-style lecture videos. *ACM Transactions on Graphics (TOG)*, 34(6):240, 2015.

- [11] Pei-Chen Sun, Ray J Tsai, Glenn Finger, Yueh-Yang Chen, and Dowming Yeh. What drives a successful e-learning? an empirical investigation of the critical factors influencing learner satisfaction. *Computers & education*, 50(4):1183–1202, 2008.
- [12] Thierry Volery and Deborah Lord. Critical success factors in online education. *International Journal of Educational Management*, 14(5):216–223, 2000.
- [13] Tom Yeh, Konrad Tollmar, and Trevor Darrell. Searching the web with mobile images for location recognition. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–76. IEEE, 2004.
- [14] Christoph Zauner. Implementation and benchmarking of perceptual image hash functions. Master’s thesis, FH Hagenberg, 2010.

Modeling & Simulation

Dynamic Simulation of Virtual Agents and Obstacles in Virtual Cities

Roman Mankovecký*

Supervised by: Fotis Liarokapis

HCI Lab, Faculty of Informatics
Masaryk University
Brno / Czech Republic

Abstract

The aim of this paper is to investigate a simple model for simulating virtual crowds for virtual environments and computer games. This model is based on the Social Forces model and enhanced using Monte Carlo simulation. The focus is given on the behavior of the actual simulation. In this model we can see interactions between virtual agents (virtual pedestrians) in two scenarios, walking towards a path and crossroads. In both scenarios, these agents are avoiding each other, avoiding obstacles and walls in different scenarios like crossroad or narrowed street. Moreover, users can move, scale or rotate these obstacles and place them interactively into the scene.

Keywords: crowd modeling, social forces model, Monte Carlo simulation, virtual agents

1 Introduction

Modeling human behavior of crowds is extremely interesting for a wide range of applications, ranging from games, film effects, simulators, evacuation simulations to urban planning. Understanding the movement of the crowd can help us to improve public places, we can expedite and facilitate the movement of citizens. However, pedestrians in real-life are reacting to a diverse and complicated stimuli that cannot be easily reproduced to computer simulations. As a result, most of the studies that have been up to now done focus on specific scenarios, such as cross-road crossings or evacuations. On the other hand, simple models can be computationally efficient. They are really good solution for simulations of the crowd [1].

Human behavior and interrelationships of humans can be divided into microscopic, mesoscopic and macroscopic [2]. In microscopic model, every single pedestrian is a simple entity, simple agent who is situated in the space in some specific time [3]. Mesoscopic models is accurately observing the behavior of individuals while relatively large number of individuals in the crowd is simulated [3]. On

the other hand, in macroscopic models we can observe flows similar to gas or water flows [3], [18].

This paper aims in investigating the behavior of a mesoscopic model for agents walking towards a path and for crossroads. This is done by implementing the social forces model and agents are represented as cubes. To make their movement more realistic, the approach is enhanced by Monte Carlo simulation. Each agent has specific goal, destination, that has to reach. During this journey, the agent can collide

with different agents (cubes), obstacles (specific cubes agents cannot walk through) but also walls. The rest of the paper is structured as follow. Section 2 describes simple models used for modeling crowds in general. In chapter 3 we discuss implementation of this model in detail. Section 4 presents results and experiments done with this model and finally chapter 5 illustrates conclusions and future work.

2 Background

One of the first attempts to model crowd behavior was based on simulation of the motion of a generic population in a specific environment [4]. The individual parameters were created by a distributed random behavioral model which is determined by few parameters.

Nowadays, simple but effective crowd modeling systems include a decision-making component [5], pathfinding navigation [6], [17], [19] and local steering mechanics [17]. Another popular approach is the social forces model [18] or 'agent-based' model [19], [10], [11] which can be used to describe the forces of an virtual avatar (agent) to perform movement.

Monte Carlo method for simulating the dynamics of crowds has been used previously in different occasions. In one approach, Monte Carlo dynamics were used for the rearrangement of the group of agents [12]. The rules for the combined steps are determined by the specific setting of the granular flow from stampedes in panic scenarios to organized flow around obstacles or through bottlenecks.

Numerical simulations based on a Monte Carlo particle method demonstrated that when applied to crowds it

*mankoveckyroman@gmail.com

has the capability to qualitatively depict emerging behaviors and to provide a realistic description of the crowd dynamics in complex evacuation scenarios [13]. In another work, computation of escape probabilities using Monte Carlo method was presented for evacuation simulations in the context of the fire safety engineering [13].

An evacuation model using game theory combining the greatest entropy optimization criterion with stochastic Monte Carlo methods to optimize the congestion problem and other features of emergency evacuation planning was also proposed [15].

Moreover, macroscopic models simulate a particular pattern of moving crowd (Figure 1) [3]. Since the model is only interested in the overall look of the crowd, not interested in individual attitudes of individuals in the crowd and their behavior in the crowd, it is possible to simulate the abstract behavior of the large crowd.

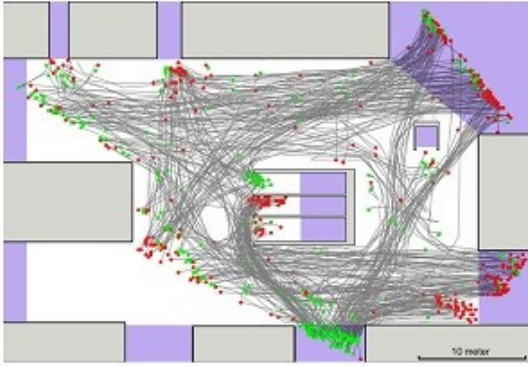


Figure 1: Flow (pattern) of agents [16]

The model uses basic concepts of physics, like the flow or movement of the particles. It does not include the individual behavior of the individual, does not deal with its characteristics and does not address the interaction or collisions with other individuals in the crowd. Thus it is engaged by simulating a large number of individuals in the crowd, respectively density of the crowd. These models have the advantage in terms of computational burden because there is no need to address the logic and behavior of individuals in the crowd. In particular, mesoscopic models are accurately observing the behavior of individuals while relatively large number of individuals in the crowd is simulated.

Mesoscopic models are based on cellular automata [3]. The area of movement has a regular square grid, where each cell may contain one or zero individual or an obstacle for one simulation step (Figure 2 on the right). Every individual has the opportunity to avoid with eight different directions, which are predefined and fixed (Figure 2 on the left). This behavior is not so suitable for a realistic simulation, but on the contrary, in this model, the individuals can move easily everywhere in the area because of pre-defined directions [16].

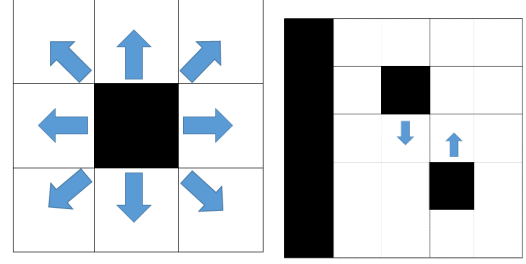


Figure 2: On left: Possible movement of one individual, black cube in the middle is an individual. On right: square grid, where we can see two individuals and a wall.

3 Implementation

3.1 Social force model

As mentioned earlier, to examine the behavior of the crowd simulation, the Social forces model was implemented allowing dynamic interactions between obstacles and agents using other stochastic techniques like Monte Carlo.

Cubes are representing the agents used in the project (Figure 3). Each agent has starting point and a point where it 'dies', which is the destination. Speed of agent is randomly chosen at start (0.5/ 1/ 1.5).

3.1.1 Speed

Speed is changing in some scenarios when agent has to slow down, because he has other agent with slower speed (v_p) ahead and he cannot overtake him in that specific time.

$$v(t) = \begin{cases} v^p(t), & \text{if } v(t) - v^p(t) \leq 0.5 \\ v(t), & \text{if otherwise} \end{cases} \quad (1)$$

Agents are moving forward with speed v in time t . We can define the force that is applied to agents $F_{(X,0)}^F$.

3.1.2 Changing directions

The next step required to add two rays (Figure 4a, red arrows) which are pointing ahead (leftRay, rightRay), so they can predict if something is ahead (i.e. other agents or obstacles). Next, we can define Y from $F_{(0,Y)}^{CH}$, which represents force applied while changing direction in specific time.

$$F_{(0,Y)}^{CH}(t) \rightarrow Y \quad (2)$$

$$Y = \begin{cases} 1, & \text{if leftRay} \neq \text{null} \\ -1, & \text{if rightRay} \neq \text{null} \end{cases} \quad (3)$$

Because of these rays, agents can change direction: if the left ray is hitting agent or obstacle and right one is hitting nothing, that means, agent will change direction to the right ($Y = 1$) and vice versa ($T = -1$).

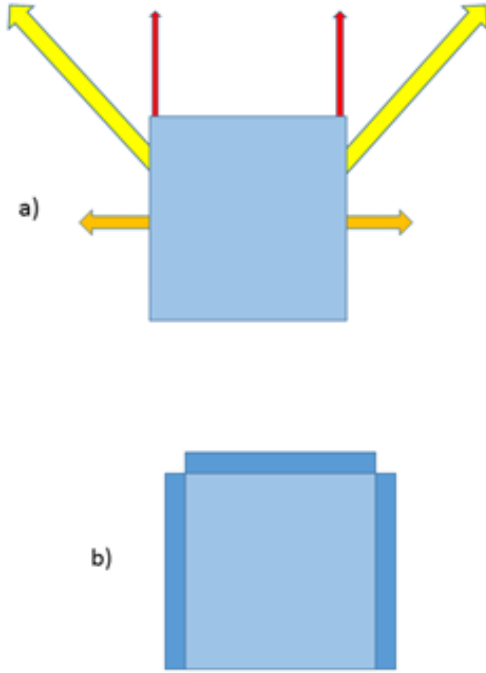


Figure 3: a) Blue cube is representing agent in real life, arrows in this picture are representing rays: red arrows are checking if something like obstacle or other agent is ahead, yellow arrows, “obstacle rays”, are checking obstacles and walls for better orientation and steering, orange arrows are mainly for checking, if something is around agent, so he can overtake. b) blue cubes around agent (cube in the middle) are colliders, they are checking if somebody or something is really close to the agent, they are creating comfortable zone.

3.1.3 Colliders

Agents have colliders as it can be illustrated from Figure 4b. These colliders are situated on the left, right and front side of each agent. They are checking, if other agents or obstacles or walls are nearby. Basically, these colliders are creating area around agents, where they are ‘feeling comfortable’. It is repulsive force $F_{(X,Y)}^R$ that is applied on other agents.

3.1.4 Obstacle rays

Secondly, “obstacle rays” were added, with different angle ($1/4\pi, 3/4\pi$), see Figure 7a yellow arrows. These rays are checking if on the right or left side is obstacle or wall. If for example an obstacle is on the right side, and agent will hit the obstacle, the agent will try to avoid it and he will steer left because right ray was hitting the obstacle. In cases where both these rays are hitting obstacle, only the first one is the key one, so the other ray has no function in these cases.

3.1.5 Overtaking

Each agent has also an overtaking parameter, so faster agents are overtaking slower ones. For this overtaking method two more rays were needed (Figure 4a, orange arrows). These rays (R^{left}, R^{right}) are checking, if the agent, obstacle or wall is on the left or right side. Agent will overtake only in cases he has the speed and space, in cases agent has the speed but not the space around, he will wait till there will be some free space. In waiting part he is checking with these rays, if there is or there is not the free space around. In cases we do not have speed, they will slow down only.

$$F_{(0,Y)}^O(t) \rightarrow Y \quad (4)$$

$$Y = \begin{cases} 1, & \text{if } leftRay \neq null, \\ & R^{right} = null \\ -1, & \text{if } rightRay \neq null, \\ & R^{left} = null \end{cases} \quad (5)$$

3.1.6 Total equation

All these forces influence a agent’s decision at the same moment, it can be assumed that their total effect is given by the sum of all forces:

$$F_{(X,Y)}(t) = F_{(X,0)}^F + \sum_P F_{(0,Y)}^{CH} + \sum_O F_{(0,Y)}^O + \sum_P F_{(X,Y)}^R \quad (6)$$

3.2 Monte Carlo

Monte Carlo is a stochastic method based on the use of random numbers and probability statistics to simulate problems [21]. First, we need to determine the probability density function, then perform random sampling from this function. Monte Carlo method allows us to examine complex system. Solutions are imprecise and it can be very slow if higher precision is desired.

In this work, Monte Carlo method was used to simulate random motion of agents. Total equation of the movement in this model is:

$$F_{(X,Y)}(t) = F_{(X,Y)} + F_{(0,Y)}^{MM} \quad (7)$$

Here we can see $F_{(X,Y)}$ which represent movement of the agent during the time t, $F_{(0,Y)}^{MM}$ is Monte Carlo force.

Monte Carlo simulation, forces agent to move with some probability to the left or right or nowhere. This means that during the time our agent has random movement.

$$F_{(0,Y)}^{MM} \rightarrow Y \quad (8)$$

$$array[10] \in (0, 1) \quad (9)$$

$$Y = \begin{cases} random(-1, 1), & if array[\\ random(0, Length(array)) = 0 \\ 0, & otherwise \end{cases} \quad (10)$$

Before the start of simulation, it is possible to set the probability required, in the array. Every single agent is choosing movement each second, if he moves right ($Y = -1$) or left ($Y = 1$) or if he stays without moving

($Y = 0$). So, in the end every frame agent will check and compare number 0 with a number in the array, if they are the same, will move to the right or left, if not will stay at the same position.

3.2.1 Monte Carlo Simulation

In this simulation, a scene with 100 agents with random speed (0.5 - 1.5) is shown. In first simulation, agents do not use the Monte Carlo method (Figure 4). The movement is occurring in a straight line, changing direction only in cases when agent wants to overtake slower ones. These results are not really realistic. However, in the second simulation (see Figure 5) we can see agents walking with Monte Carlo method. More randomness is in this case better and more realistic and it is better for overtaking as well.

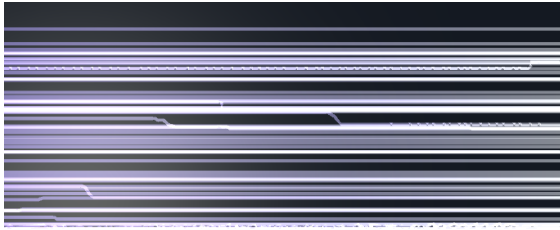


Figure 4: A scenario without Monte Carlo method. We can see lines, these lines are paths of the agents.

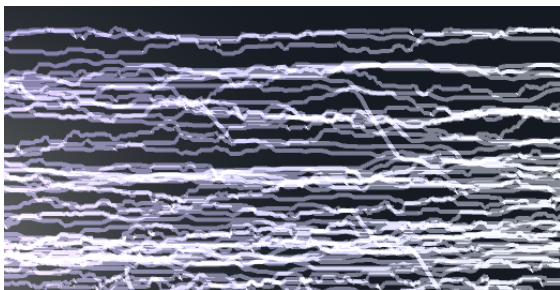


Figure 5: A scenario with Monte Carlo method. We can see lines, these lines are paths of the agents. We can also see the better randomness.

4 Results

In this paper, all these simulations were applied in two specific scenarios: path and crossroad as shown in Figure 6. In the path scenario, we can see some specific interactions (overtaking, grouping) but we can also see interactions between agents and obstacles. On the other hand, in crossroad scenario, we can observe interactions between agents walking from all sides, as well as path choosing.

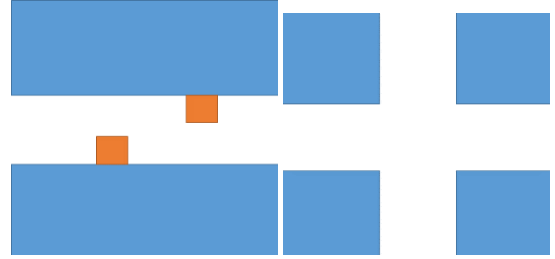


Figure 6: On left: path scene (with orange obstacles), on right: crossroad scene.

4.1 Grouping and tracking trajectories

There are various self-organization phenomena that lead to fascinating collective patterns of motion. For example, when agents are entering a corridor on two sides, we observe the formation of lanes of uniform walking direction [20]. Agents are grouping together, they are following each other, because of better movement, faster movement like we can see in Figure 7. They are avoiding other agents faster because of repulsive forces. This is classic scenario we can see during a normal day, walking the crowded street.



Figure 7: Grouping effect (blue circles), each agent is following the leader. Red agents are going from left to right, green ones are going from right to left.

With more agents, it is possible to track the trajectories of the agents. It is necessary to have more agents, so they can follow each other. As a proof of concept, 100 simulated agents are illustrated in Figure 8. The trajectories of these agents are shown, as well as the fact that they are following exactly the same way as the 'first' agent was moving. With even more agents, it is possible to observe

bigger groups (bigger flocks). In conclusion, this model is showing that flocking is part of our life, we are doing it in normal crowded situations without notice.

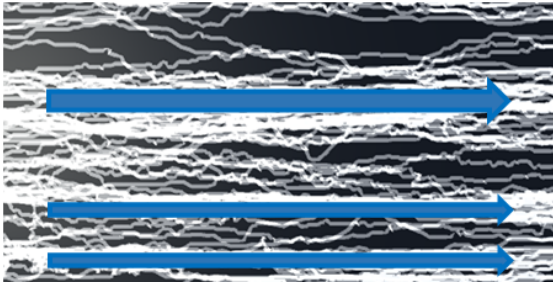


Figure 8: Movement tracking, we can see motion of agents, red arrows are indicating the motion of agents.

Finally, it will be of great importance to perform a user testing to assess how 'realistic' the behavior of the simulation is.

Acknowledgements

Authors would like to thank Human-Computer Interaction (HCI) Lab members for their support and inspiration. A video that demonstrates the functionality of the system can be found at: <https://www.youtube.com/watch?v=4QaOdOVeYlk%20>

4.2 Frame rate measurements

In this section, the effectiveness of the simulation was measured. In particular, the aim was to quantify how many agents this model can simulate. In Figure 9, both tables are showing us frames per second (FPS) in different cases (with 50, 100, 500 and 700 agents in scene). Case 1 is without obstacles, without Monte Carlo method, case 2 is with obstacles but without Monte Carlo method, case 3 is with no obstacles but with Monte Carlo method and in the end we have case 4 with obstacles and using Monte Carlo method. In these measurements I used 2 obstacles. All these measurements have offset error $\pm 1 - 10$ FPS, depending how many agents we are simulating.

In conclusion we can see that this model can simulate big number of agents in different scenarios. It is possible to observe that interactions with obstacles or using / not using Monte Carlo method are causing frame drops, but not so significant. All these measurements were computed on a laptop computer with specifications: Intel Core i7, RAM 8GB DDR3L, NVIDIA GeForce GTX 950M 2GB DDR3.

5 Conclusions

This paper has examined mesoscopic behavior of crowds for virtual environments and computer games based on the social forces model. In this model we can see interactions between virtual agents (virtual agents) in virtual city. These agents are avoiding each other, avoiding obstacles and walls in different scenarios like crossroad or narrowed street. In this model, user can move, scale or rotate these obstacles and place them into the scene.

In the future, it will be certainly useful to implement better steering. Obstacles can have different shapes, not only cubes and of course interactions with agents are sometimes not so realistic. Big bonus can be exchanging cubes with three-dimensional representations of humans.

Framerate measurement				
path	no MC, no obs.	no MC, with obs.	no obs., with MC	with obs., with MC
50	~160	~130	~150	~125
100	~80	~80	~80	~80
500	~20	~15	~18	~15
700	~5	~3	~5	~3

Framerate measurement				
cross	no MC, no obs.	no MC, with obs.	no obs., with MC	with obs., with MC
50	~100	~80	~80	~80
100	~45	~35	~40	~40
500	~7	~6	~8	~7
700	~4	~3	~4	~3

Figure 9: Framerates in different scenarios (path scenario on top, crossroad scenario on bottom).

References

- [1] Stuart O'Connor, Fotis Liarokapis, Jayne Chrisina, *Perceived Realism of Crowd Behaviour with Social Forces*, Proc. of the 19th International Conference on Information Visualisation (IV 2015), IEEE Computer Society, Barcelona, Spain, 494-499, 2015.
- [2] Dirk Helbing, *Models for pedestrian behavior*. arXiv preprint cond-mat/9805089, 1998.
- [3] Kiran Ijaz, Shaleeza Sohail, Sonia Hashish, "A Survey of Latest Approaches for Crowd Simulation and Modeling using Hybrid Techniques."
- [4] Soraia R. Musse, Daniel Thalmann, *A Model of Human Crowd Behavior : Group Inter-Relationship and Collision Detection Analysis*. Computer Animation and Simulation 97, Part of the series Eurographics pp 39-51, 1997.
- [5] Rachel McDonnell, Fiona Newell, Carol O'Sullivan, "Smooth movers: perceptually guided human motion simulation.". In Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 259-269. Eurographics Association, 2007
- [6] Xiao Cui, Hao Shi, "A*-based pathfinding in modern computer games.". International Journal of Computer Science and Network Security 11, no. 1, pp. 125-130. 2011.
- [7] Craig W. Reynolds, "Steering behaviors for autonomous characters.". In Game developers conference, vol. 1999, pp. 763-782. 1999
- [8] Dirk Helbing, Peter Molnar. "Social force model for pedestrian dynamics.". Physical review E 51, no. 5, 4282, 1995.
- [9] Xiaoshan Pan, Charles S Han, Kincho H. Law. *A multi-agent based simulation framework for the study of human and social behavior in egress analysis*. In: Proceedings of the ASCE International Conference on Computing in Civil Engineering. 2005.
- [10] John Funge, Xiaoyuan Tu, Demetri Terzopoulos, *Cognitive modeling: knowledge, reasoning and planning for intelligent characters*. In: Proceedings of the 26th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 1999. p. 29-38.
- [11] Joo E. Almeida, Rosaldo Rosseti, Antnio Lea Coelho, "Crowd simulation modeling applied to emergency and evacuation simulations using multi-agent systems.". arXiv preprint arXiv:1303.4692 (2013).
- [12] Francesco Piazza, *A simple Monte Carlo model for crowd dynamics*. Physical Review E - 82, 02611, 2010.
- [13] Nicola Bellomo, Livio Gibelli, *Behavioral Crowds: Modeling and Monte Carlo Simulations toward Validation*. Computers and Fluids, 2015, In Press.
- [14] Timo Korhonen, Simo Hostikka, Olavi Keski-Rahkonen, *A Proposal For The Goals And New Techniques Of Modelling Pedestrian Evacuation In Fires*. Fire Safety Science 8: 557-567, 2005.

- [15] Wenhui Li, Jinlong Zhu, Huiying Li, Qiong Wu, Liang Zhang, *A Game Theory Based on Monte Carlo Analysis for Optimizing Evacuation Routing in Complex Scenes*. Mathematical Problems in Engineering, Volume 2015 (2015), Article ID 292093, 11 pages.
- [16] Victor Blue, Jeffrey Adler, *Cellular automata model of emergent collective bi-directional pedestrian dynamics*. In: Proc. Artificial Life VII. 2000. p. 437-445.
- [17] Cherif Foudil, et al, "*Path finding and collision avoidance in crowd simulation*". CIT. Journal of Computing and Information Technology 17.3 (2009): 217-228.
- [18] Penny Marno, "*Crowded-macroscopic and microscopic models for pedestrian dynamics*". (2002).
- [19] Miho Asano, Takamasa Iryo, Masao Kuwahara, "*Microscopic pedestrian simulation model combined with a tactical model for route choice behaviour*". Transportation Research Part C: Emerging Technologies 18.6 (2010): 842-855.
- [20] Dirk Helbing, *SOCIAL FORCES: Revealing the causes of success or disaster*. Available from: <http://futurict.blogspot.sk/2014/12/social-forces-revealing-causes-of.html/>, Accessed at: 30/01/2016.
- [21] Investopedia.com, *Monte Carlo Simulation*, [online]. Available: <http://www.investopedia.com/terms/m/montecarlosimulation.asp>, [Accessed: 3-January-2016].

Procedural Generation using Grammar based Modelling and Genetic Algorithms

Karl Haubenwallner*

Supervised by: Markus Steinberger†

Institute of Computer Graphics
Graz University of Technology / Austria

Procedural modeling with shape grammars is a powerful tool to create complex 3D models, but the results are often difficult to control. In this paper we investigate the use of Genetic Algorithms as an optimization algorithm to find a suitable solution for a given target shape. We present a genome representation, a crossover-operator and mutation operators for shape grammars. Furthermore, we demonstrate the feasibility of this approach, using a grammar for spaceships and a volumetric evaluation method, and evaluate the parameters for the genetic algorithm.

Keywords: Procedural Generation, Genetic Algorithms, Shape Grammar, Computer Graphics

1 Introduction

The last years have seen an increasing use of vast open worlds in games such as *The Elder Scrolls* series, the *Fallout* series and *Grand Theft Auto*, among others. Such worlds can increase the immersion during game-play immensely, but creating such large worlds is a time-consuming and mostly tedious task for artists.

Procedural generation and shape grammars offer the possibility to create arbitrarily large and complex worlds and models algorithmically from a small set of rules, thus allowing artists and designers to focus on the narrative and compelling aspect of those worlds. Unfortunately shape grammars are notoriously difficult to control, and small changes in the rules can produce huge differences in the outcome, which only changes the task of creating the world to the equally time-consuming task of finding the right rules and parameters.

Recently there has been some progress in using various methods from machine learning to control the result of procedural generation, where an algorithm takes a shape grammar and a high-level specification, e.g. a sketch or volumetric shapes, as input and generates a derivation tree for the grammar to produce a model that best matches the specification.

While existing work mostly uses Markov Chain Monte Carlo (MCMC) methods and variations thereof, we

present an alternative method using Genetic Algorithms (GA), which have the advantage of creating many equally viable solutions, and therefore providing access to a creative solution process.

1.1 Shape Grammars

One possible approach when creating 3D models is to start with a basic shape (e.g. a cube) and deform and modify it until it resembles the desired model. This is done by applying various operations to the basic shape like translation and rotation or more involved ones like extrusion or splitting, which results in a complex model that bears only a slight resemblance to the initial shape.

Shape grammars try to codify this approach by defining an initial state (called *axiom*), assigning *symbols* to the shapes and defining *production rules*, which declare how to generate symbols and how apply the different operations. This allows them to define a complex model with an initial axiom and a sequence of production rules only. By passing the axiom and the rules to a production system, which applies the operations defined in the rules, an actual model is generated. Since the operations can create new shapes, and the resulting model depends on the sequence of operations applied to certain shapes, the sequence is usually stored as a tree structure, called a *derivation tree*.

1.2 Genetic Algorithms

Genetic Algorithms, a subset of Evolutionary Algorithms (EA), are iterative optimization algorithms with the ability to generate many different, equally viable solutions for any given problem, and therefore provide access to a creative solution process.

The method is inspired by evolution and natural selection, where traits and characteristics of individuals of a species are encoded as genes and chromosomes, and individuals with successful traits get more chances to pass on their genes to future generations, while less successful traits tend to disappear, thus leading to a better adapted population.

GAs follow this process by introducing a *genome representation*, which is an indirect encoding of the problem space. These genes, traditionally symbols of a bit-string,

*karl.haubenwallner@student.tugraz.at

†markus.steinberger@mpi-inf.mpg.de

are then assembled into a *chromosome*, which form an *individual* that represents one possible solution for a given problem. They then go on to create a random set of individuals, which form the initial population, and explore the problem space by evaluating the individuals and assigning each a fitness value, and by selecting the fittest individuals and combining them to form a new generation of possible solutions, which in turn serve as the parents for the next generation.

2 Related work

Shape Grammars Shape grammars were first introduced by Stiny [20], and Lindenmayer [7] used them to create plant models using algorithms. They were expanded with various operators by Stiny [21] and Wonka et. al. [23] among others. Muller et al. [9] introduced the shape grammar *CGA Shape* to generate architectural models on a large scale by iterative refining shapes from a basic vocabulary, and Schwarz et al [15] expanded *CGA Shape* with *CGA++* by introducing boolean operators, and simultaneous operations on groups of shapes.

Procedural modeling There have been several works with procedural modeling using shape grammars, such as generating road networks [12], or generating and rendering architecture and cities on the GPU [6, 19], and several works using inverse procedural modeling, such as recreating trees with biological models [18], or creating derivation trees for shape grammars using MCMC methods [14, 22], or using constraint systems [8].

Genetic Algorithms and evolutionary computing GAs were introduced by Holland in [5], and in turn have been adapted to a wide range of problems, several of which make use of the inherent creativity, such as creating and evolving simulated lifeforms (Sims, [16]), or designing radar antennas (DeJong et. al. [3]). There have also been some applications using GAs to evolve shape grammars in 2D (O'Neill et. al. [11]), or improve the structure of power pylons (Byrne et. al. [2]).

3 Approach

In this paper we use GAs as a method to stochastically explore all possible derivation trees for a given grammar and select those that best fit a given criteria.

The GA requires only minimal explicit knowledge about the rules of the grammar and can optimize towards any criteria that can be used to rank the derivation trees, and can provide many different viable solutions.

To illustrate our approach we focus on a grammar that creates spaceship models by accumulating geometric shapes, and use the volume of a target model as an optimization criteria. To ensure the volume of the target model

is possible to reach, we generate it using the same production system and a fixed derivation tree, but any target volume could be used.

3.1 Genetic Algorithm

Key points for the functionality of GAs are the distinction between genome encoding (*genotype*) and the expression of the genomes in the problem space (*phenotype*), and genetic operators. Operators modify the chromosomes of individuals without necessarily having any information about how the modifications affect the solutions. This distinction allows the GA to operate on a variety of problems, but requires the definition of genome representation and genetic operators for each specific problem. In this chapter we specify the genome representation and the genetic operators, and give details about the implementation of the GA.

3.1.1 Genome Representation

The genome representation should encode complex operations in the problem space in a way that allows the GA to identify and combine building blocks for good solutions, while being as simple as possible, to keep the chromosomes manageable. At the same time, since the fitness evaluation of a given individual usually requires a translation of the genes into their expression, they should also be easy to decode.

For shape grammars, the traditional approach of using bit-strings of fixed size is somewhat limited, but the structure of the derivation tree facilitates these features, so we use it as our genome representation. This representation is similar to the ones used by [10, 16, 22], where a tree structure is encoded within a chromosome in various ways.

In our representation a single gene consists of a production symbol and its parameters, and a reference to the parent gene within the tree structure. A vector of genes form a chromosome, which can be expressed as a derivation tree and used by a production system to generate a model.

This structure also allows for a variable length of the chromosomes and easy insertion of new genes into the chromosome without changing the already existing entries.

To allow the operators to keep the generated or modified chromosomes within the confines of a valid derivation tree, each possible type of gene is defined by a symbol descriptor (Fig. 1), which contains the possible child symbols, how likely they are to be generated during mutation and a description of the parameters. The information in these descriptors is inferred from the rules for the given grammar.

3.1.2 Crossover Operator

The crossover operator produces a viable pair of children given a pair of parents. The simplest form is the single-

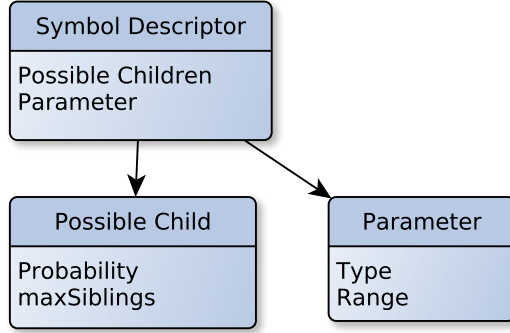


Figure 1: Structure of the symbol descriptor.

point-crossover, which selects a random crossover point in both parent-chromosomes and creates the offspring by swapping the genes after the crossover points. The number of possible different children given the same set of parents is limited by the number of valid crossover points.

Our operator uses a variation of the single-point-crossover operator, adapted to tree structures. It selects a random gene connection from the first parent, and chooses another random connection from all compatible connections in the second parent, and exchanges the genes at these connections. This operation is outlined in Fig. 2, and presented in more detail at Algorithm 1.

3.1.3 Mutation Operator

Mutation allows the GA to explore the problem space outside the already existing population by introducing random changes in ways that are very unlikely to occur by using crossover operators alone. Our operator uses the following changes that arise intuitively from the tree structure of the chromosomes:

- **Grow:** Adds a suitable gene as child of a random gene and initializes the parameters. This doesn't replace already existing genes.
- **Cut:** Removes a random gene and all child-genes.
- **Permutate:** Change the parameter values of a random gene.

When the operator is applied, one of these changes is chosen at random.

3.1.4 Selection Methods

The selection method is one of the central parts of GAs, as it allows the algorithm to select good parents for the new generation, while at the same time denying bad solutions the chance to reproduce. This is usually done by selecting the individuals according to their fitness values, with some margin for error.

With a purely deterministic selection method the same parents would be selected again and again, thereby limiting the exploration of the problems space to the proximity of the fittest individual of the initial population, whereas a purely random method would lead to an entirely

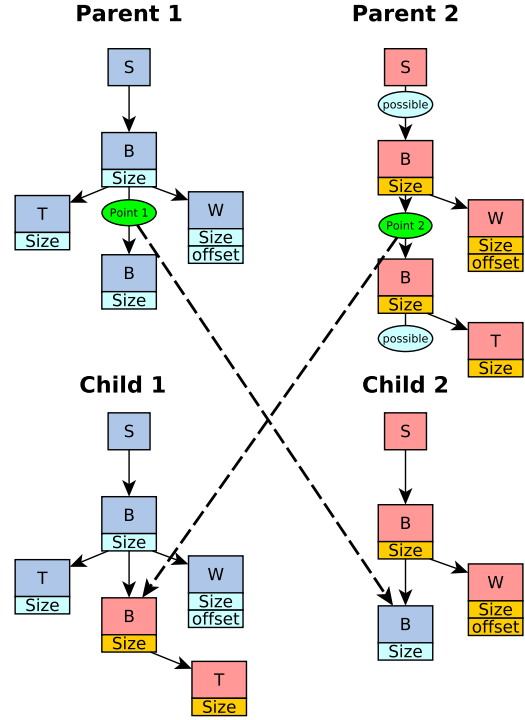


Figure 2: The crossover operator. For a random connection in the first parent, a fitting connection in the second parent is selected, and the offspring is generated.

undirected exploration, which decreases the probability of finding a good solution considerably. There are several possible selection operators, as compared by [1], but the ones most widely used are roulette wheel selection and k -tournament selection.

Roulette Wheel selection calculates the probability p_i that the individual i is chosen such that it is proportional to its fitness value f_i in relation to the overall fitness of the population:

$$p_i = \frac{f_i}{\sum_{j=1}^N f_j} \quad (1)$$

The name stems from the informal description of the method as a roulette wheel, where the size of each possible spot relates to the fitness of the individual occupying it.

k -tournament selection consists of selecting k individuals at random from the population, and choosing the individual with the highest fitness value, i.e. letting them fight in a tournament. This method is quite fast, and the selection pressure can be increased by increasing the size of the tournament.

Algorithm 1 Crossover operator

```
procedure CROSSOVER(parent1, parent2)
  while tries < Max Retries do
    Point1 ← Random connection from parent1
    C ← SELECTPOSSIBLE(parent2, Point1)
    if C is empty then tries ← tries + 1
    else
      Point2 ← random connection from C
      child1 ← CLONEUNTIL(parent1, Point1.start)
      child1 ← CLONEFROM(parent2, Point2.end)
      child2 ← CLONEUNTIL(parent2, Point2.start)
      child2 ← CLONEFROM(parent1, Point1.end)
      return child1, child2

procedure SELECTPOSSIBLE(parent2, Point1)
  for all Connections C in parent do
    if C.end is possible Child of Point1.start then
      // Count siblings, including the current one
      S ← Number of siblings with type of C.end
      // Check if there can be more genes of this type
      Dc ← PossibleChild (Fig. 1) of C.end
      if Dc.maxSiblings == S then
        connections.add(C)
      else
        P ← random percentage value
        if P < Replace probability then
          connections.add(C)
        else
          Cnew ← new connection
          Cnew.start ← C.start
          Cnew.end ← none
          connections.add(Cnew)

  return connections
```

3.1.5 Implementation Details

There is a large variety in the details of GAs, which differ slightly in each implementation. The variants used for this paper are as follows:

The Initial Population is created by repeatedly applying the grow-mutation operator to initially empty chromosomes. The GA converges faster if the chromosome-length of the initial population is comparable to the desired target, but the length is self-correcting to a large extent.

A new generation is created by selecting two individuals from the population, and either applying the crossover-operator or the mutation operator to both individuals.

Additionally we use **elitism**, whereby some individuals with the best fitness values are copied unchanged to the new generation, but are still used as parents for crossover. This ensures that the quality of the solution never decreases, and can also improve the quality of the solution, since good individuals are preserved and produce more

offspring. But if the population size is too small, this can lead to stagnation.

3.1.6 Fitness function

The fitness function evaluates the quality of a single individual and assigns a fitness value to it. Since the GA only optimizes the fitness value, the used fitness function very much defines the visual quality of the solutions. It also should not be defined too restrictive, to allow the GA to explore non-optimal solutions. Additionally one has to pay attention to the complexity of the function, since the fitness calculation is often the most time consuming task of a GA. There are many possible different fitness functions for shape grammar, like evaluating the silhouette from a certain perspective, or several volume-based approaches, such as filling or avoiding a given volume.

We use a volume-based fitness function, where we generate the model using the derivation tree defined by a chromosome, and compare the volume of the model to a given target volume. The comparison is done by converting the generated model into voxels using a basic ray-based voxelisation method and counting the voxels. Then, with v_{target} as the number of voxels of the target volume, v_{inside} as the number of generated voxels that fall inside the target volume, $v_{outside}$ the number of voxels outside the target volume, and $v_{overlap}$ as the number of voxels that are self-overlapping within the generated model, the fitness value f is calculated with

$$f_{good} = step(v_{inside}, 0, v_{target}) \quad (2)$$

$$f_{bad} = step(v_{outside} + v_{overlap}, 0, 2 \cdot v_{target}) \quad (3)$$

$$f_{length} = step(l, l_{opt}, l_{max}) \quad (4)$$

$$f = \alpha \cdot f_{good} - \beta \cdot f_{bad} - \gamma \cdot f_{length} \quad (5)$$

with α, β, γ as weights. Furthermore l is the length of the chromosome, and l_{opt} and l_{max} are given parameters, since it has been shown by [17] that including the length of the chromosome in the fitness calculation is a good way to prevent it from growing considerably, which would increase the evaluation time.

Finally we use a *step* - function to provide a normalization of the fitness value, based on the *smootherhstep* - function defined in [13].

$$step(x, min, max) = \begin{cases} 1 & \text{if } x \geq max \\ 0 & \text{if } x \leq min \\ 6t^5 - 15t^4 + 10t^3 & t = \frac{x-min}{max-min} \end{cases} \quad (6)$$

This is the most time-consuming step of our implementation, but by using an efficient implementation on the GPU and an variant of *CGA-Shape* previously used by [19], we were able to keep the calculation time manageable.

4 Evaluation

To evaluate the method presented in this paper, we implemented a framework using C++ and CUDA.

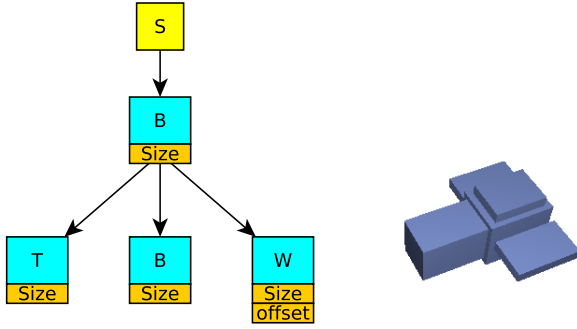


Figure 3: Basic symbols of the grammar and the generated model using fixed values for the parameters.

4.1 Spaceship Grammar

The implemented grammar is inspired by [14], and is designed to produce simple spaceships with wings. It produces axis-aligned boxes of varying sizes, and consists of a start symbol **S**, and three terminal symbols **B**, **W** and **T**. The rules are not explicitly required for this method, but can be outlined as follows:

$$\begin{aligned}
 S &\rightarrow B \\
 B &\rightarrow BB \mid BW \mid BT \mid \varepsilon \\
 W &\rightarrow WW \mid \varepsilon \\
 T &\rightarrow TT \mid \varepsilon
 \end{aligned}$$

The three basic symbols relate to the following shapes:

- **B**: central spaceship-body, attached to the parent symbol along the main axis.
- **W**: wings, mirrored at the main axis and attached along the secondary axis.
- **T**: top, attached along the third axis.

A basic example is shown in Fig. 3.

4.2 Parameter Selection

The selection of parameters for a GA is a very complicated task, since the parameters are interdependent, e.g. a high mutation rate can produce good results, but only if the population size is large enough. There have been various attempts to optimize this selection, such as using statistical models [4], or even using other optimization algorithms to find the best set of parameters, which introduces the problem of finding parameters for that algorithm. Since the execution time of our implementation is manageable, we were able to find a good set of parameters by changing one parameter at a time and comparing the results. The parameters during our evaluation are fixed to the following baseline, unless stated otherwise:

Population size:	50 individuals
Initial length:	10 symbols
Max. generations:	50
Elitism:	1 individual
First selection:	roulette wheel
Second selection:	k -tournament, size 10
Mutation prob:	30%
Mutation operator:	cut/grow/perm. uniform distr.
Crossover retries:	3

Due to the simplicity of the grammar, the crossover operator was able to produce an offspring reliably, with only about 0.2% of all cases requiring at most two tries.

The parameters for the fitness calculation do alter the look of the generated models, but do not alter the behavior of the GA significantly. They were set to the following values:

$$\begin{aligned}
 \alpha &= 1 & \beta &= 0.8 & \gamma &= 0.2 \\
 l_{opt} &= 40 & l_{max} &= 100
 \end{aligned}$$

While the target can be an arbitrary volume, to ensure it is reachable we created it using the same grammar with a fixed derivation tree. The random number generator (RNG) was the uniform distributed mersenne-twister implementation provided by c++11 (mt19937). All the generated values are averaged over three discrete runs.

4.2.1 Selection method

The available selection methods are a roulette wheel selection, a k -tournament selection of size 10, and random selection. As shown in Fig. 4, using a semi-stochastic method for at least one parent produces better results than purely random selection, with tournament selection performing better. The best results were produced by a combination of tournament and roulette wheel selection, although initially tournament selection for both parents increases the fitness values slightly faster.

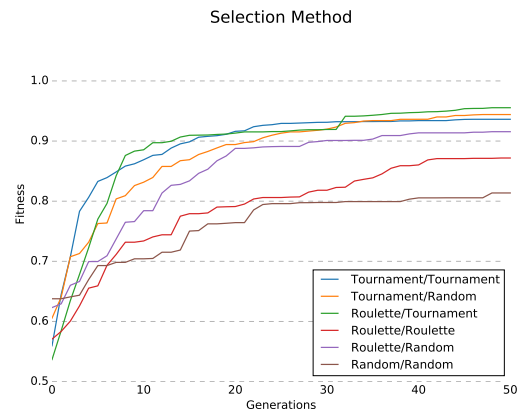


Figure 4: The fitness values with different selection methods.

4.2.2 Population size

Since the GA recombines parts of already existing solutions, a bigger population increases the chance to combine two good parts to create a better solution, and it also increases the probability that an individual already has a good fitness value from the beginning, therefore increasing the fitness of the solution immediately. Unfortunately, an increase in the population size also increases the execution time significantly, which requires finding a trade-off between speed and fitness. When increasing the population from 10 to 500 individuals, as shown in Fig. 5, the fitness increases as well, but after a size of 200 individuals the increase is negligible compared to the increased execution time.

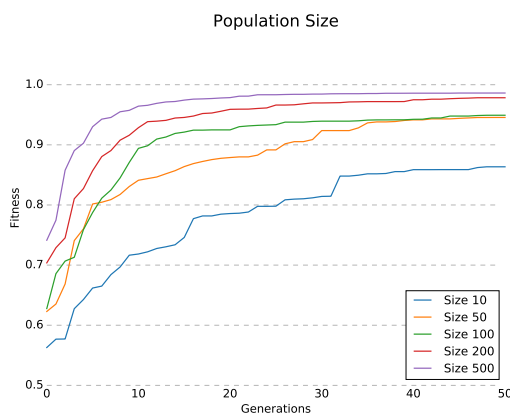


Figure 5: The fitness values with increasing population size.

4.2.3 Mutation probability

If the mutation rate is too low, the probability to produce beneficial changes is low as well, while a high mutation rate can introduce disadvantageous changes to already good solutions. This can be mitigated to some extent with elitism, which can introduce another set of problems. When the mutation rate is increased from 0% (only crossover) to 100% (only mutation) (shown in Fig. 6), the fitness increases as well, although the difference is only significant in later generations.

Using only mutation produces good results, but we found in further evaluation that the effect diminishes with a higher population size.

4.2.4 Elitism

Elitism allows the GA to explore the problem space surrounding good solutions by keeping them unchanged from one generation to the next, while still using them as parents. A small elitism rate in a large population can lead to a replacement of the elite in every turn, thus having no impact at all, while a large elitism rate can lead to stagnation. By changing the elitism rate from 0 to 45 individuals

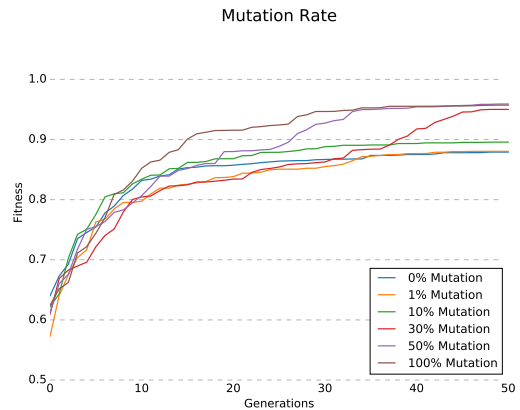


Figure 6: The fitness values with increasing mutation rate.

(90%), as shown in Fig. 7, we find that the use of 10 to 15 individuals (20 - 30%) produces the best results, although the impact is not very significant. It does, however, ensure a steadily increasing fitness value. We also found in further evaluations that the impact of elitism is highest with small populations, and diminishes with increasing population sizes. And, as expected, keeping a large part of the population as elite does decrease the quality of the result significantly, and also leads to periods of stagnation.

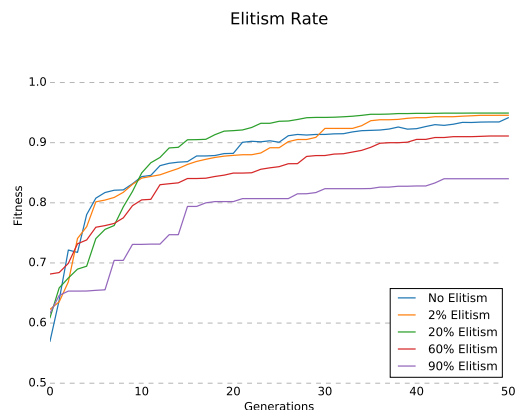


Figure 7: The fitness values with increasing elitism levels.

4.2.5 Initial length

The length of the initial population was increased from 10 symbols up to 100 symbols (Fig. 8). Due to the influence of the length on the generated model, this significantly alters the fitness of the initial population, which in turn impacts the performance of the algorithm. But we found that the impact diminishes with increased generations.

4.2.6 Final parameters

Overall we found that the population size and the use of any selection method other than random selection have the

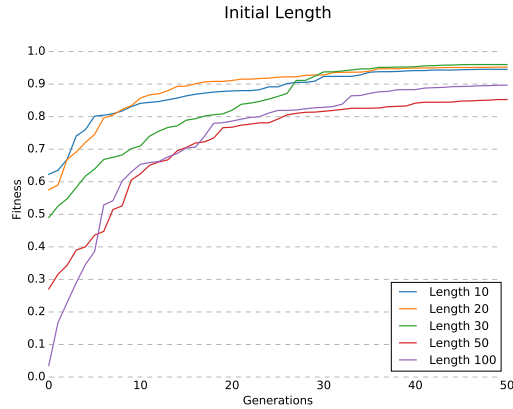


Figure 8: The fitness values with an increasingly complicated initial population.

largest impact on the quality of the result, while other parameters have the most impact during the first few generations. Thus we were able to generate good results using the following parameters:

Population size: 200 individuals
Initial length: 20 symbols
Max. generations: 50
Elitism: 30 individuals
First selection: roulette wheel
Second selection: k -tournament, size 10
Mutation prob: 30%
Mutation operator: cut/grow/perm. uniform distr.

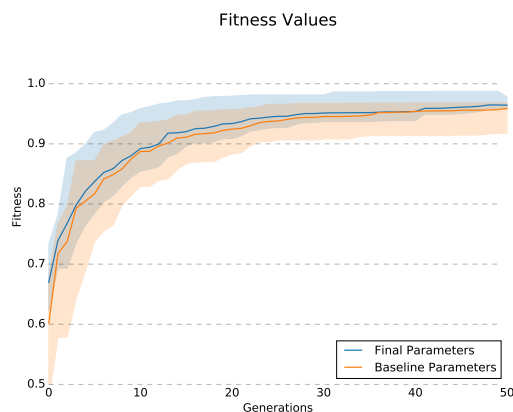


Figure 9: The fitness values for the baseline and final parameters.

The results of the baseline and the final parameters are very similar (shown in Fig. 10), since the baseline parameters already produce good results, but the final parameters tend to perform more reliably. But because the bigger population creates a longer execution time, the parameters can be optimized with regards to time for specific targets.

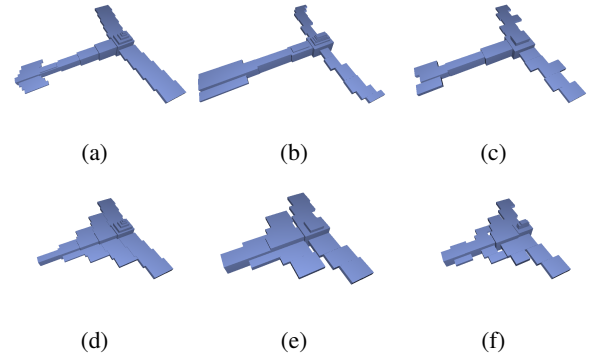


Figure 10: Targets and results of various runs: *a* and *d* are targets, *b* and *e* were generated using the baseline parameters, and *c* and *f* were generated with the final parameters.

Population	Evaluation (avg)	Crossover (avg)	total
50	0.25sec	2ms	12.38sec
200	0.9sec	11ms	45.63sec

Table 1: Average evaluation and reproduction times per generation, and the total duration for one run.

4.3 Performance

The results of the performance evaluation are shown in table 1. Most time is spent calculating the fitness values, while the time for generating a new generation is minimal. But since most of the evaluation time is spent on generating the model from the derivation tree, this time could be improved by moving the production system to the GPU, and exploiting the inherent parallelism of GAs.

These values were achieved on an Intel Core i5-5200U CPU @ 2.20GHz with 8 GB of RAM and a Nvidia Geforce 840M.

5 Conclusion and Future Work

We presented a genome representation and genetic operators that are suitable for an application of GAs to control the derivation tree for shape grammars. Furthermore we demonstrated the basic viability of this approach by presenting the implementation for a specific grammar for simple spaceships and a volume-based fitness function, and evaluated the influence of the parameters required for GAs.

A clear opportunity for future work is the evaluation of this method with different, more complicated shape grammars, since GAs generally tend to perform worse when the complexity of the problem space increases.

Also is the volume-based fitness function very restrictive and limits the creative capabilities of GAs, therefore a different approach for the fitness calculation might be

preferable, such as a image-based evaluations, as used successfully by [14, 22].

Furthermore could an implementation of the algorithm on the GPU improve the performance considerably, since GAs and shape grammars are inherently parallel in nature.

References

- [1] T. Bickel and L. Thiele. A Comparison of Selection Schemes Used in Evolutionary Algorithms. *Evolutionary Computation*, 4(4):361–394, 1996.
- [2] J. Byrne, M. Fenton, E. Hemberg, J. McDermott, and M. O’Neill. Optimising complex pylon structures with grammatical evolution. *Information Sciences*, 316:582–597, 2015.
- [3] C. M. De Jong Van Coevorden, A. R. Bretones, M. F. Pantoja, F. J. García Ruiz, S. G. García, and R. G. Martín. GA design of a thin-wire bow-tie antenna for GPR applications. *IEEE Transactions on Geoscience and Remote Sensing*, 44(4):1004–1009, 2006.
- [4] O. François and C. Lavergne. Design of evolutionary algorithms - A statistical perspective. *IEEE Transactions on Evolutionary Computation*, 5(2):129–148, 2001.
- [5] J. H. Holland. *Adaptation in natural and artificial systems*. 1992.
- [6] L. Krecklau, J. Born, and L. Kobbelt. View-dependent realtime rendering of procedural facades with high geometric detail. In *Computer Graphics Forum*, volume 32, pages 479–488. Wiley Online Library, 2013.
- [7] A. Lindenmayer. Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of theoretical biology*, 18(3):300–315, 1968.
- [8] P. Merrell and D. Manocha. Model synthesis: A general procedural modeling algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):715–728, 2011.
- [9] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *ACM Transactions on Graphics*, 25(3):614, 2006.
- [10] E. Murphy, M. O’Neill, E. Galván-López, and A. Brabazon. Tree-adjunct grammatical evolution. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.
- [11] M. O’Neill, J. M. Swafford, J. McDermott, J. Byrne, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg. Shape grammars and grammatical evolution for evolutionary design. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO ’09*, page 1035, 2009.
- [12] Y. I. H. Parish and P. Müller. Procedural Modeling of Cities. *28th annual conference on Computer graphics and interactive techniques*, (August):301–308, 2001.
- [13] K. Perlin. Improving noise. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’02, pages 681–682, New York, NY, USA, 2002. ACM.
- [14] D. Ritchie, B. Mildenhall, and P. Goodman, N. D. and Hanrahan. Controlling procedural modeling programs with stochastically-ordered sequential monte carlo. *ACM Transactions on Graphics (TOG)*, 34(4):105, 2015.
- [15] M. Schwarz and P. Müller. Advanced procedural modeling of architecture. *ACM Transactions on Graphics*, 34(4 (Proceedings of SIGGRAPH 2015)):107:1–107:12, August 2015.
- [16] K. Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.
- [17] T. Soule, J. A. Foster, and J. Dickinson. Code Growth in Genetic Programming. *GECCO ’96 Proceedings of the 1st annual conference on genetic and evolutionary computation*, pages 215–223, 1995.
- [18] O. Stava, S. Pirk, J. Kratt, B. Chen, R. Měch, O. Deussen, and B. Benes. Inverse procedural modelling of trees. In *Computer Graphics Forum*, volume 33, pages 118–131. Wiley Online Library, 2014.
- [19] M. Steinberger, M. Kenzel, B. Kainz, J. Mueller, W. Peter, and D. Schmalstieg. Parallel generation of architecture on the GPU. *Computer Graphics Forum*, 33(2):73–82, 2014.
- [20] G. N. Stiny. *Pictorial and Formal Aspects of Shape and Shape Grammars and Aesthetic Systems*. PhD thesis, 1975.
- [21] G. N. Stiny. Spatial Relations and Grammars. *Environment and Planning B: Planning and Design*, 9(1):113–114, mar 1982.
- [22] J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. Metropolis procedural modeling. *ACM Transactions on Graphics*, 30(2):1–14, 2011.
- [23] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. Instant Architecture. *ACM Trans. Graph.*, 22(3):669–677, 2003.

Guided 2D Modeling of 3D Buildings using Oriented Photos

Lisa Kellner*

Supervised by: Michael Schwärzler[†]

VRVis Research Center
Vienna / Austria

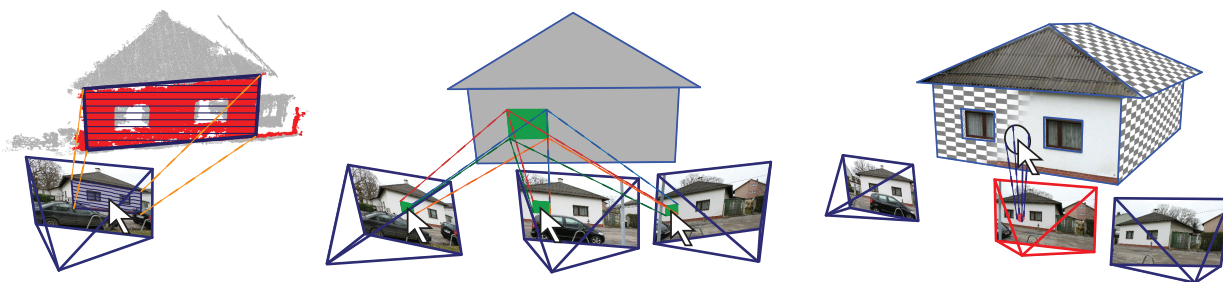


Figure 1: Modeling operations taking both oriented images and point cloud data into account. Left: Point Cloud-supported Single Shot sketching, exploiting planar structures in the data. Middle: Multi-View Shot View sketching. Right: Texturing the generated polygons using an interactive brushing method

Abstract

Capturing urban scenes using photogrammetric methods has become an interesting alternative to laser scanning in the past years. For the reconstruction of CAD-ready 3D models, two main types of interactive approaches have become prevalent: One uses the generated 3D point clouds to reconstruct polygonal surfaces, while the other focuses on 2D interaction in the photos to define edges and faces.

We propose a novel interactive system that combines and enhances these approaches in order to optimize current reconstruction and modeling workflows. Our main interaction target are the photos, allowing simple 2D interactions and edge-based snapping. We use the underlying segmented point cloud to define the 3D context in which the sketched polygons are projected whenever possible. An intuitive Visual Guiding interface gives the user feedback on the accuracy to expect with the current state of modeling to keep the necessary interactions at a minimum level.

Keywords: 3D-Modeling, Guidance, Photogrammetry

1 Introduction

The use of three-dimensional point cloud data of buildings gathered with different sensors is becoming part of the standard workflow in surveying and mapping – may it be from tachymetric devices, laser scans, or photogrammetry. While solutions for acquisition, storage and viewing of the generated point clouds have become commercially avail-

able, the derivation of low-polygonal, CAD-ready models still poses a mostly unsolved challenge.

Only recently, researchers have started tackling this problem by not only using meshing algorithms to triangulate the point cloud, but to detect and use underlying structures first in order to use geometric primitives to represent the building (see Section 2). By doing so, the emerging 3D models correlate a lot more to the way a human artist would reconstruct a building in a 3D modeling tool: Not only is the number of polygons usually considerably lower, they also contain sharp edges and hierarchical definitions, and are therefore a lot easier to manipulate interactively.

Still, reconstructing complete data sets is often hardly possible due to limitations in the point data: Laser scanners cannot be arbitrarily positioned, so that it is common that parts of the data are missing. Tachymetric point clouds are too sparse for reconstruction, and photogrammetric point clouds often have large holes caused by uniformly colored areas in them (see Section 3). This shortcoming in photogrammetric point clouds can be partly compensated by considering the source photographs as well, making it possible to identify and model sharp building edges using line features that can be identified in images more easily.

In this paper, we pursue the concept of using both point cloud and photo data together, but combine it with the approaches from Sinha et al. [18] or Arikan et al. [2] of simplifying interactive 3D modeling to a 2D problem whenever possible: We propose an interactive modeling approach based on oriented photos, in which the user

sketches the desired geometry directly on the image using simple 2D operations, supported by an edge-based snapping feature. Based on the available data quality of the underlying point cloud, the 2D polygon is either directly projected to 3D space whenever possible, or the user is required to define the polygon geometry in multiple further photos, until a unique mapping to 3D space can be defined. This interactive process is supported and guided by providing suggestions for potential polygon candidates in neighboring images, as well as by giving visual feedback on the estimated accuracy for the projection to 3D space. This allows the user to make use of both point cloud and image data, while relying on an optimal, flexible workflow with minimal manual intervention.

2 Related Work

The field of urban reconstruction has gained a lot of scientific attention in the last years. A complete overview is out of scope for this paper, and we refer the interested reader to the recent state-of-the-art report by Musialski et al. [14]. Instead, we put our focus on methods that incorporate photos in the geometric reconstruction pipeline, or that try to simplify the 3D modeling by either reducing the interaction dimensions or providing suggestions to the user.

From a user-oriented perspective, our novel system is most closely related to systems presented by Debevec et al. [5] or Sinha et al. [18]: These interactive tools also rely on image-based modeling operations, and use photogrammetric data sets to calculate geometric correspondences in order to reconstruct 3D geometry. While operations like snapping to edges in images or multi-view texture generation have been integrated in these tools as well, neither of them exploits the availability of the underlying photogrammetric point cloud in order to simplify the sketching progress as in our method. Furthermore, we introduce an additional *guiding indicator* in the graphical user interface that operates as a feedback provider to give the user an easy-to-grasp preview on how much more modeling work is needed (see Section 5.3).

The derivation of polygonal meshes from point clouds has been intensively studied in recent years [11, 3, 10, 1, 16], Wang et al. [20] use additional image-based data in their interactive tool in order to regularize the building by proposing a scaffold-like structure. Still, especially in the domain of urban reconstruction, the resulting 3D buildings differ significantly from typical models designed with CAD tools: While human users construct building models consisting primarily of geometric primitives with exact intersections, meshed point clouds are inherently noisy and contain holes. Additionally, the absence of hierarchical relations makes operations like geometric editing or semantic classification cumbersome.

To tackle these problems, Arikan et al. [2] have proposed an interactive method that first identifies the basic planar shapes in a point cloud, on which initial coarse

polygons are created. Holes between the polygons are automatically closed by an optimization step whenever possible. In unclear cases, the user can edit, fix or add polygons using simple 2D operations on the corresponding segmented plane. Unfortunately, this approach relies on point clouds that resemble nearly the whole surface, and especially in the case of photogrammetrically generated data, the often-occurring large holes cannot be accurately reconstructed. Reisner-Kollmann et al. [15] proposes using image information for filling holes in the surface in an automatic method. In our approach, we employ the ideas of both these approaches: planar surfaces in the point cloud are identified and used as a sketching plane for 2D modeling interaction – but by using the photos as additional input in the 2D domain, the sketched polygons can additionally snap to image edges, and polygons for which no point cloud data is available can be accurately reconstructed.

Our novel work is therefore a combination and extension of the before-mentioned interactive modeling tools, striving for simplicity in terms of modeling operations (2D image-based sketching and snapping) and exploiting structural information (planar point cloud segments, image edges) from all data sources available – while helping and guiding the user through the process and leaving all decisions to his artistic freedom.

3 Photogrammetric Data



Figure 2: Photogrammetric Network (bottom), consisting of a 3D point cloud and photos (top), for which their relative positions and orientations have been computed. We refer to them as *Shots* in this work.

As this work focuses on interactive modeling using photogrammetric data, we describe the properties and distinctive characteristics of this input type: A *Photogrammetric Network* consists of a 3D point cloud and images, which are overlapping photos of an object or – like in our case – buildings. By applying *Structure from Motion (SfM)* techniques, the relative positions (i.e. the location and the ori-

entation) on which the photos were taken from, as well as a 3D point cloud, consisting of matching image features that have been reprojected to 3D space, can be computed (see Figure 2). The point cloud can be further densified using algorithms proposed by Furukawa et al. [8, 7], but since these points have not been measured, but were calculated using image features, photogrammetrically generated points may not be as dense and – more important – not as uniformly distributed as point clouds from laser scans, leading to more holes in the data. For example, it is difficult to extract robust features and therefore closely spaced 3D points from completely flat, featureless walls. We therefore strive for compensating this missing information by defining polygons in multiple photos, see Section 4.1.

The oriented photos – we refer to them as *Shots* – in the Photogrammetric Network are positioned around the point cloud. By having access to intrinsic and extrinsic camera parameters, transformations from the 2D image space to the 3D world space and vice versa can be achieved. In the case of our work, this is necessary for simple 2D editing and sketching steps and their according impact on the 3D world space, see Section 5.

Another advantage of the availability of Shots is their use in further reconstruction steps, as for interactive line snapping or texture generation. Furthermore, the acquisition process can be done with a consumer-level photo camera and freely available SfM-Tools, making it a cheap and easy solution compared to other methods.

4 Definition of Polygons using Shots

The primary interaction and sketching target in our framework is a Shot, selected from a photogrammetric network as described above. Sketching directly in a Shot photo for the purpose of creating 3D geometry has two major advantages in terms of usability:

- The user immediately grasps the scene to reconstruct, as a photo is a very close approximation of what one perceives when looking at an object.
- The interaction is performed in a 2D environment. This does not only make the modeling tools less complex to handle – humans are usually used to sketching or drawing on a flat sheet of paper since their early childhood.

While defining the approximate outline of a flat polygon in 2D space is therefore comparably easy to achieve, the derivation of the corresponding representation in 3D space requires additional information: The *3D plane* on which the 2D outline has to be projected from the photo is completely unknown at first, but can be calculated by taking additional constraints into account. We therefore propose three methods to estimate this needed information in an intuitive way with the least possible user effort, and without having to leave the 2D sketching domain.

4.1 Multi Shot Sketching

One method to obtain the 3D positions for the vertices of a sketched polygon in 2D image space is to define it not only in one, but in multiple photos. Since the orientation of the Shots is known in 3D space, each pixel on the image plane can be used to define a ray from the focal point of the camera through the pixel position in world space. If this is done for a polygon vertex in multiple images, the intersection point of the corresponding rays defines its 3D position (see Figure 1, middle). This is repeated for all vertices, and the unknown plane can then be estimated using the least-squares method.

We implemented these ray intersections using the linear triangulation method based on homogenous direct linear transformation (DLT) as described by Hartley and Zisserman [9] resulting in a least squares optimal solution. This approach is just one possible solution to this intersection problem. We opted for it as the authors state, that the “homogenous linear method [...] often provides acceptable results. Furthermore, it has the virtue that it generalizes easily to triangulation when more than two views of the point are available”. We take this into account during our guided sketching feedback, where we encourage the user to define the polygon in more than 2 shots (see Section 5.3).

4.2 Point Cloud Supported Single Shot Sketching

Even though the multi-view approach described above is an algorithmically well-working solution, a human user would prefer to minimize ones efforts and wants the system to “understand” what one intended to do after sketching a polygon in a single Shot, and reproject it into 3D space. This can in fact be made possible by exploiting the point cloud data: Similar to Arıkan et al. [2], we segment the point cloud into planar segments using the RANSAC algorithm by Schnabel et al. [17]. After an initial polygon has been sketched, we transform the points of each segment from 3D world space into 2D image space, and test which points of each segment lie inside the polygon. Note that in our current implementation, we perform this test for all segments, which could be easily optimized by performing a culling step (e.g. by using the bounding boxes of the segments).

We compute a heuristic $h \in [0, 1]$ which gives us an estimation of how well a polygon fits a planar segment. We use the number of points lying *inside* the 2D polygon as well as the *uniformity* of the distribution of these points, i.e. whether the projected point cloud segment has “holes” in it. The uniformity is estimated by rasterizing all points as splats over the polygon and then determining a fill ratio r , where 1 means fully filled and 0 not filled at all.

$$h = \frac{mr}{n}$$

where n is the total number of points of the segment and m is the number of points inside the polygon.

Splat size q is based on the average point distance, where d_i is the distance between point i and its nearest neighbor.

$$q = \frac{1}{n} \sum_{i=1}^n d_i$$

If there is at least one segment that passes the (adjustable) acceptance threshold, we choose the one with the highest result as the potential candidate, and inform the user about the outcome (see Section 5.3). If the user decides to make use of it, the polygon is projected onto the plane that has been fit to the point cloud segment (see Figure 1). Otherwise, the user continues sketching the polygon in further views, and the Multi-view Shot Sketching algorithm is applied. Nevertheless, the initially found candidate segment can still be helpful: if the normal of the polygon calculated using the multi-view method differs only 10 degrees from the segment plane normal, the polygon is adjusted to it accordingly.

4.3 Sketching Using the Plane of Existing Polygons

Since it is obviously possible for a human user to assign a semantic meaning to polygons that are being sketched, it is often an easy task to recognize that some elements lie on the same plane in 3D space. This is especially the case for elements like windows, doors or balconies on a facade. We therefore allow the user to simply define the polygon of an existing element as the 3D sketching plane for the next polygon, and can therefore reproject the 2D outline to 3D space immediately.

5 Guided Polygon Creation

After providing information on the theoretical background on the View-based Shot Sketching in the previous Section, we describe how we integrated these concepts in interactive workflows that are designed to give the user an optimal modeling experience. All interactive concepts described in the following Sections only guide and support the user – despite all suggestions of our system, the assume that the “user knows best” what his intentions are. Every suggestion and guidance step in our system can therefore also be safely ignored by the user.

5.1 Shot View Navigation

As described above, all shot view sketching operations are performed in the 2D photos of the Shots for reasons of simplicity. Although the sketching takes place in the images, it is of utter importance that the user implicitly always knows about the current view location in the 3D world, so that the spatial context can be used to sketch polygons in multiple Shots and not mix them up.

During sketching, the user is presented a 2D view of the current photo. Nevertheless, since the corresponding Shot

incorporates 3D information, we allow the opacity value to be changed arbitrarily, so that the 3D content (e.g. already modeled polygons or even the point cloud) can be made visible. Furthermore, the navigation between the shots has been designed to help to retain the information of the current user position: Instead of changing the displayed photo immediately, the camera starts a flying animation to show where the user is going. We also allow the user to leave the “Shot View” at any time and fly around in the 3D scene, and fly back to the current shot or the next shot with a smooth transition later on. For these reasons, the shots needs to be internally sorted according to their spatial neighborhood relation and not according to the time the photo was taken or even the file name. We therefore decided to sort the shots with a *Traveling Salesman* algorithm with the distance between the shot centers as weight function, resulting in an order that humans would intuitively describe as the proper natural way of describing the neighborhood relations.

5.2 Sketching and Snapping in Shot View

Once the user decides to start modeling a new polygon in a selected Shot, the initial polygon only needs to be sketched roughly on the photo, as we allow it to snap to near edges in the image. For this, we use an implementation of the Line Segment Detector described in the work of von Gioi et al. [19] to find edges in the underlying image. The outline of the initially sketched polygon is compared to the line set of the image. Two lines of these sets are matching if they are nearly parallel and spatially close. If no matching image line to a polygon edge is found, the initial edge will be used. To conclude the snapping process, the matching lines are intersected with each other, and the intersection points are the vertices of the new, snapped polygon. Figure 3 shows the polygon snapping workflow.

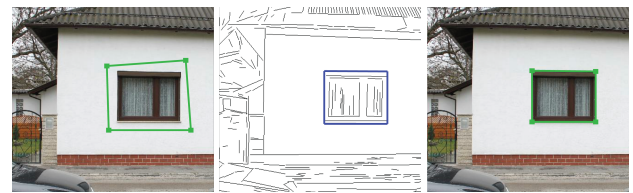


Figure 3: Polygon Snapping: sketched 2D polygon on an image (left), extracted image lines with matchings in blue (middle), snapped polygon (right)

If the user opts for using the Multi-View Shot Modeling mode, the modeling workflow requires the same polygon to be available in different images. Instead of having to sketch the corresponding polygon again, the proposed system tries to minimize the needed efforts: As soon as the user switches to the next Shot, the initial polygon is not only projected into the other image, but it is positioned to “fit the same sketched object”. For example, if the polygon snapped to window edges in the initial image, the pro-

jected polygon in the neighbor image also tries to find and snap to the same window edges.

To achieve this, we transform the edges of the initial polygon outline into the underlying photo and compute normalized color histograms of them. Then the polygon outline histograms are compared with histograms of edges found in the target image to find matching lines. A match is a pair of lines, one from the polygon outline and its corresponding edge in the target image. The best polygon-outline image-edge pair is the basis of the polygon in the other shot. Further adjacent edges are stepwise added to the polygon depending on histogram matches, or approximated if no match was found.

Especially in the case of buildings, it becomes obvious why an interactive approach with minimized input that mostly consists of deciding on proposed suggestions is important: Architectural objects often consist of extremely similar and repetitive patterns, and an initially found window can be found multiple times on the next photo, resulting in the same amount of candidates for the correspondence. Since the user is able to retain a global spatial overview more easily, the correct window can be picked with a single click.

5.3 Visual Guidance Feedback

As stated before, we want to minimize the needed user interaction while keeping the possibility to influence any design decision at the user level. It is therefore important for the user to be provided with feedback on whether the polygon should be sketched in further views, or if enough information on the needed 3D plane is available to compute the world space position of the object.

During each polygon creation process, the user gets permanent feedback via our novel Visual Guidance interface to realize this: In the user interface, a state bar appears as soon as the initial polygon is sketched. The states can switch between *red* (not enough information), *yellow* (the system can suggest a 3D polygon, but it may be inaccurate or ambiguous) and *green* (an accurate polygon can be provided, and no other plane candidates can interfere). See Figure 4 for a visualization of the guidance element in the user interface.

Concretely, we use the red state whenever an initial polygon has been sketched, and no plane to project the 2D outline onto is available. This is the case when no neighboring 3D polygon has been selected (see Section 4.3) and no fitting planar point cloud segment can be found (i.e. the metric returns no value above a certain user-definable threshold for all point cloud segments, see Section 4.2). The yellow state is used when either two or more potential planar point cloud segment candidates that are of equal quality are available, or, when using Multi-view Shot Sketching mode, the polygon has only been defined in two Shots yet (which may be inaccurate, see Section 4.1). The user can stop sketching anytime this quality estimator is not in red state, and a 3D polygon is created –

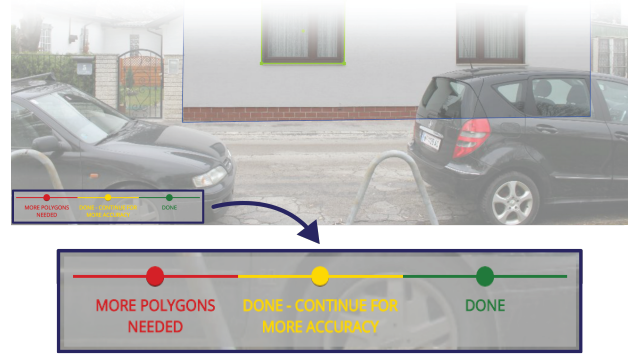


Figure 4: The Visual Guidance interface shows whether enough polygons have already been sketched to compute a 3D polygon, or if the user should continue sketching.

but has to be aware that the result may not be as accurate as needed when he stops too early.

6 Additional Photo-based Modeling

Apart from sketching the initial polygons, we have integrated further possibilities that demonstrate the combined use of photo data, point clouds and geometry in a single environment.

6.1 Model refinement

All ordinary 3D polygon-modeling tasks in our system can also be performed via the Shot view. This is especially true for existing polygons that have not been modeled in this particular view, but can be reprojected and edited in the corresponding photo anyway. Following the same principle, polygons snap to edges, and can be aligned according to the image content: Model refinement through the shot view is more intuitive and accurate for a user than, for instance, fitting a polygon to the point cloud.



Figure 5: Left: By defining hierarchical relations, holes and side faces are automatically extracted. Right: Interactive removal of occluding objects from the texture by overlaying the photo semi-transparently using the Shot view.

In addition to the Shot-based sketching operations described in the previous Sections, our framework supports regular manipulation and polygon creation known from other 3D modeling packages. This comes handy in cases when not the whole building has been photographed, but

the surface needs to be closed (even though this cannot be seen as part of the reconstruction anymore). We also included an optimization-based snapping approach to close the small gaps between the polygons as proposed by Arikan et al. [2], which moreover takes further constraints like parallelism or orthogonality of edges into account – an often-needed requirement for CAD-ready models. Moreover, we allow the definition of hierarchical relations: This makes it easily possible to define “holes” for windows and doors in the facades (the corresponding side faces are added automatically), like in Figure 5, left.

6.2 Texture Brushing

The Shots can furthermore be used to generate textures for the created polygons by reprojecting the photos onto the 3D plane. We employed the technique proposed by Musialski et al. [13], where the texture is a composition of the photos, and each pixel gets colored according to the image of the best fitting shot. While the initial source shot of each pixel is initially selected automatically based on angle and distance, arbitrary parts of the texture can be “repainted” with the content of a user-selected shot photo in order to remove occluders or artifacts. Figure 6 shows textures with different coloring according to shots.



Figure 6: Top left: The initial texture containing occluders. Bottom Left: The associated Shots, visualized using a false color mask. Top right: The cleaned texture after interactive brushing. Bottom right: The correspondingly modified mask.

We extended the original method, in which the user could brush a polygon only directly in 3D view, to be also used via the Shot view, in which the transparency can be adjusted interactively. This way, the user simultaneously has access to the current state of the textured polygon in the 3D world, and the 2D Shot photo. In the Shot view brushing mode, the brush paints over the texture with the content of the shot image the user is actually looking at, so that the user can easily find the proper image part for a specific texture position (see Figure 5, right). Switching between Shots and leaving the Shot view is possible at any time as described in Section 5.1.

7 Implementation

Our novel modeling and reconstruction framework has mainly been implemented using an internal rendering framework based on the .NET framework and OpenGL. The Visual Guidance feedback element has been realized using a web-based overlay that was created using HTML5 and the D3.js toolkit [4]. The interactive texture brushing method makes use of a Poisson solver implemented in OpenCL. For the polygon snapping, we were given access to the original implementation of Arikan et al. [2].

The tool currently supports photogrammetric data sets generated with either the PMVS/CMVS toolkit [6] or with the commercially available software Agisoft Photoscan [12]. During the import process, the Shot neighborhood relations as described in Section 5.1 are computed, and the image edges for snapping are extracted in preprocessing steps.

We believe that our proposed workflows and interaction methods can technically be integrated into existing 3D modeling packages, but it has to be carefully evaluated whether the interactions described in this paper conflict with the standards established there.

8 Results

We have evaluated our novel modeling framework by trying to reconstruct several buildings from photogrammetric data sets. All operations can be performed completely interactive once the data set is imported, and the real-time frame rates allow fluent work on a consumer-level computer.

As can be seen in Figure 7, the targeted goal of creating low-polygonal, textured, CAD-ready 3D buildings in just a few minutes could be reached: The modeling times for the buildings lie between five and fifteen minutes – including the generation of textures for the polygons. It is important to notice that especially the side parts of the buildings, where no complete point cloud was available due to the limited access for the photographer to the area, could be accurately reconstructed using our image-based approach. The front facades, where the point cloud is usually quite dense, could be successfully modeled with the single Shot method described in Section 4.2. Once a single window of a certain type was modeled, all the others on the same facade could be created using the same plane and the edge-based snapping feature within seconds.

8.1 Limitations

Even though we have shown that using both photos and point clouds from photogrammetric data sets in an interactive workflow makes it possible to reconstruct more areas accurately, our approach still suffers from the fact that objects that are hidden, occluded or only visible in a single



Figure 7: Three textured 3D building models generated with our approach. In the left column, the photogrammetric point cloud is visualized, followed by the geometric reconstruction including hierarchical definitions in the middle column. In the right column, the final model with textures generated from multiple photos and using interactive occluder removal are shown. While parts that are not depicted in the point cloud could be reconstructed using the photos, that the backsides of the houses were modeled freely, as they were not accessible for the photographer. Modeling time including texture generation: Top row 5 minutes, seconds row 15 minutes, third row 10 minutes, fourth row 20 min.

photo require manual, inaccurate modeling steps. Furthermore, we are (similar to the methods proposed by Arkan et al. [2] and Sinha et al. [18]) limited to the reconstruction of planar surfaces. Even though curved surfaces can be approximated using multiple polygons, the handling of such primitives is more challenging than it is for planar shapes.

9 Conclusion & Future Work

We have demonstrated how to combine interactive techniques from both image-based and point cloud-based methods to reconstruct CAD-ready 3D models of buildings within a few minutes. 3D planes, on which sketched 2D polygons are reprojected, can not only be computed from multiple views, but also from planar segments detected in the corresponding point cloud. Image-based snapping features and suggestions further improve the sketching workflow.

Our novel method is a natural extension of these related techniques, and does not interfere with their concepts, but improves them. By introducing an intuitive *Visual Guidance Indicator*, users can take shortcuts during the image-based modeling steps, while being aware of the quality impact this has.

As this project is ongoing work, we have to especially evaluate and fine-tune user-oriented interaction methods in the future. Not only will a decent user study be performed and more data sets be used, but we will also investigate if we can use learning algorithms to replace currently user-defined parameters and thresholds, as they may vary depending on input data. As we already managed to simplify and minimize the interactions to a level that allows reconstruction of a building with just a few mouse clicks, we will evaluate if these concepts can be used on a touch-based interface as well, opening the door for the use on mobile devices.

Acknowledgements

We wish to express our thanks to Thomas Ortner and Stefan Maierhofer from the VRVis Research Center for their valuable feedback. This work was supported by the Austrian Research Promotion Agency (FFG) through the FIT-IT project Replicate, project no. 835948. The competence center VRVis is funded by BMVIT, BMWFJ, and City of Vienna (ZIT) within the scope of COMET Competence Centers for Excellent Technologies. The program COMET is managed by FFG.

References

- [1] Pierre Alliez, David Cohen-Steiner, Yiyi Tong, and Mathieu Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 39–48, Aire-la-Ville, Switzerland, 2007. Eurographics Association.
- [2] Murat Arikan, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. O-snap: Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics*, 32:6:1–6:15, January 2013.
- [3] Jean-Daniel Boissonnat and Steve Oudot. Provably good sampling and meshing of surfaces. *Graph. Models*, 67:405–451, September 2005.
- [4] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D³ data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.
- [5] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *SIGGRAPH*, pages 11–20, 1996.
- [6] Yasutaka Furukawa. Clustering views for multi-view stereo (CMVS). <http://www.di.ens.fr/cmvs/>. Accessed: 2016-02-29.
- [7] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, Richard Szeliski, and Google Inc. R.: Towards internet-scale multiview stereo. In *Proceedings of IEEE CVPR*, 2010.
- [8] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1362–1376, August 2010.
- [9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, chapter 12.2, page 312f. Cambridge University Press, second edition, 2004.
- [10] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the 4th Eurographics symposium on geometry processing, SGP '06*, pages 61–70, Aire-la-Ville, Switzerland, 2006. Eurographics Association.
- [11] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanerke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01*, pages 57–66, New York, NY, USA, 2001. ACM.
- [12] Agisoft LLC. Photoscan. <http://www.agisoft.com/>. Accessed: 2016-02-29.
- [13] Przemyslaw Musialski, Christian Luksch, Michael Schwärzler, Matthias Buchetics, Stefan Maierhofer, and Werner Purgathofer. Interactive multi-view façade image editing. In *VMV 2010*, pages 131–138, November 2010.
- [14] Przemyslaw Musialski, Peter Wonka, Daniel G. Aliaga, Michael Wimmer, Luc van Gool, and Werner Purgathofer. A survey of urban reconstruction. *Computer Graphics Forum*, 32(6):146–177, September 2013.
- [15] Irene Reisner-Kollmann, Christian Luksch, and Michael Schwärzler. Reconstructing buildings as textured low poly meshes from point clouds and images. In Nick Avis and Sylvain Lefebvre, editors, *Eurographics 2011 - Short Papers*, pages 17–20, April 2011.
- [16] Nader Salman, Mariette Yvinec, and Quentin Merigot. Feature preserving mesh generation from 3D point clouds. *Computer Graphics Forum*, 29(5):1623–1632, 2010.
- [17] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.
- [18] Sudipta N. Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. Interactive 3d architectural modeling from unordered photo collections. *ACM Trans. Graph.*, 27(5):159, 2008.
- [19] Rafael Grompone von Gioi, Jrmie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 32(4):722–732, 2010.
- [20] Jinglu Wang, Tian Fang, Qingkun Su, Siyu Zhu, Jingbo Liu, Shengnan Cai, Chiew-Lan Tai, and Long Quan. Image-based building regularization using structural linear features. *Transactions on Visualization and Computer Graphics*, 1(99):1, 2015.

Perception

Preferred Speed of Visual Adaptation to Darkness in Computer Games

Marek Wernikowski*

Supervised by: Radosław Mantiuk[†]

West Pomeranian University of Technology
Szczecin / Poland

Abstract

The human visual system has the ability to adapt to various lighting conditions. Models of this visual adaptation process are applied to increase the attractiveness of graphics while playing computer games. The main goal of this work is to implement the light adaptation process in a test game framework and evaluate the perceptual impact of adaptation to darkness on the gameplay. We take care of the reliability and physical correctness of the simulation but also artificially modify the adaptation speed to test the player's preferences. The results reveal that faster visual adaptation to darkness is more preferable than an approach which follows the natural behaviour of the human visual system.

Keywords: visual adaptation, adaptation to darkness, visual adaptation in computer games, perception, tone mapping, real time rendering

1 Introduction

Lighting conditions vary significantly depending on the environment in which we are located so a mechanism, which allows the humans to see objects in both bright and dark conditions is indispensable to survival. This process within the human visual system (HVS) is called *visual adaptation* - it allows HVS to adjust to various light conditions ranging from very dark scenes lit by the stars to bright environments illuminated by millions of candelas.

The main focus of this work is *adaptation to darkness*. This is the process which takes place when we switch from well lit environment to a darker one. The adaptation to darkness proceeds with a constant and rather slow speed. It takes tens of seconds to fully adapt from a bright environment to a very dark one. In contrary, adaptation in the opposite direction from darkness to brightness is very fast and strongly non-linear. At first people are blinded by the light but after a short time they begin to see the objects. During this time, human is adapting to the *adaptation luminance* - the average luminance HVS adapts to

considering an arbitrary gaze direction. By measuring the brightness in two separate situations, it is possible to calculate the required time for full adaptation. Therefore, the bigger the difference between current and previous luminance is, the shorter it takes to adapt to the new setting.

The adaptation is *gaze-dependent* which means it takes into consideration the gaze point of the observer. Humans frequently change their gaze direction and try to adapt to different regions. As a result, HVS is in the maladaptation state, in which the adaptation luminance is changing towards a target value but never reaches this value because in the meantime the target is changed.

Models of visual adaptation are used in computer games to make the graphics more realistic and plausible. A noticeable example is the "Uncharted 2" game in which the tone mapping with visual adaptation is implemented. In this game temporal adaptation is applied by assigning fixed spots on the floor, in which the eye should adapt to light or dark. Depending on a place the player has stepped into, view was properly configured. The advantage of this solution is the simplicity of calculations. However, any dynamic light source cannot be used, which can be essential for the realistic simulation [5]. The other technique worth mentioning is the one used in Unreal Engine. Here adaptation is based on the average luminance of the scene. It is also possible to adjust the time needed to adapt to light and dark separately. It is an accurate way of simulation, although it does not take into consideration the actual place where the observer is looking - the gaze point. Example screenshots from Unreal Engine are presented in Fig. 1.

A main goal of this work is to model the visual adaptation process in a correct way in terms of the human perception, so that it could reflect the actual behaviour of HVS. However, the adaptation to darkness lasts even tens of minutes and, from a computer game perspective, it often does not make sense to model this process in the same way as it happens in nature. This adaptation would be too slow for fast modern games.

In this paper, we evaluate whether using the perceptually correct visual adaptation operators is practically justified. We perform an experiment, in which people choose a preferred speed of the adaptation to darkness. To mimic the behavior of HVS, we model the adaptation to brightness based on the perceptual formulas, while the adapta-

*mwernikowski@wi.zut.edu.pl

[†]rmantiuk@wi.zut.edu.pl

tion to darkness is simulated by the linear luminance transformation. To find the target speed of the adaptation to the dark environment, we vary the adaptation time and ask people to choose the most plausible approach.

Section 2 gives background information how the adaptation process works for different lighting conditions. Section 3 is focused on our game framework and shows how we simulate the visual adaptation mechanisms. Section 4 presents the results of the perceptual experiment evaluating the adaptation to darkness. The paper is concluded in the last Section.

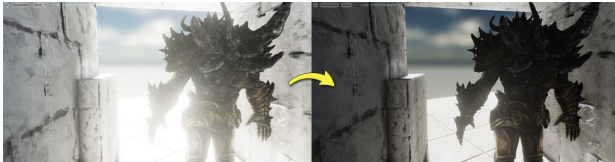


Figure 1: A screenshot generated by the Unreal Engine during (left) and after (right) adaptation to the bright environment [2].

2 Background

In this Section we give basic information on the human visual system and visual adaptation process.

2.1 Rods and cones

Human vision is based on two types of photoreceptors *rods* and *cones* that absorb the light from the environment [18]. The cones allow for colour vision under appropriate lighting conditions, while the rods are responsible for recognizing shapes and monochrome vision in low light conditions.

The colour vision is activated when light interacts with the chromophore inside the visual pigment (opsin) which can be found on top of the cone. There are three kinds of opsin (L, M and S) related to three colours they produce (red, green and blue). The chromophore changes shape and triggers a cone. If at least two kinds of cones are triggered, a signal is sent to the photoreceptor, in which light is converted into an electrical signal, and later transferred to the brain. There are between 6 and 7 million cones with the majority of them being L (64%) or M (32%) types [11]. The "blue" cones (S) represent just 2% of all cones. In comparison to L and M, the S cones are much more sensitive to light and reside outside the fovea.

The rods are activated in low light conditions and are responsible for the scotopic vision, in which people do not see colour but are very sensitive to contrast changes (a thousand times more sensitive than cones) [18]. There are about 120 million of rods and they can be triggered with just individual photons. They adapt to shorter wavelengths than cones, therefore, people see blue objects quite clearly in the dark, while red objects might even be completely invisible. Since all rods are outside the fovea and

the highest acuity area, it is sometimes impossible to see an object at which we are directly looking in the dark, e.g. when we want to observe stars in the night. The only way to improve this vision is to look at a star "out of the corner of the eye". Then, it is observed with the part of retina containing mostly rods. The scotopic vision is possible because of the *rhodopsin* pigment on top of the rod. This process is very similar to the one in the cones: when light (at least one photon) strikes the rod, the chromophore changes shape and triggers an electrical signal sent to the brain.

2.2 Visual adaptation

The human eye is able to adapt to luminances which differ greatly, even 14 orders of magnitude - from moonlight ($10^{-6} \frac{cd}{m^2}$) up to sunlight ($10^8 \frac{cd}{m^2}$) [15]. The *visual adaptation* process takes place when the lighting condition, to which the observer is currently adapted, changes. For example, this happens as a result of someone entering a darker room, turning on the light, or walking outside. The time for the eye to adapt to the new environment depends on whether the cones or the rods are being activated/deactivated.

In the case of increasing the ambient luminance, the photopigment in rods gets bleached [1]. For a few seconds, they are completely blind and the sensitivity of cones begins to increase. The whole adaptation takes up to 5 minutes but the vision might be fully clear in less than one second [8]. During this short period the vision is heavily impaired - the colours are barely visible and all objects seem to be too bright.

During adaptation from bright to dark, a reverse process takes place - at the beginning it is hard to see anything [3]. It is caused by the fact that cones are currently in the low sensitivity state and rods are bleached. Then, cones regain their sensitivity and rods are regenerated. When cones achieve highest sensitivity, rods begin to increase their sensitivity until they are fully adapted.

The adaptation to darkness is a sustained process - depending on the amount of light it could take from 10 minutes to 2 hours, sometimes even more [8]. This process is presented in Fig. 2. From obvious reasons in simulations and computer games this time has to be shortened, so that the observer would not need to wait minutes to see any information.

2.3 Maladaptation

The human eyes mainly adapt to an area covering approximately 2-4 degrees of the viewing angle around the gaze direction [17]. Other areas of the scene, observed not in foveal but in para-foveal and peripheral regions, have significantly less impact on the adaptation level, although, a human frequently changes gaze direction (even a hundred times per second) and tries to adapt to different regions [10]. As the process of the luminance adaptation is

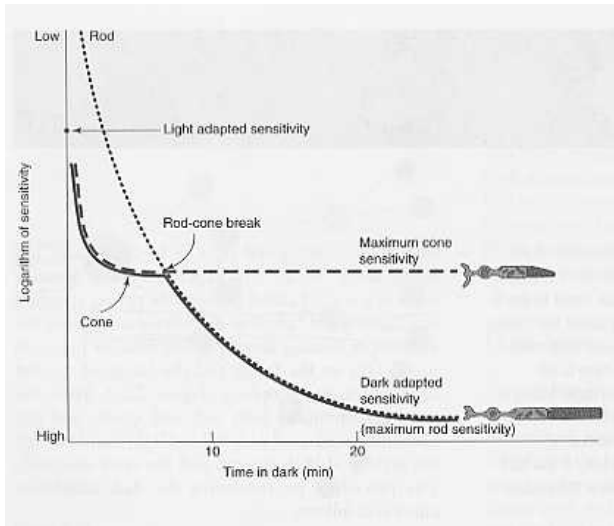


Figure 2: The dark adaptation process (after Gordon et al. [3]).

slower than changes of gaze direction, the HVS is permanently in the *maladaptation* state, in which the adaptation luminance is changing towards a target value but never reaches this value because in the meantime the target is changed.

2.4 Previous work

Visual adaptation has been a topic of several articles. In 1996, the model of visual adaptation was described in [4]. The model discussed in this paper is used for realistic image synthesis which takes into consideration threshold visibility, colour appearance, visual acuity, and sensitivity. It is however only usable in static pictures. Another article, published in 2000, uses much simpler equations [13]. The RGB is created using the adaptation model with very efficient technics. This makes it possible to use in dynamic scenes. The main disadvantage of this method is the fact, that it does not consider the gaze point of the observer. Another article, published in 2004, operates on High Dynamic Range images [8]. It models rods and cones separately, and the local adaptation is computed using Gaussian function. This method is very precise and performs very well in simulations and static images, however in our work we want to produce effects of similar quality with lower performance impact.

3 Test game framework

In this Section we present our prototype game framework. The main goal of this approach is to implement the luminance adaptation models and provide a testbed for the perceptual experiments.

3.1 Implementation

The framework has been implemented in C++ based on the OpenGL library (version 4.0) supported by GLFW for input/output operations, Assimp for loading 3D models, and FreeImage for loading textures. The lighting computations are based on the Phong shading model. We built a scene consisting of 15 objects and approximately 18.000 triangles, which presents the interior of an office. The scene contains very bright object (lamp on the ceiling) and a number of dark objects with a luminance that is lower by almost four orders of magnitude. An example rendering is presented in Fig. 3.



Figure 3: An example screenshot from our test game framework.

3.2 Visual adaptation module

A core module in our framework is the visual adaptation mechanism presented in Fig. 4. We do not use an eye tracker, so the gaze location is assumed to be at the centre of the screen. We compute the weighted average of the pixel luminance from the whole image, wherein the weights are delivered as a texture mask (see Sect. 3.3). The whole process is repeated for each frame taking into consideration the maladaptation mechanism (see Sect. 3.4). The obtained temporary adaptation luminance is used to tone map the image based on the sigmoidal tone curve (see Sect. 3.5). The actual visual adaptation is implemented by varying the global luminance level of the rendered image.

3.3 Spatial extent of visual adaptation

In HVS, the highest impact on the vision has the high-acuity area, the 8-degrees surrounding of the gaze point. Recently Vangorp et al. [17] proposed an adaptation model, in which the local adaptation luminance is based on the pixel values in this area. However, in our framework we apply a simpler approach based on the gaze-dependent contrast sensitivity function [14]. This function roughly follows distribution of the cones in the retina. The *spatial cutoff frequency* is the way of measuring the smallest visible object. It is measured in cycles per millimeter. As the number of cones decreases with the eccentricity, we assume that adaptation luminance is affected mostly at the areas of the highest frequency [9].

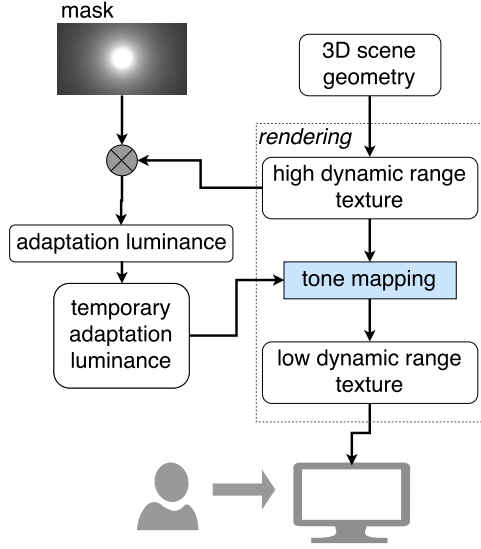


Figure 4: The diagram of the visual adaptation module.

The below equation models the spatial cutoff frequency f_c of the human retina, i.e. the highest frequencies that are still visible for the eccentricity d :

$$f_c = 43.1 * \frac{E_2}{E_2 + d}, \quad (1)$$

where E_2 denotes the eccentricity at which spatial frequency drops to half (we use a value of 43.1 cpd). The graph of this function is presented in Fig. 5. The mask based on this equation is shown in Fig. 6.

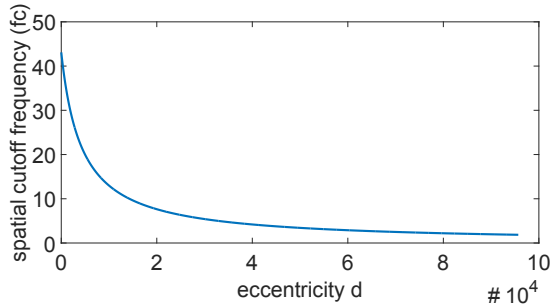


Figure 5: Graph of the f_c function.

3.4 Temporal adaptation

The adaptation luminance changes over time because observers move their gaze location. As the adaptation time can last longer than the animation frame, the maladaptation state should be considered. For this task we use a shader, which receives the luminance value from the previous and current frames. These values are used to modify the adaptation over time [7]:

$$\hat{L}_a^{new} = \hat{L}_a + (\hat{L}_{HDR} - \hat{L}_a)(1 - e^{-\frac{T}{\tau}}), \quad (2)$$



Figure 6: The mask used to approximate the adaptation luminance.

where $\hat{X} = \log_{10}(X)$, L_a^{new} denotes a new adaptation luminance, L_a is the adaptation luminance from the previous frame, L_{HDR} - luminance of the input HDR image, T is the time which elapsed between the display of the current and previous frame, and τ is the adaptation speed. The above exponential function gives the accurate results, however we simplified this approach to a formula:

$$L_a^{new} = \begin{cases} \min(L_a + \frac{T}{\tau_{rod}}, L_{HDR}), & \text{if } L_a < L_{HDR} \\ \max(L_a - \frac{T}{\tau_{cone}}, L_{HDR}), & \text{if } L_a \geq L_{HDR} \end{cases} \quad (3)$$

where τ_{rod} and τ_{cone} indicate rod and cone adaptation time, respectively ($\tau_{rod} = 9.0s$, $\tau_{cone} = 0.1s$). The above equation gives a rough approximation of the exponential formula. Apart from that, it is much simpler and more appropriate for real-time rendering.

As shown in Fig. 7, switching off the lamp starts the slow adaptation to dark (top row). When the lamp is switched on, this process is interrupted and the adaptation to bright begins before the eyes become fully adapted to dark (middle row). After switching off the lamp again, the observer can fully adapt to the darkness (bottom row).

3.5 Tone compression

In our test framework we implemented the sigmoidal tone compression proposed for the gaze-dependent tone mapping in Mantiuk and Markowski [10]. The shader used for this compression converts colour to luminance using the formula: $L = c_r * 0.212656 + c_g * 0.715158 + c_b * 0.072186$, where c_r , c_g , and c_b are the red, green, and blue components, respectively.

The tone compression is based on the Naka-Rushton equation [12]:

$$L_{LDR} = \frac{L_{HDR}}{L_{HDR} + s}, \quad (4)$$

where s denotes the momentary adaptation luminance L_a^{new} .

The output colour values of the rendered image are computed based on the formula [16]:

$$c_{new} = \left(\frac{c_{HDR}}{L_{HDR}}\right)^s * L_{LDR}. \quad (5)$$



Figure 7: Top row: observer is fully adapted to the bright environment, then the lamp is switched off and slow adaptation to dark begins. Middle row: the lamp is switched on, the eye is blinded by the bright environment but it quickly adapts to the bright environment. Bottom row: the lamp is switched off again, after some time observer adapts to the darkness.

We chose the colour desaturation factor equal to 1.2. Finally, the output image is gamma corrected and presented on the display calibrated to the sRGB colour profile.

Example images rendered for different values of the adaptation luminance are presented in Fig. 8.



Figure 8: Images rendered for observer adapted to the bright areas (top) and dark area (bottom). The orange cross depicts the gaze location.

4 Experimental Evaluation

We performed a pilot experiment, in which game players were asked to assess the most suitable speed of the visual adaptation to darkness.

4.1 Stimuli and procedure

During the experiment observers sat in front of the display in a 60 cm distance. They were asked to observe an animation sequence. At the beginning, the grey background with the experiment instruction was displayed. Then, a scene presenting the interior of the room was rendered (see example in Fig. 3). The camera was looking at the bright lamp for 5 seconds, which caused the adaptation to high brightness. Next, the camera smoothly moved from the lamp to the desk, whose luminance was one order of magnitude lower than that of the lamp. The camera remained in this position until the adaptation to dark was finished. The whole procedure was repeated again but for different times of adaptation to dark. Then, the observer had to decide, which adaptation speed was more plausible - the exact question he had been given was 'Which adaptation speed do you like more?'. The answer was provided by moving the slider on the bar which was scaled from -1 ('definitely first') through 0 ('I do not see a difference') to 1 ('definitely second'). The bar was continuous, so that the observer could answer as precisely as possible.

We tested every combination of 2, 8, 16, 25, and 33

seconds, while the pairs were chosen randomly. Each observer repeated the experimental session three times for each pair of the adaptation times.

The experiment was performed in a darkened room. Images were displayed on a 27 LCD display with a native resolution of 2560 x 1440 pixels and a screen width and height of 62.4 cm and 40.1 cm, respectively. The computer was equipped with a Geforce 780 Ti GPU.

Participants

The experiment was performed on a group of 13 volunteer observers (age between 17 and 50 years). They declared normal or corrected to normal vision and correct colour vision. The participants were aware that the visual adaptation is tested, but they were naïve about the purpose of the experiment.

4.2 Results

Fig. 9 presents a bar plot with the results of the experiment, which shows the preference as a function of the speed of adaptation to the dark environment. This preference is a number of votes cast for the adaptation time normalised by a number of times this time has been tested.

The best score was obtained for 2 seconds, however for longer adaptation of 8 seconds the observers' preference is very close to this result (0.7647 and 0.7639 respectively). The obtained results suggest that players prefer shorter adaptation times in comparison with the natural HVS behaviour.

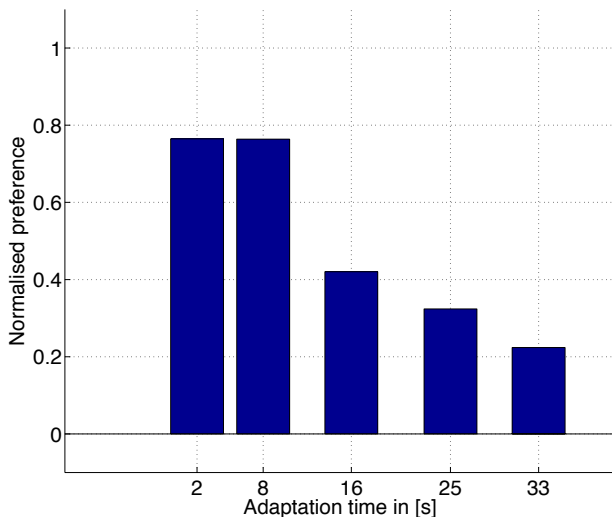


Figure 9: Results of the perceptual experiment studying speed of the adaptation to darkness.

5 Conclusions and future work

In this work, we conducted an evaluation of the preferred speed of the visual adaptation to darkness. We imple-

mented a test game framework supporting the adaptation to both dark and bright environments. In this framework the output rendering is tone mapped using the global compression curve, whose shape is modified based on the adaptation luminance value. This value is computed using the weighted average luminance of the scene. The adaptation luminance is changed over time to simulate the mal-adaptation conditions.

We performed an objective experiment, in which a number of adaptation times to darkness were compared. The results revealed that the most favourable is adaptation lasting 2 and 8 seconds, which is in contrast to the physical model of the human visual adaptation suggesting much longer timings.

Our study approaches the problem of the adaptation speed from a perspective of computer games. Even though we proved that the users prefer quicker adaptation, it might not be true for every game time. Our solution is well-suited for the games based mainly on exploration and the analysis of the environment. Games which require higher level of realism, like simulators, would look better with longer adaptation. Long adaptation could be also used as an obstacle of some sort, e.g. in the First Person Shooter games - the adaptation time could make it harder to spot enemies and it would be necessary to adapt new tactics.

In future work we plan to conduct perceptual experiments that assess the game player's preferences towards a model of the adaptation to brightness. We would like to check whether a complex non-linear mechanism is really needed during the gameplay. Also, we plan to implement gaze-dependent adaptation controlled by an eye tracker and evaluate if this local adaptation is more preferable than the simplified approach based on adaptation to the screen centre. Another implementation worth investigating are the human saliency models [6] that do not require an eye tracker and could point out direction of the observer attention.

References

- [1] Hugh Davson. *Physiology of the Eye*. Elsevier, 2012.
- [2] Epic Games, <https://docs.unrealengine.com/1>. *Unreal Engine documentation*, 2004-2016.
- [3] Gordon L Fain, Hugh R Matthews, M Carter Cornwall, and Yiannis Koutalos. Adaptation in vertebrate photoreceptors. *Physiological reviews*, 81(1):117–151, 2001.
- [4] James A Ferwerda, Sumanta N Pattanaik, Peter Shirley, and Donald P Greenberg. A model of visual adaptation for realistic image synthesis. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 249–258. ACM, 1996.

- [5] John Hable. Filmic tonemapping. In *Uncharted 2 - HDR Lighting*, http://filmicgames.com/Downloads/GDC_2010, 2010.
- [6] Jonathan Harel. Saliency map algorithm. <http://www.vision.caltech.edu/~harel/share/gbvs.php>. 2012.
- [7] G. Krawczyk, K. Myszkowski, and H.P. Seidel. Perceptual effects in real-time tone mapping. In *Proc. of the 21st Spring Conference on Computer Graphics, Budmerice, Slovakia*, pages 195–202, 2005.
- [8] Patrick Ledda, Luis Paulo Santos, and Alan Chalmers. A local model of eye adaptation for high dynamic range images. In *Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 151–160. ACM, 2004.
- [9] Radosław Mantiuk and Sebastian Janus. Gaze-dependent ambient occlusion. *Lecture Notes in Computer Science (Proc. of ISVC 2012)*, 7431:523–532, 2012.
- [10] Radosław Mantiuk and Mateusz Markowski. Gaze-dependent tone mapping. *Image Analysis and Recognition*, pages 426–433, 2013.
- [11] E. Montag and R.M. Boynton. *Vision Research*, volume 27. University of California, 1897.
- [12] K.-I. Naka and W. A. H. Rushton. S-potentials from luminosity units in the retina of fish (cyprinidae). *J. Physiol.*, 185:587–599, 1966.
- [13] Sumanta N Pattanaik, Jack Tumblin, Hector Yee, and Donald P Greenberg. Time-dependent visual adaptation for fast realistic image display. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 47–54. ACM Press/Addison-Wesley Publishing Co., 2000.
- [14] Eli Peli, Jian Yang, and Robert B Goldstein. Image invariance with changes in size: The role of peripheral contrast thresholds. *JOSA A*, 8(11):1762–1774, 1991.
- [15] E. Reinhard, W. Heidrich, P. Debevec, S. Pattanaik, G. Ward, and K. Myszkowski. *High Dynamic Range Imaging, Second Edition: Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann (2nd edition), 2010.
- [16] Christophe Schlick. Quantization techniques for visualization of high dynamic range pictures. In *Photorealistic Rendering Techniques*, pages 7–20. Springer, 1995.
- [17] Peter Vangorp, Karol Myszkowski, Erich W. Graf, and Rafał K. Mantiuk. A model of local adaptation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia)*, 34(6):166:1–13, 2015.
- [18] Samuel J Williamson and Herman Z Cummins. Light and color in nature and art. *Light and Color in Nature and Art*, by Samuel J. Williamson, Herman Z. Cummins, pp. 512. ISBN 0-471-08374-7. Wiley-VCH, February 1983., 1, 1983.

Using Perception-Based Filtering to Hide Shadow Artifacts

Felix Kreuzer*

Supervised by: Michael Hecher†

Institute of Computer Graphics and Algorithms
TU Wien / Austria

Abstract

Combining filtering techniques with shadow mapping is a common tool to simulate soft shadows in real-time applications. A positive side-effect of such approaches is that the filtering also blurs aliasing artifacts caused by low resolution shadow maps, thereby improving the visual quality of the shadow. In this work we investigate the correlation between filter radius and shadow map resolution to optimize computational performance while mostly preserving the visual quality of the soft shadow. We present the results of a user study and offer a ready-to-use function to compute for shadow map aliasing artifacts a respective filter size that makes it unrecognizable.

Keywords: Soft Shadows, Shadow Mapping

1 Introduction

Shadows are crucial for identifying spatial relations between objects. In real-time graphics shadows are frequently implemented by using *shadow maps* [9]. A shadow map stores distances of the visible points in the scene from the point of view of the light source. When the scene is rendered from the camera viewpoint, those values can be compared to the respective distances of the points visible to the camera. If they are farther away from the light source the point is shaded. This basic approach produces hard shadow silhouettes and would theoretically need a sampling density that matches the size of the texel drawn on the screen. Modern implementations commonly average multiple shadow map samples per texel in order to produce softly blurred shadow transitions called penumbræ. Such soft shadows provide a higher degree of visual quality and increased artistic freedom.

Real-time rendering applications require shadow maps to be generated and filtered for every frame and consume lot of performance on the Graphics Processing Unit. Keeping the shadow map's resolution to a minimum reduces memory transfers and generation costs, and increases cache hits.

Our aim is to find a perceptually sound method to determine a minimal shadow map resolution. We exploit the

low-pass filtering property of soft shadow penumbræ and introduce a linear function which allows shadow map resolutions and aliasing artifacts to be reduced to a minimum.

We can summarize our contributions to real-time soft shadowing as follows:

- We investigate the relatively complex problem of artifacts generated by arbitrary aligned shadow maps in soft shadow algorithms and break down the huge parameter space, which can be hardly investigated in a user study, into a simplified version.
- A novel approach to dynamically adjusting shadow map sizes for real-time soft shadowing algorithms. By reducing the number of depth samples in a shadow map we can increase performance in shadow map generation since there are less fragments to process and fewer texture lookups. This also improves cache-efficiency because shadow samples are tightly packed and redundant samples are being avoided.
- Our method is flexible and can be applied to several existing soft shadow mapping algorithms.

2 Related Work

Shadow mapping was first introduced by Williams [9] in 1987 and has evolved ever since.

Nowadays a variety of filter based extensions to the traditional shadow mapping algorithm exist such as the following:

- Percentage Closer Filtering (*PCF*) [6] addresses the problem of anti-aliasing in shadow maps. Traditional shadow maps contain depth information, hence pre-filtering cannot be achieved directly. The solution is a screen space averaging approach. By increasing the filter size it can be used to simulate soft shadows with a constant penumbra.
- Variance Shadow Maps [3] approximate the depth values by storing mean and variance of the depth distribution. Instead of averaging multiple samples like in PCF, the probability of a fragment being lit is calculated through the moments using Chebyshev's inequality. Storing mean and variance of the depth

*falichs@gmail.com

†hecher@cg.tuwien.ac.at

distribution instead of actual depth values allows pre-filtering of the shadow map.

- Convolution Shadow Maps [1] use Fourier expansion to store and reconstruct depth values. This approach allows shadow maps to be pre-filtered but requires a lot of memory and expensive memory transfers to retrieve the Fourier coefficients.
- Exponential Shadow Maps [2] adopt an exponential function to approximate the shadow test. The main benefits are pre-filter-ability and cheap memory and computational costs.
- Percentage Closer Soft Shadowing (PCSS) [4] extends the capabilities of PCF by evaluating the filter radius for each fragment based on the distance from shadow occluder to receiver. This approach features a more plausible penumbra behavior in regions where occluder and receiver merge (*contact hardening*).

Hecher et al. [5] present a comparison of some of these algorithms using a comprehensive perceptual study.

In this article we will focus on PCF and PCSS as representative examples of filter based methods, but our findings can be applied to any of the above.

3 Investigating Shadow Map Sampling and Filtering

Our first goal is to find a relation between depth sampling resolution and shadow map filtering. We will further investigate this relation in Section 4 and propose a formula for practical use in Section 6.

Shadow maps are generated per frame by sampling depth values of a scene from the light sources perspective. The sampling process makes use of the hardware graphics pipeline by transforming scene geometry into the perspective view space of the light source and storing the nearest depth values per viewport fragment.

Later, when the scene's per-pixel lighting is calculated, the fragment shader projects each fragment to light-space and queries the shadow map to compare depth values. In case of soft shadows multiple queries are performed in the neighboring vicinity of the fragment in question and filtered by averaging in order to achieve a penumbra effect on shadow borders.

This blurring filter hides high frequency detail on the shadow boundary, leading us to the hypotheses that with increasing softness of the shadow (i.e., a bigger filter size) a less detailed shadow map resolution is required to produce visually sound results. Figure 1 demonstrates this observation by comparing the visual impact of varying filter sizes with different resolutions of the same shadow silhouette. We can see that the resolution required for displaying a perceptually sound soft shadow seems to be directly affected by the filter size.

We will take a closer look on this relation between resolution and filter size in the following sections. Our final goal will be to exploit filtering in order to minimize the shadow map's resolution. This would not only potentially reduce a shadow map's memory footprint, but also improve memory transfer performance during Percentage Closer Filtering, since the probability of hitting the right samples in a cache-segment will be higher.

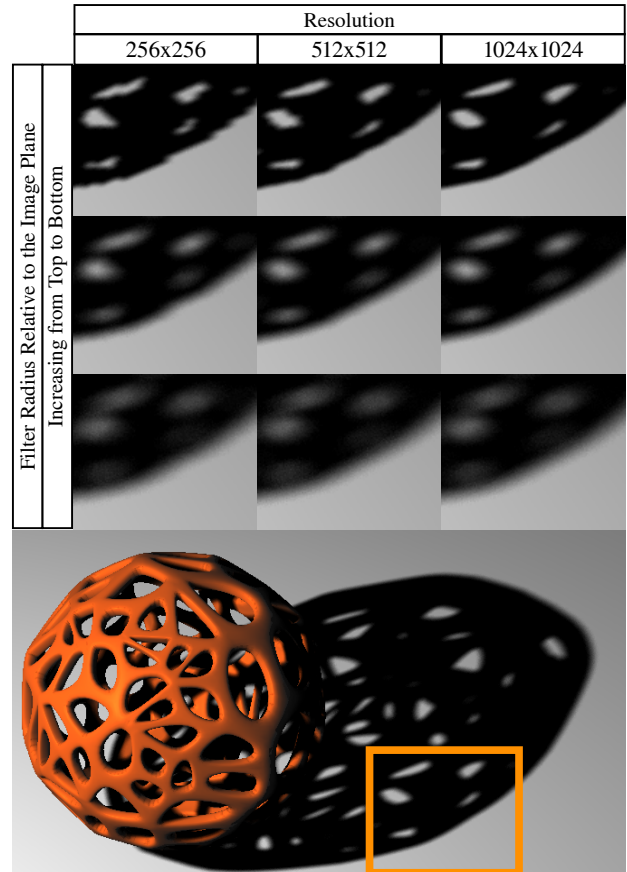


Figure 1: By doubling the penumbra radius we can reduce the shadow map's resolution by half without noticeable negative impact on the visual quality.

4 Study Design

Our goal is to hide shadow artifacts by finding an optimal parameterization for shadow mapping based algorithms. This means we need to consider several potentially important parameters related to viewer, light source, shadow map, shadow casting objects and shadow receiving objects. Additionally, because the major target of this research are real-time applications, we also need to take performance into account. In this section we discuss how we can map this relatively complex task to a simple 2D setup which reduces the parameter space greatly and allows us to find a solution to this problem.

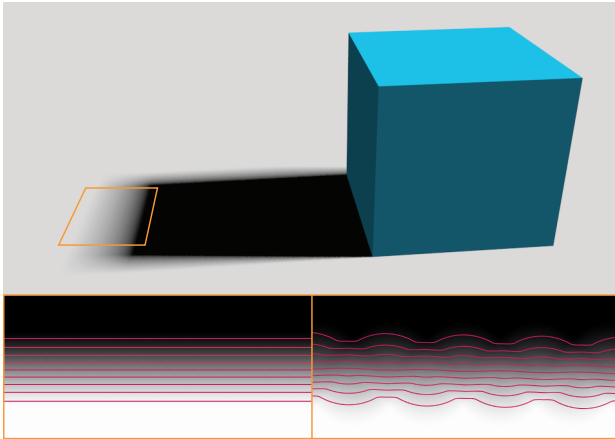


Figure 2: A comparison of soft shadow iso-contours produced by an artifact-free shadow map (left) and a shadow map with a regular artifact pattern (right).

4.1 Reducing the Parameter Space

Looking at all the factors involved in computing soft shadows, we find that naively sampling this huge parameter space in a user study would be next to impossible. Too many configurations are necessary to allow for a meaningful evaluation of all the setups that produce perceptually artifact-free images. We therefore need to reduce the complexity of the problem.

Let's continue our investigation from the previous section and take a closer look at the factors influencing the visibility of artifacts. Our first observation from Section 3 is that by increasing penumbra size the visibility of artifacts can be reduced and increasing the resolution of the shadow map has the same effect. Changing any of the other parameters mentioned above might impact the perceivability of shadow artifacts, as they can influence the projected shadow map pixels in the scene (by changing light source, shadow caster and/or shadow receiver position), the projection of the artifact onto the viewer's image plane (by changing the viewer's position) or the contrast of the produced artifacts (by changing the surface color and/or intensity of the light source). So the problem can be separated into three parts. The first part involves the artifact projected from the light source into the scene, the second part how the artifact is projected onto the image plane of the viewer and the third part the contrast and color of the artifact.

Looking at the first part from an analytical standpoint, we make the following observation: The result of computing the soft shadow from a shadow map using filter-based approaches, can be basically seen as a set of iso-contours representing the filtered hard shadow (see Figure 2). These iso-contours form patterns depending on the angle of the light source and the structure of the shadow receiving surface. Artifact-prone and artifact-free solutions will have different contour patterns and we argue that the user evaluates the dissimilarity in their curvature and slope to iden-

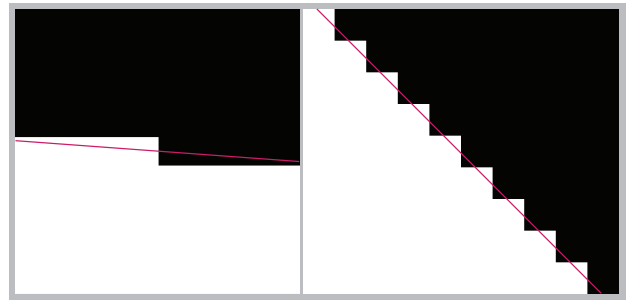


Figure 3: An illustration of the two silhouette artifact patterns used in our user study. The left image shows a single step, the right image a regular stair pattern. The red lines represent the actual silhouette of the sampled geometry.

tify artifacts. Because changing the scene setup (viewer, lights source, objects) can only effect these two factors (curvature and slope), we can simplify the parameter space to said variables. Hence, the problem becomes a simple 2D evaluation of filtering differently sized artifacts with increasing filter size. So we do not have to consider the scene setup at all. The question then is when do these differences become indistinguishable to users?

4.2 Selecting the Stimuli

Now that we were able to greatly simplify the problem, we have to sample the remaining parameters in a meaningful way.

Filter Size We treat the filter size as the dependent variable in our experiments. Our goal is to understand how much an artifact has to be blurred so users cannot recognize it anymore.

Artifact Size Selecting meaningful artifact sizes is actually not that trivial, as we have to consider that the monitors the experiments will be conducted on have a specific pixel resolution. Choosing the artifact size too big or too small can bias the user in his or her decision in whether the original stimulus actually was an artifact (e.g. if the artifact is below pixel size or so big that the filter necessary to hide it needs to be bigger than the screen). We therefore choose the minimal artifact size to be at least five pixels on the screen and at most 5% of the screen size (in our case 30 pixels). In-between we set two additional sample points at 10 and 20 pixels.

Silhouette Patterns The patterns formed by the shadow map depend on the angles of the object silhouette on the shadow map. If a silhouette is horizontally or vertically aligned with the shadow map, artifacts are not visible. In the case of diagonal 45 degree silhouette artifacts are visible at regular intervals, to which we will refer to as *stair pattern*. Cases in-between result in irregular or a mixture

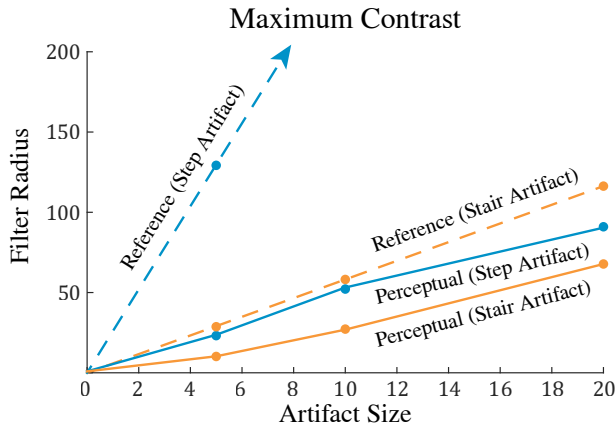


Figure 4: A comparison between reference filter size and perceptual filter size for the investigated artifact patterns (stair and step). The filter size necessary to hide artifacts from users (perceptual filter) is significantly lower than for the reference filter size.

of irregular and regular patterns. We decided to investigate cases where a single artifact (which we will call *step pattern*) is generated and the 45 degree case.

Light-Shadow Contrast The last independent variable we want to investigate is the contrast between lit and shaded areas. We decided to include the worst case scenario, which is the contrast between completely black and white screen pixels. Additionally we reduce the intensity of the lit part by 50% to have an additional sample case.

To summarize, we need to find the right filter size for all artifact size, artifact pattern and light-shadow contrast combinations, resulting in a total of 24 stimuli.

4.3 Task

We decided to use the QUEST procedure [8] to find the threshold at which users can no longer infer from the filtered stimuli whether the original had artifacts in it or not.

We used the Matlab Psycho Toolbox to control the QUESTs. The threshold guess was set to 3% of the artifact size, which is also used as an initial guess and the standard deviation guess. As a probability threshold we used 0.82. The gamma parameter was set to 0.5 and the delta parameter to 0.01. As beta parameter we used 3.5, which was optimized using data obtained by one of the authors performing a beta analysis over 60 trials of the experiment.

5 Evaluation

Due to limited time and resources the study was conducted with a relatively small population of ten users (nine male, one female), all of them were experts in computer-graphics aged 28.2 years on average (standard deviation 3.1 years).

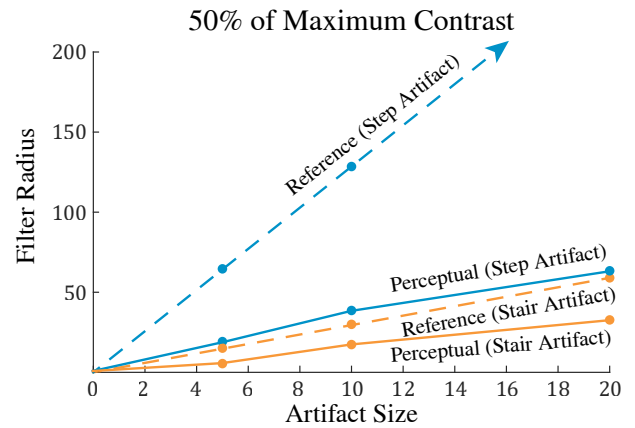


Figure 5: Reducing the contrast makes it slightly harder to spot artifacts as can be seen in comparison to the results in Figure 4.

Based on the user study's results we can observe the impact of the different parameters on the optimal filter size.

Impact of Artifact Size As already indicated by our observations conducted in Section 3, the artifact size proportionally corresponds to the filter radius that is required to hide them. Figure 4 shows a dependence between filter size and artifact size that is nearly linear. While the stair-artifacts of size 10 are hidden using a filter size of ~ 25 pixels or larger, the 20 pixel artifacts on the other end require a filter of at least 50 pixels.

Impact of Patterns The study shows that the single artifact pattern needs a larger filter size in general to be hidden from the user. Figure 4 shows that the stair patterns of size 10 pixels are perceived as a straight contour when the filter size is at or above 25 pixels. Using the same filter size single step-artifacts of the same size are still identifiable by the users. We assume that the regularity of the 45 degree pattern is beneficial to the user's perception of a straight contour.

Comparison to Reference Filter Size In order to measure the actual benefits of the perceptual approach, we need to compare it to a reference solution. We decided to use the same setup we employed in our user study, with the assumption that in the worst case artifacts will be noticed if at least one pixel differs from the expected outcome. In other words if the rasterization of two filtered solutions, one with and one without artifacts, produces the same image (assuming a typical 8 bit representation for intensities), the perfect user will not be able to spot any artifacts. This corresponds to finding the minimal filter size where this condition is met. We will refer to this filter size as the *reference filter size*.

Impact of Light-Shadow Contrast Reducing the contrast between the lit- and the umbra region makes it slightly harder for the user to identify artifacts as can be seen in Figure 5.

In the next section we will use the data gathered by the user study to fit a linear function which describes the perceptually optimal relation between filter- and artifact size.

6 Applications

While the previous Sections investigated the impact of changes of given parameters on the optimal filter radius, real-time shadowing setups need to approach the problem from the opposite direction, that is, to calculate the optimal parameters for a certain filter radius.

Given an arbitrary scene, we have to assume the worst case of artifact pattern and the worst case of light-shadow intensity to appear, hence the only parameter left to find is the right artifact size which is governed by the shadow map’s sampling resolution.

Fitting a linear function to the results shown in Figure 4 (blue line) we get Equation 1, which allows us to calculate the optimal filter radius r to a shadow map by multiplying the pixel size a (artifact size) with the slope c of the linear function.

$$r = a \cdot c \quad (c = 3.47) \quad (1)$$

In order to conduct a practical evaluation of our observations, we implemented a real-time rendering environment and applied Equation 1 to dynamically resize the resolution of a shadow map. We demonstrate the effectiveness of reduced shadow map resolution by applying PCF as well as PCSS filtering. In the case of PCF the filter always needs to have a radius of 3.47 times the pixel size. To use our findings for PCSS we first need to map the penumbra onto the shadow map of the light source to obtain the filter radius. Then the optimal shadow map can be computed by dividing the shadow map with the maximally allowed artifact size. Because the penumbra size can differ within parts of the scene and shadow map artifacts should be avoided, we need to use the maximum shadow map size calculated for each surface point visible for the viewer that lies in shadow.

These rendered results are shown in Figure 7. For each filtering method we show two situations in particular, small and large filter size and compare the reduced shadow map resolution rendering to the unreduced reference rendering side-by-side. While the results look almost identical, we observe an increase in frame-rates on a Geforce GTX 960 GPU by up to 100%.

Poisson Disc Sampling A common approach to reduce the amount of shadow map samples needed to compute soft shadows is to utilize randomly rotated Poisson disc kernels. We will shortly discuss why we expect our findings to be usable in Poisson disc based algorithms as well.

Let’s first consider that Poisson disc sampling introduces noise in the soft shadow, which makes it harder for the user to perceive the iso-contours discussed in Section 4.1 (the noise obfuscates the contours). We therefore expect our findings to be compatible with such algorithms, as artifacts should be even less noticeable when they are used.

7 Limitations And Future Work

Although our implementations show promising results there are some noticeable limitations: In cases where the penumbra width is large, high frequency geometry details might be omitted. An example is shown in Figure 6. One solution to overcome this problem, would be to use a second shadow map dedicated for high frequency geometry.

In cases where the penumbra width is very small, i.e. for hard shadows or contact shadows, the penumbra width has to be enlarged, because the required resolution would tend to be infinitely large. Schwärzler et al. describe a possible solution to this problem in their adaptive light source subdivision approach [7].

Due to limited resources we conducted our user study on a small group of ten expert users. We expect to further reduce the shadow map’s footprint by questioning inexperienced users.

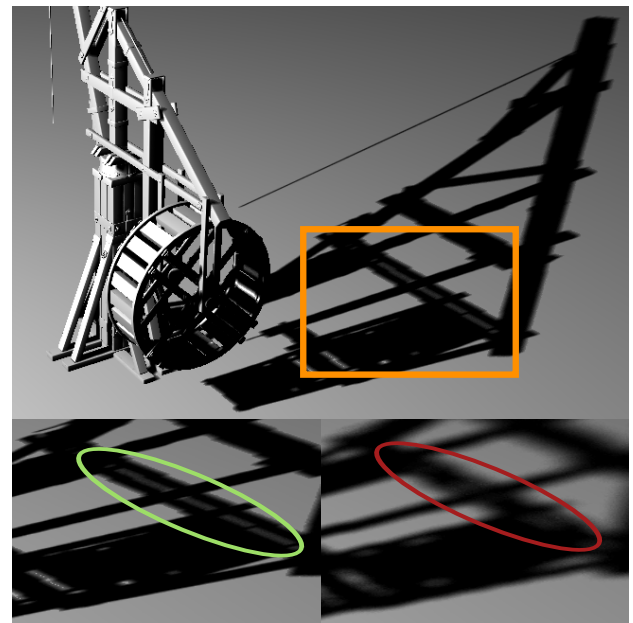


Figure 6: Lowering the resolution might have the unwanted side-effect of detail being omitted.

8 Conclusions

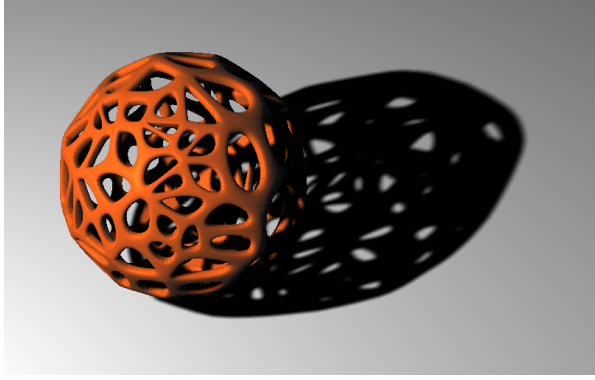
We investigated how soft shadow filtering can be exploited to hide shadow mapping related artifacts. Reducing the

complex feature space of shadow perception allowed us to design a user study to find out at which point filtered artifacts become unnoticeable to users. By interpreting the results of the user study we were able to describe the connection between shadow filter width and shadow map resolution from a perceptual point-of-view with a function that can be used in practical shadow mapping setups. When we applied this function to common shadow filtering algorithms, we were able to save resources by using perceptually optimized algorithms (as can be seen by comparing the shadow map resolutions and *fps*-timings in Figure 7).

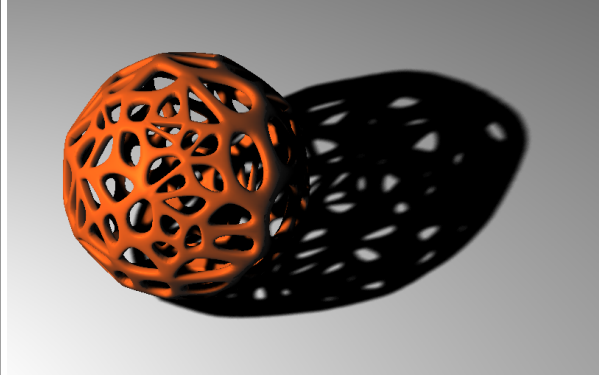
Our findings can be used to dynamically adjust shadow map resolution in real-time, or to calculate a feasible shadow map resolution tailored to a desired penumbra width.

References

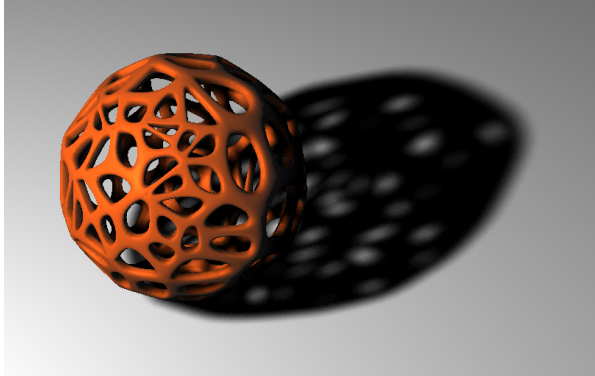
- [1] Thomas Annen, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. Convolution shadow maps. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 51–60. Eurographics Association, 2007.
- [2] Thomas Annen, Tom Mertens, Hans-Peter Seidel, Eddy Flerackers, and Jan Kautz. Exponential shadow maps. In *Proceedings of graphics interface 2008*, pages 155–161. Canadian Information Processing Society, 2008.
- [3] William Donnelly and Andrew Lauritzen. Variance shadow maps. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 161–165. ACM, 2006.
- [4] Randima Fernando. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, page 35. ACM, 2005.
- [5] Michael Hecher, Matthias Bernhard, Oliver Mattausch, Daniel Scherzer, and Michael Wimmer. A comparative perceptual study of soft shadow algorithms. *ACM Transactions on Applied Perception*, 11(5):5:1–5:21, June 2014.
- [6] William T Reeves, David H Salesin, and Robert L Cook. Rendering antialiased shadows with depth maps. In *ACM Siggraph Computer Graphics*, volume 21, pages 283–291. ACM, 1987.
- [7] Michael Schwarzler, Oliver Mattausch, Daniel Scherzer, and Michael Wimmer. Fast accurate soft shadows with adaptive light source sampling. In *Proceedings of the 17th International Workshop on Vision, Modeling, and Visualization (VMV 2012)*, pages 39–46. Eurographics Association, November 2012.
- [8] Andrew B. Watson and Denis G. Pelli. Quest: A bayesian adaptive psychometric method. *Perception & Psychophysics*, 33(2):113–120, 1983.
- [9] Lance Williams. Casting curved shadows on curved surfaces. In *ACM Siggraph Computer Graphics*, volume 12, pages 270–274. ACM, 1978.



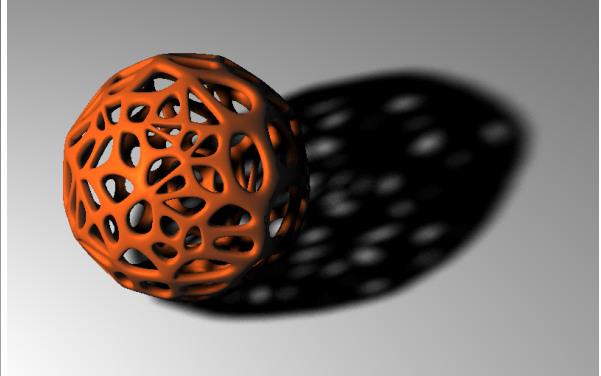
(a) PCF 2048×2048 , 193.4 fps



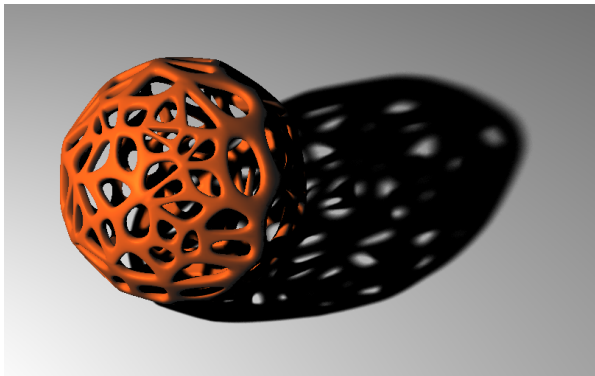
(b) PCF 598×1244 , 294.2 fps



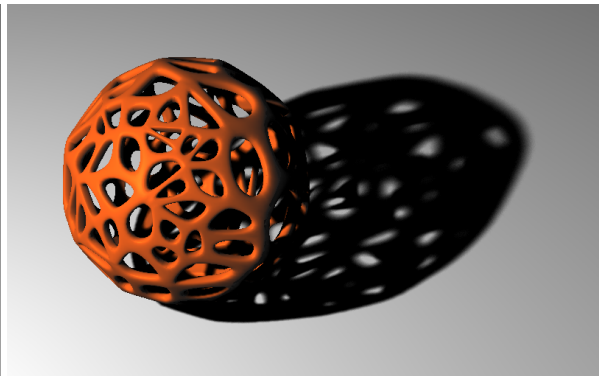
(c) PCF 2048×2048 , 157.4 fps



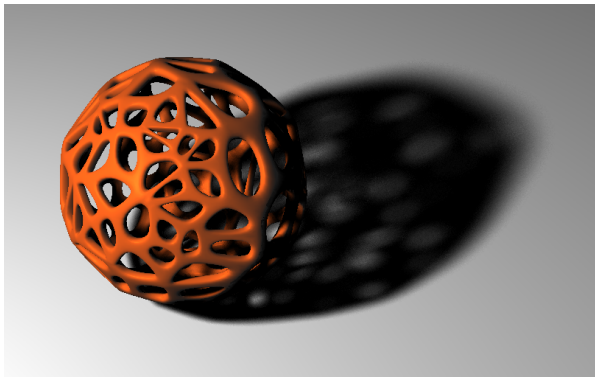
(d) PCF 199×414 , 290.5 fps



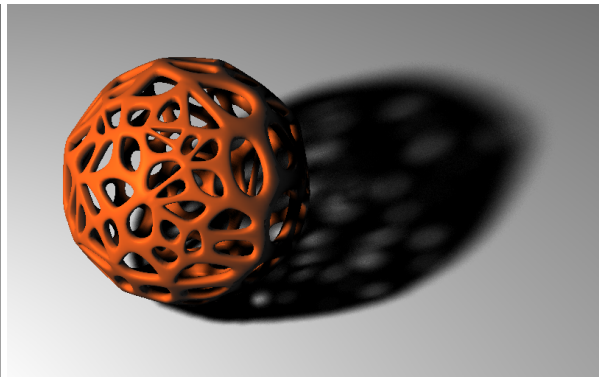
(e) PCSS 2048×2048 , 192.4 fps



(f) PCSS 582×1211 , 307.5 fps



(g) PCSS 2048×2048 , 163.2 fps



(h) PCSS 190×395 , 303.4 fps

Figure 7: Side-by-side comparison of the same scene rendered with large and reduced shadow map resolution. The shadow map resolutions and achieved frame-rates can be seen in the respective captions.

Acceptable System Latency for Gaze-Dependent Level of Detail Rendering

Michał Chwesiuk*

Krzysztof Wolski†

Supervised by: Radosław Mantiuk‡

West Pomeranian University of Technology
Szczecin / Poland

Abstract

The human visual system is unable to perceive all details in the entire field of view. High frequency features are noticeable only at a small angle of 1-2 degrees around the viewing direction. Therefore, it is a reasonable idea to render a coarser object representations for the parafoveal and peripheral visions. A core problem of this gaze-dependent level-of-detail rendering is the minimisation of the system latency. In this work we measure how fast the whole process of rendering and visualisation should be to prevent that a level-of-detail change will be visible for human observers. We noticed that even for distant periphery, the change from coarser to fine object representation should take less than 24 ms. It can be obtained only in systems equipped with the high-end eye tracker and a display with a refresh rate of 120 Hz or faster.

Keywords: system latency, gaze-contingent display, level-of-detail, LOD, eye tracking, real time computer graphics

1 Introduction

One goal of the *level-of-detail* (LOD) technique is to quickly change between coarse and fine representations of the object geometry [3]. Objects with smaller number of polygons can be rendered faster than their fine representation. Therefore, if an object occupies limited number of pixels in the final rendering or it is merely visible, it is more efficient to use its coarser representation. The goal is to find a proper level of detail for an object, taking into account its visibility on the screen.

The human visual system (HVS) is unable to perceive all details in the entire field of view. High frequency features are noticeable only at a small angle of 1-2 degrees around the viewing direction, otherwise details are imperceptible. Areas outside foveal are called parafoveal and peripheral regions. This nonlinear sensitivity of the eye

is defined by the gaze-dependent contrast sensitivity function (CSF) [15], which models the sensitivity to contrast as function of eccentricity (i.e. distance from the gaze direction).

In order to increase performance of rendering, it is a reasonable idea to render the coarser object representations for the parafoveal and peripheral visions. In the gaze-contingent graphics systems, information about the gaze direction must be delivered to the rendering engine. The angular distance between momentary gaze location and position of the object in the screen space, will be a determinant of the model simplification.

A core problem with the implementation of such systems is the minimisation of the *system latency*. The gaze direction must be captured by the eye tracker, the image must be rendered, and finally the display device needs some time to present the image on the screen. If the total processing time would be too long, the observer could see the object changing between the coarse and fine representation. In this work we investigate how short the system latency should be to make LOD modifications imperceptible to a human observers.

We perform a perceptual experiment, in which two geometric objects are rendered on one side of the screen. The first object consists of a large number of polygons and acts as the reference (or fine) representation. The second object is its simplified (or coarse) version with a reduced number of polygons. We asked observers to look at the marker located on the opposite side of the screen. The eye tracker is used to detect the moment, in which observer turns his/her eyes to look at the objects. In this moment, the image is redrawn with both objects using the fine representation. The task of the observer is to identify which of the objects were rendered with the reduced number of polygons. During the experiment we changed the display refresh rate to differentiate the system latencies.

In Section 2 of this paper we provide basic information regarding the human peripheral vision, eye tracking, and latencies of the gaze-dependent rendering systems. We present our gaze-dependent LOD rendering environment and review the previous work related to similar systems in Section 3. Section 4 presents details of the conducted ex-

*mchwesiuk@wi.zut.edu.pl

†krwolski@wi.zut.edu.pl

‡rmantiuk@wi.zut.edu.pl

periment and we discuss acceptable latencies of the gaze-dependent LOD rendering in Section 4.4. The paper ends with conclusions and propositions for future work in Section 5.

2 Background

2.1 Visual resolution

The *visual resolution* of the human eye is measured in terms of contrast sensitivity [8]. A stimulus consisting of the alternating bars of a grating (e.g. Gabor pattern) is presented to observers. They decide what contrast is needed to see the bars at each frequency, while the contrast is defined as the difference in brightness between light gray and dark gray bars. The threshold contrast values as a function of spatial frequencies form the *contrast sensitivity function* (CSF, [2]).

However, people can see details with the frequency defined by the CSF only in a small viewing angle, which subtends 1-2 degrees around the gaze direction. The loss of visual resolution increasing of viewing angle is caused by decreasing number of cones (light-sensitive cells) in the parafoveal and peripheral regions of the retina. This trait is described by a *gaze-dependent contrast sensitivity function*, which shows how contrast sensitivity varies as a function of distance from the fovea [15, 8]. Fig. 1 presents a plot of the perceptible signal frequency as a function of eccentricity. This frequency defines the highest frequency of the Gabor pattern, which is still recognized by human observer. An important observation is that the visual resolution decreases rapidly for higher spatial frequencies and e.g. for a eccentricity of 20 degrees it becomes one-tenth of the maximum resolution. A typical 22-inch LCD display is seen in a viewing angle of 40 degrees, therefore, the geometry of the object located at the screen corner can be significantly reduced if the observer does not look directly at it.

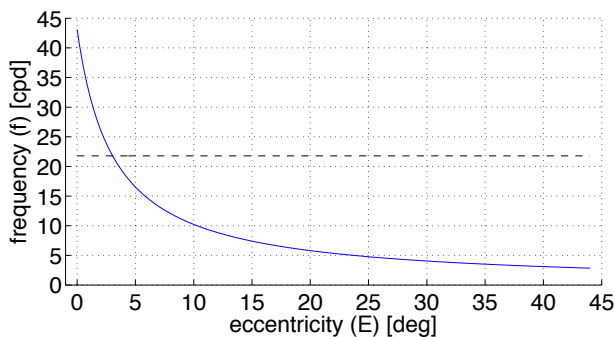


Figure 1: Visible spatial frequencies as a function of viewing angle (the plot is based on the formulas delivered in [8]). The dashed line shows the maximum frequency of a typical LCD display.

2.2 Eye tracking

The principle of eye tracking is based on the observation that the pupil follows the gaze direction during eye movement [4]. Therefore, the location of the pupil centre can be used to estimate the gaze direction. A popular technique employed to localized the pupil center is the modeling of the iris shape (for an excellent review of models for the eye detection we refer to Hansen and Ji work [5]). The eye tracker camera captures an image of the eye. The location of the pupil centre is detected in this image. This location must be transformed from the camera space to the screen space to estimate the gaze position on the screen. This is done using a polynomial transformation as a mapping, which parameters are determined during eye tracker calibration. During calibration, people are asked to look at the target points displayed on the screen. Then, known locations of the target points and data captured by the eye tracker are used to compute the polynomial coefficients. Finally, this polynomial is applied to transform the pupil centre from the camera to screen space.

The human visual system scans the surrounding with its eyes to build a complete view of the environment. The rapid repositioning of the pupil (called saccadic movement) can reach up to 900°/sec. To capture this movement, the eye tracker should work with a latency less than 5 ms [12], which is equivalent to a frequency of 200 Hz. In practical systems, this frequency needs to be even higher, because of the additional time needed to render and display the image.

2.3 System latency

The gaze-dependent rendering system uses the gaze direction captured by the eye tracker to control the image rendering process. For example, an object's geometry can be simplified if the object is positioned far away from the gaze location. Gaze-dependent systems work in real time, i.e. the image redrawing (including its visualisation) must be imperceptible to the human. According to Loschky and McConkie [12, 9], the *latency* of such systems should be less than 22 ms (5 ms for gaze capture and rendering, and additional 17 ms for visualisation on a 60 Hz display).

As shown by Saunders and Woods [17], the latency of the gaze-dependent rendering system ranges from 12 ms for CRT display, 18 ms for DLP projectors, to over 30 ms for low quality LCD displays. However, high-end LCDs with a short display lag can speed-up this process to about 18 ms, which is enough for the gaze-dependent LOD simplification. Loschky et al. [10] measured the system latency using a technique proposed in [1]. They report mean latency of 20 ms for the 1000 Hz EyeLink eye tracker working with a 85 Hz CRT display.

In this work, we use a LCD display, which has a maximum display frequency of 144 Hz (or a display latency of less than 7 ms).

2.4 Previous work

An early work on the gaze-dependent level-of-detail was presented by Mark Levoy [6]. The complexity of the volumetric data was reduced to speed-up the volume rendering method. The author used a precomputed pyramid of 3D texture volumes to skip some complex data structure that were far away from the viewing direction.

In Ohshima et al. [14] a concept for the visual acuity was proposed. This model examines the central/peripheral vision, kinetic vision, and fusional vision to cluster objects of low acuity and render them using simplified versions. The model was tested using a head tracker.

Reddy [16] investigates the perceptual content of a computer-generated image in terms of spatial frequency. The level-of-detail of each object is based on a screen-based measure of the degree of spatial detail which the user can perceive at different distance, angular velocity, and the degree to which it exists in the peripheral field. The author reports a factor 4.5 improvement in frame rate. Reddy also proposes a polygon simplification framework to complement the use of perceptually modulated LOD. However, it is not clear how this framework was used and whether a eye tracker was applied during experiments.

Watson et al. [18] studied the effect of peripheral LOD degradation on the visual search performance. He used a head mounted display to show a high resolution inset within a low resolution display field. The obtained results indicate that the area of the high detail central inset is a significant factor in search performance. However, Watson suggests that visual spatial and chrominance complexity can be reduced by almost half without degrading perceived quality.

A remote monocular eye tracker was used in [11] to measure the viewers real- time gaze location. The authors developed a classic LOD technique, in which objects geometry is simplified according to eccentricity.

In Murphy and Duchowski [13] objects degradation is applied nonisotropically, i.e. only a parts of large object are smoothly reduced. A three-dimensional spatial degradation function is obtained from human subject experiments and applied directly to object geometries prior to rendering. The technique was implemented in the rendering system integrated with a binocular eye tracker. The results indicate a frame rate improvement ranging from a factor of at least 2, up to a 15-fold gain in performance over full resolution display.

Players perception to level of detail (LOD) changes while playing a computer game is investigated in Lopez et al. [7]. The simplified models were unrelated to the task assigned to the player and located away from the area in which the task was being accomplished. Thus, a perception of LOD modification was tested under the inattention blindness. The results show that players were able to detect only about 15% of LOD changes during the game.

In this work we implemented the LOD simplification approach similar to Luebke [11] technique. However, our

main goal is to investigate a perception of the LOD change in a real time rendering application. Therefore, we used apparatuses and techniques that enable the fastest rendering and visualisation of an image possible.

3 Gaze-dependent LOD

3.1 Implementation

Fig. 2 illustrates a gaze-dependent rendering and visualization system. The observer looks on the display. Her/his gaze direction is captured by the eye tracker, which computes the gaze point location on the screen. The graphics engine uses this gaze location to render the scene. The scene contains objects whose complexity depends on the eccentricity. The object close to the gaze point consists of a larger number of triangles than its simplified version seen from a high angle.

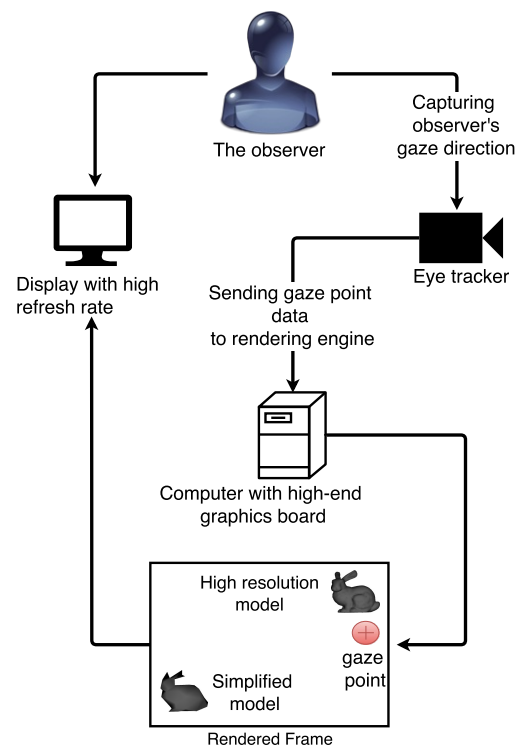


Figure 2: Diagram of the gaze-dependent rendering system.

3.1.1 Eye tracker

In our rendering system we use the Mirametrix S2 eye tracker equipped with a 60 Hz camera. The S2 is a portable device, which should be placed under the display in front of the observer. Before each session the eye tracker must be calibrated. After successful calibration the software

sends gaze locations to our eye tracker communication server using the TCP/IP protocol (see Fig. 3), which collects the gaze data and send them to the rendering engine using the shared memory. We obtained an accuracy of Miramatrix S2 close to 1 degree of visual angle, which is sufficient for our experimental application.

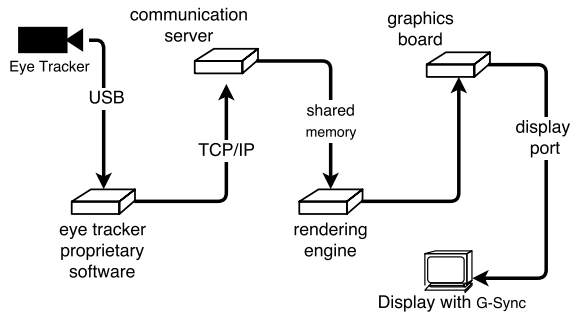


Figure 3: Hardware and software architecture.

3.1.2 Rendering framework

We developed a test framework for fast visualisation of the complex objects. This application is able to render objects consisting of more than 4k triangles in less than 1 ms. It was implemented in C++ and is based on the OpenGL library supported by GLEW and GLFW extensions. The application applies Phong shading and 16-samples multi-sample antialiasing.



Figure 4: Example screenshot from our application. The green cross on the left side depicts location of the gaze point captured by the eye tracker.

In Fig. 4 an example screenshot from our application is presented. The Stanford Bunny models are rendered in 0.89 ms. The object at the top was reduced to 100 triangles, while the bottom one consists of 4k triangles.

3.1.3 Visualisation

The rendered images are displayed on the fast LCD with a display lag of 1 ms and the maximum screen refresh rate



Figure 5: Hardware setup used in the experiment.

of 144 Hz. This display is equipped with the G-Sync electronic, which has a positive impact on the gaze-dependent LOD systems that works on slower hardware. However, we do not use this feature because our rendering application is fast enough to finish the calculations in the required time interval. The main advantage is that G-Sync-supported displays were the fastest commercially available LCD display at the time when we performed our experiments (recently, 165 Hz displays were issued).

4 Experiment

The main goal of the experiment was to find the acceptable system latency, i.e. how fast the object should be redrawn on the display after changing the LOD level to avoid that a human observer would notice this change.

4.1 Procedure

The observer sat in front of the display and used the chinrest adopted from an ophthalmic slit lamp to stabilise her/his eyes in 75 cm distance from the screen (see Fig. 5). The experiment started with a 9-point calibration of the eye tracker. This procedure took about 20 seconds and involved observation of the markers displayed in different areas of the screen. The data processing related to the calibration and further gaze location computation was performed by the proprietary eye tracker software.

During the actual experiment the observer was asked to look at a red cross presented on a 18% grey background (see Fig. 6, top row). After half a second two objects were shown on the left side of the screen in 10°, 20°, or 35° angular distances from the red cross (see Fig. 6, middle row). One of the objects was composed of a large number of polygons and considered as a reference. We reduced the mesh complexity of the second object to a number of polygons that prevent distinguishing this object from the reference. This simplification depends on the angular distance between the observer's gaze point and the object (based on the lower resolution of the human eyes in the periphery). The objects were displayed above each other. Each time it

was randomly chosen whether the high or low resolution mesh would be displayed at the top.

Then, the observer's task was to look at the objects and decide which one was a simplified version. She/he pressed the up/down cursor buttons to indicate the choice. As the observer's gaze were captured by eye tracker, we could replace the simplified version of the object with the reference consisting of 4000 polygons as soon as the gaze moved away from the initial position (see Fig. 6, bottom row). More precisely, we switched the level-of-detail when the gaze location moved by 4 degrees from the initial position.

Our eye tracker operates at a 60 Hz frequency, i.e. we were able to replace objects not earlier than after a 17 ms delay. Additional latency derived from the display. We tested the display working at 30 Hz, 60 Hz, 120 Hz, and 144 Hz, which corresponds to the delays of 33 ms, 16 ms, 8 ms, and 7 ms, respectively.

The experiment was repeated for 3 angular distances and 4 display frequencies resulting in 12 trials per observer. Additionally, we repeated each trial 30 times in random order to obtain averaged results. The experiment was performed in a darkened room. We used 22 ASUS ROG SWIFT PG278Q LCD display with native resolution of 2560 x 1440 pixels. The rendering was performed on a PC equipped with NVIDIA 780 GTX graphic card.

4.2 Stimuli

We generated simplified versions of the Stanford Bunny geometric model using the Quadric Edge Collapse Decimation algorithm in MeshLab¹. The degree of simplification has been chosen in a separate pilot experiment. We searched for a minimum number of polygons, which do not cause the perceptual difference in comparison with the reference model consisting of 4,000 polygons. The experiment was repeated for 3 angular distances because, as we noticed, smaller distances require more precise models. The results of this pilot experiment show that object consisting of 2000, 1600, and 1000 polygons are suitable for 10°, 20°, and 35°, respectively (see Fig. 7).

4.3 Participants

We performed the experiment for a group of 10 volunteer observers (age between 20 and 23 years, 2 females and 8 males). They declared normal or corrected to normal vision and correct color vision. The participants were aware what they should do, but they were naïve about the purpose of the experiment. An average experimental session lasted approximately 12 minutes.

4.4 Results

The results of the experiment are presented in Fig. 8. The plot shows the normalised ratio of correct answers (correct

¹<http://meshlab.sourceforge.net/>

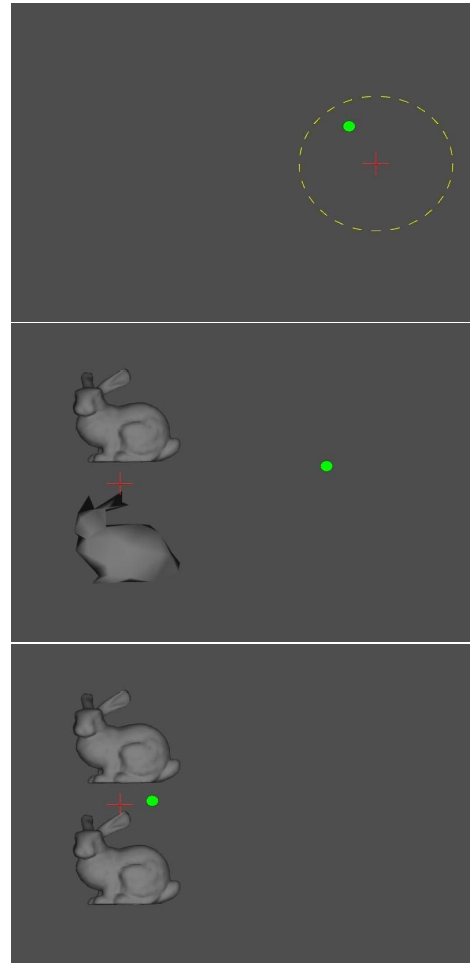


Figure 6: Succeeding phases of the experiment. The green spot depicts the observer's gaze location.

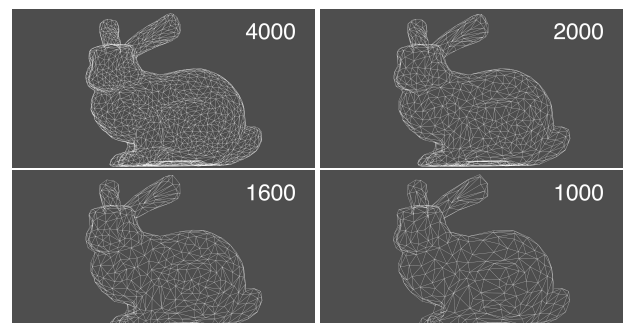


Figure 7: Stanford Bunny reference object (4000 polygons) and its simplified versions with 2000, 1600, and 1000 polygons.

indications on simplified objects) as a function of the display frame rate. The ratio of 0.5 (horizontal dashed line in Fig. 8) is equivalent to the random choice, i.e. indicates inability to distinguish between reference and simplified models. In our study only for the display refresh rate of 144 Hz and the angular distance of 35° the results are close to this line. In all other cases the system latency was too long to ensure imperceptible change of the level-of-detail. Especially, for smaller viewing angles the redrawing is clearly visible.

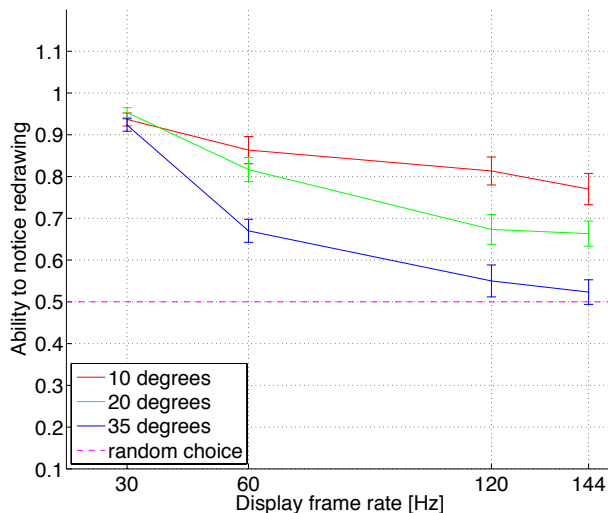


Figure 8: Results of the perceptual experiment. The error bars show the standard error of the mean.

5 Conclusions and Future Work

In this work we conducted a comprehensive evaluation of the acceptable system latency for the gaze-dependent LOD rendering. Our study included a psychophysical experiment which allowed us to evaluate perception of the LOD change for various display refresh rates, ranging from 33 to 7ms, and for different viewing angles. In the experiment we used a fast 144 Hz display but also slow a 60 Hz eye tracker, which introduced additional 17 ms delay. The results of the experiment show that the total system latency in our gaze-contingent system is too long for the imperceptible LOD change. Only for the angular distance of 35° and the latency close to 24 ms (17 ms for eye tracker and 7 for display), the LOD redrawing was unnoticeable for observers.

In future work we plan to test faster eye trackers, which captures the gaze location in less than one refresh cycle of the display (less than 7 ms in our case). We also plan to develop a technique of the LOD blending instead of immediate switching object's geometry from coarse to fine. This solution would increase the acceptable system latency. Finally, we would like to perform a user study of the gaze-dependent level-of-detail rendering in a complex computer

game environment.

References

- [1] Jean-Baptiste Bernard, Scherlen Anne-Catherine, and Castet Eric. Page mode reading with simulated scotomas: A modest effect of interline spacing on reading speed. *Vision research*, 47(28):3447–3459, 2007.
- [2] Fergus W Campbell and JG Robson. Application of fourier analysis to the visibility of gratings. *The Journal of physiology*, 197(3):551–566, 1968.
- [3] James H Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, 1976.
- [4] Andrew T. Duchowski. *Eye Tracking Methodology: Theory and Practice (2nd edition)*. Springer, London, 2007.
- [5] Dan Witzner Hansen and Qiang Ji. In the eye of the beholder: A survey of models for eyes and gaze. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):478–500, 2010.
- [6] Marc Levoy and Ross Whitaker. Gaze-directed volume rendering. *ACM SIGGRAPH Computer Graphics*, 24(2):217–223, 1990.
- [7] Francisco Lopez, Ramon Molla, and Veronica Sundstedt. Exploring peripheral lod change detections during interactive gaming tasks. In *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization*, pages 73–80. ACM, 2010.
- [8] Lester Loschky, George McConkie, Jian Yang, and Michael Miller. The limits of visual resolution in natural scene viewing. *Visual Cognition*, 12(6):1057–1092, 2005.
- [9] Lester C Loschky and George W McConkie. Investigating spatial vision and dynamic attentional selection using a gaze-contingent multiresolutional display. *Journal of Experimental Psychology: Applied*, 8(2):99, 2002.
- [10] Lester C Loschky, Ryan V Ringer, Aaron P Johnson, Adam M Larson, Mark Neider, and Arthur F Kramer. Blur detection is unaffected by cognitive load. *Visual cognition*, 22(3-4):522–547, 2014.
- [11] David Luebke and Carl Erikson. View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 199–208. ACM Press/Addison-Wesley Publishing Co., 1997.

- [12] George W McConkie and Lester C Loschky. Perception onset time during fixations in free viewing. *Behavior Research Methods, Instruments, & Computers*, 34(4):481–490, 2002.
- [13] Hunter Murphy and Andrew T Duchowski. Gaze-contingent level of detail rendering. *EuroGraphics 2001*, 2001.
- [14] Toshikazu Ohshima, Hiroyuki Yamamoto, and Hideyulu Tamura. Gaze-directed adaptive rendering for interacting with virtual space. In *Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996*, pages 103–110. IEEE, 1996.
- [15] Eli Peli, Jian Yang, and Robert B Goldstein. Image invariance with changes in size: The role of peripheral contrast thresholds. *JOSA A*, 8(11):1762–1774, 1991.
- [16] Martin Reddy. *Perceptually modulated level of detail for virtual environments*. PhD thesis, University of Edinburgh. College of Science and Engineering. School of Informatics., 1997.
- [17] Daniel R Saunders and Russell L Woods. Direct measurement of the system latency of gaze-contingent displays. *Behavior research methods*, 46(2):439–447, 2014.
- [18] Benjamin Watson, Neff Walker, and Larry F Hodges. Managing level of detail through head-tracked peripheral degradation: a model and resulting design principles. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 59–63. ACM, 1997.

State of the Art

State of the Art in Real-time Registration of RGB-D Images

Patrick Stotko*

Supervised by: Tim Golla†

Institute of Computer Science II - Computer Graphics
University of Bonn
Bonn / Germany

Abstract

3D reconstruction gained more and more interest in the last years due to low-budget depth scanning devices and high-performance graphics hardware. Several approaches were developed in order to get accurate reconstructions in real-time at high frame rates and opened up new applications in the field of augmented reality (AR) and 3D scanning by providing immediate feedback to the user. But since measurements are corrupted with noise, registration of the captured RGB-D images needs to be very accurate to get high quality results. In this state of the art report, we will review several registration algorithms that were developed in the last years and compare their performance.

Keywords: State of the Art, Real-time, Registration, RGB-D, GPU, Kinect, 3D Reconstruction, Comparison

1 Introduction

During the last years, increasing effort was spent in the field of 3D reconstruction using depth cameras. The origin of this rising interest lies in the availability of low-budget, high-resolution depth sensors like the Microsoft Kinect. One of the first and most prominent systems in this field was KinectFusion, a 3D reconstruction framework developed by Izadi et al., (2011) and Newcombe et al., (2011). This system is capable of producing highly detailed 3D models from RGB-D images in real-time by integrating the captured data into the volumetric data structure of Curless and Levoy, (1996) in parallel on the GPU.

However, it has several limitations. In KinectFusion, Izadi et al., (2011) and Newcombe et al., (2011) used a full 3D volumetric grid to store the reconstruction. Since GPU memory is quite limited, creating large scale reconstructions was not possible. To overcome this issue, several techniques to reduce memory consumption were developed including Moving Volume approaches (Roth and Vona, 2012; Whelan et al., 2012), fast data structures (Laine and Karras, 2010; Zeng et al., 2012; Zeng et al., 2013) and streaming algorithms (Chen et al., 2013). Recent work

done by Nießner et al., (2013) came to the result that by using hash tables only relevant regions of the volume needed to be stored leading to great advances in terms of performance and efficiency.

The other major drawback of the initial KinectFusion system is its registration algorithm. Before the captured data can be integrated into the volume, it must be transformed into the global coordinates system in which the volume is specified. Therefore, an estimate of the six-dimensional camera pose is needed. Unfortunately, even small errors in the estimation cause significant drift in the final reconstruction since those errors accumulate rapidly. The origin of this problem lies in the uncertainty of the data that is captured by the sensor. Usually, these data contain noise whose exact nature is still unknown. So a perfect registration is not possible and much effort has been spent to obtain good reconstructions in the end.

The purpose of this state of the art report is to review current approaches that tried to solve this problem. We also investigate how well the solutions fit into our reconstruction pipeline (Stotko and Golla, 2015) that is based on the VoxelHashing framework of Nießner et al., (2013). In particular, we will focus on:

1. Reviewing several approaches and explaining their strengths and weaknesses
2. Analyzing how efficient they can be implemented on the GPU
3. Comparing the different algorithms in terms of accuracy and efficiency

2 Problem Statement

Before we start to present and compare the different algorithms, we first need to formally state the problem that should be solved.

2.1 Preliminaries

In our settings, we want to reconstruct 3D geometry automatically and in real-time using a low-cost depth sensor. Such sensors have been becoming popular in the past years because they offer moderate reconstruction quality

*stotko@cs.uni-bonn.de

†golla@cs.uni-bonn.de



Figure 1: Example of an RGB image (left) and a depth image (right) captured by a RGB-D camera, taken from Henry et al., (2012).

at a cheap price. The probably most prominent one is the Kinect from Microsoft. Besides a depth image I_d , it also captures an RGB image I_{RGB} at a particular resolution, e.g. 640×480 in case of the Kinect:

$$I_{RGB}: \Omega \subset \mathbb{N}^2 \rightarrow \mathbb{N}^3 \quad (1)$$

$$I_d: \Omega \subset \mathbb{N}^2 \rightarrow \mathbb{R} \quad (2)$$

We require these two maps to be time-synchronized and aligned. Since the cameras have a small offset, some sensor-specific transformations for aligning them may be performed in a preprocessing step. Figure 1 shows an example of such an RGB-D image. The 3D position $\mathbf{v} \in \mathbb{R}^3$ of a pixel $\mathbf{p} \in \Omega$ can then be computed by back-projection as

$$\mathbf{v}(\mathbf{p}) = \left(\frac{p_x - c_x}{f_x} I_d(\mathbf{p}), \frac{p_y - c_y}{f_y} I_d(\mathbf{p}), I_d(\mathbf{p}) \right)^\top \quad (3)$$

using the focal length $\mathbf{f} = (f_x, f_y)$ and the principal point $\mathbf{c} = (c_x, c_y)$ of the camera. On the other hand, the pixel \mathbf{p} into which the point \mathbf{v} falls is computed as the projection

$$\mathbf{p}(\mathbf{v}) = \left(\frac{f_x v_x}{v_z} + c_x, \frac{f_y v_y}{v_z} + c_y \right)^\top \quad (4)$$

These two transformations can also be written compactly by considering the intrinsic camera calibration matrix of the sensor

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (5)$$

and the projection operator

$$\mathbf{H}((x, y, z)^\top) = (x/z, y/z)^\top \quad (6)$$

Then, we can express the two projections by

$$\mathbf{v}(\mathbf{p}) = \mathbf{K}^{-1} \cdot \mathbf{I}_d(\mathbf{p}) \cdot (\mathbf{p}, 1)^\top \quad (7)$$

$$\mathbf{p}(\mathbf{v}) = \mathbf{H}(\mathbf{K} \cdot \mathbf{v}) \quad (8)$$

2.2 The Registration Problem

Our reconstruction pipeline is based on the system developed by Nießner et al., (2013). Like in their work, we use an efficient hash table on top of a set of voxel blocks to reconstruct the captured scene in a sparse volumetric grid. To perform registration, we use ray-casting to extract the implicitly stored isosurface from the volume and use the resulting RGB-D image as an estimate of the model. This image can further be processed for rendering by some shading kernels.

Now, the only missing part is the registration algorithm. Given new measurements $[I_{RGB}^{(t+1)}, I_d^{(t+1)}]$ at time step $t+1$, we wish to find the new camera transformation matrix

$$\mathbf{T}^{(t+1)} = \begin{bmatrix} \mathbf{R}^{(t+1)} & \mathbf{t}^{(t+1)} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \quad (9)$$

consisting of a rotation $\mathbf{R}^{(t+1)} \in \mathbb{R}^{3 \times 3}$ and a translation $\mathbf{t}^{(t+1)} \in \mathbb{R}^3$. Since we also know this transformation from the previous time step t , namely the current model $[\hat{\mathbf{I}}_{RGB}^{(t)}, \hat{\mathbf{I}}_d^{(t)}]$, this problem can be reduced. We only need to align the two images to get our desired result. A successful alignment will give us the incremental transformation $\Delta \mathbf{T}$. This optimization might be performed in global coordinates which is usually done when dealing with points clouds. But since we are trying to align measurements from a camera, the local camera coordinates system at time step t is the better choice and also used in some of the approaches. Notice that the local system at time step $t+1$ is moving and thus not very appropriate as a reference. In this context, we have the following relation:

$$\mathbf{T}^{(t+1)} = \mathbf{T}^{(t)} \cdot \Delta \mathbf{T} \quad (10)$$

So we first move the new measurements by $\Delta \mathbf{T}$ such that they are aligned with the previous data in their respective coordinate system and then transform them to global coordinates using $\mathbf{T}^{(t)}$. Defining the transformation in this

way, we get an intuitive meaning of the incremental transformation by rearranging the equation:

$$\Delta T = (T^{(t)})^{-1} \cdot T^{(t+1)} \quad (11)$$

For any newly measured point given in the local camera coordinate system at time $t + 1$, we can infer its coordinates in the local system at time t by applying the incremental transformation.

In the following, we will discuss three families of registration algorithms: Iterative Closest Point, Normal Distribution Transform and Sparse Methods.

3 Iterative Closest Point

We start our discussing with some popular approaches from the family of the Iterative Closest Point algorithms.

3.1 KinectFusion ICP

The original registration algorithm used in KinectFusion is a variant of the popular Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992; Chen and Medioni, 1992). In our setting, the vertex maps obtained by back-projection of the depth maps describe the scene in the local camera coordinate system at time t :

$$\hat{\mathbf{v}}_i^{(t)} := \hat{\mathbf{v}}^{(t)}(\mathbf{p}_i) \quad (12)$$

$$\mathbf{v}_j^{(t+1)} := \Delta T \cdot \mathbf{v}^{(t+1)}(\mathbf{p}_j) \quad (13)$$

For simplicity, we omit the conversion to homogeneous coordinates.

In KinectFusion, the newly captured RGB-D image at time step $t + 1$ is now aligned to the current model at time step t . It is rather important to distinguish between frame-to-frame and frame-to-model tracking. The latter is much more robust against noise since the model is very smooth and contains much less noise than an input image. Therefore, frame-to-model tracking is used in the ICP error function:

$$E_{icp} = \sum_i \sum_j w_{ij} \langle T \cdot \mathbf{v}_j^{(t+1)} - \hat{\mathbf{v}}_i^{(t)} | \hat{\mathbf{n}}_i^{(t)} \rangle^2 \quad (14)$$

This formulation is the general point-to-plane error function which describes the quadratic distance of the point $T \cdot \mathbf{v}_j^{(t+1)}$ to the tangent plane of $\hat{\mathbf{v}}_i^{(t)}$ that is defined by its normal $\hat{\mathbf{n}}_i^{(t)}$. The coefficients w_{ij} describe the level of correspondence between each pair. To find the global minimum of this error function, the weights w_{ij} and the transformation T need to be optimized jointly. Unfortunately this is an NP-hard problem, so in the ICP algorithm the weights and the transformation are optimized separately. The cost of this simplification is the loss of the global optimum, only a local one can be guaranteed.

3.1.1 Finding Correspondences

Optimizing the weights w_{ij} is further simplified using the heuristic ICP performs. It is assumed that there exists a set of correspondence pairs between the two points sets. From this assumption, it follows that the weights are binary and the error function can be simplified by using only one sum over the correspondence set:

$$E_{icp} = \sum_i \langle T \cdot \mathbf{v}_i^{(t+1)} - \hat{\mathbf{v}}_{c(i)}^{(t)} | \hat{\mathbf{n}}_{c(i)}^{(t)} \rangle^2 \quad (15)$$

This has the great advantage that the complexity is drastically reduced of being only linear instead of quadratic.

The initial objective was to optimize the general error function with respect to the weights. For our simplified formulation, this problem has an intuitive solution. Each term of the error is minimal if we choose for each point $\mathbf{v}_i^{(t+1)}$ the closest point $\hat{\mathbf{v}}_{c(i)}^{(t)}$ as the corresponding point.

KinectFusion further exploits the fact that these points are generated from a camera with known intrinsic parameters. Computing the correspondences is therefore done by the projective data association algorithm (Blais and Levine, 1995). Here, the definition of closest point is not related to being close in the three-dimensional space. Instead, for each vertex $\mathbf{v}^{(t+1)}(\mathbf{p}_i)$, which is defined in the local coordinate system at time $t + 1$, we compute the vertex $\hat{\mathbf{v}}_{c(i)}^{(t)}$ at time t that is at the same line of sight in the local system at time t . This means, we first need to know the coordinates of vertex $\mathbf{v}^{(t+1)}(\mathbf{p}_i)$ in the other coordinate system by applying ΔT , then project it to pixel coordinates and use this pixel $\mathbf{p}_{c(i)}$ as a look up for the corresponding vertex:

$$\hat{\mathbf{v}}_{c(i)}^{(t)} = \mathbf{v}^{(t)}(\mathbf{p}(\Delta T \cdot \mathbf{v}^{(t+1)}(\mathbf{p}_i))) \quad (16)$$

An example of this procedure is shown in Figure 2. If the two point sets are already perfectly aligned, then the

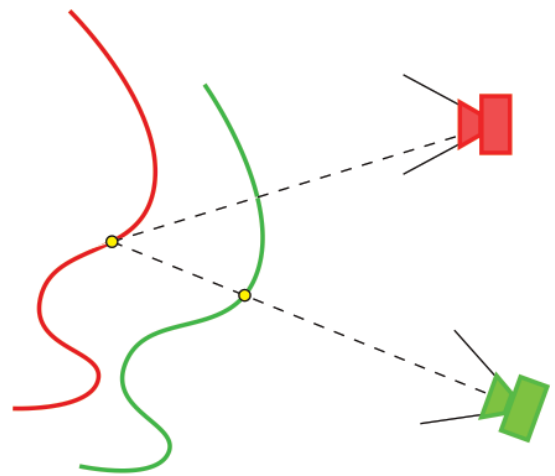


Figure 2: The Projective Data Association Operator. Correspondence pairs are found by projection to the previous camera coordinate system.

correspondence coincides with the input vertex and thus it is also the closest point in space.

3.1.2 Optimizing the error function

After having calculated the correspondences, the actual error function can be optimized. So we wish to find a rigid transformation T that minimizes the error between the correspondences. However, the rotation R of T has three degrees of freedom but nine unknown values to optimize for. In order to have a minimal representation, the following one is commonly used:

$$\xi = (\omega^\top, t^\top)^\top \in \mathbb{R}^6 \quad (17)$$

This vector can be transformed to the original matrix by

$$T = \tilde{\xi} \quad , \quad \tilde{\xi} = \begin{bmatrix} \exp[\omega]_\times & t \\ \mathbf{0}^\top & 0 \end{bmatrix} \quad (18)$$

For consistency with further approaches, we only consider the exponential mapping of the rotation and directly model the translation. This will help us to compare the approaches and will not affect the accuracy.

Since only small rotations are assumed, the exponential mapping can be approximated linearly:

$$\exp[\omega]_\times = \sum_{n=0}^{\infty} \frac{([\omega]_\times)^n}{n!} \approx I + [\omega]_\times \quad (19)$$

By plugging this approximation into the error function and rearranging the terms, one gets similarly to the derivation of Low, (2004) the following result:

$$E_{icp} = \sum_i \left(\begin{bmatrix} \mathbf{v}_i^{(t+1)} \times \hat{\mathbf{n}}_{c(i)}^{(t)} \\ \hat{\mathbf{n}}_{c(i)}^{(t)} \end{bmatrix}^\top \xi + \langle \mathbf{v}_i^{(t+1)} - \hat{\mathbf{v}}_{c(i)}^{(t)} | \hat{\mathbf{n}}_{c(i)}^{(t)} \rangle \right)^2 \quad (20)$$

$$= \|J_{icp} \xi + r_{icp}\|_2^2 \quad (21)$$

This is a standard equation system that can be solved by the normal equation. The main advantage of such an equation system here is the independence between the pairs of vertices. Each row of the Jacobian and the residual can thus be calculated separately in parallel which makes it very suitable for a GPU implementation. In such an implementation, one would directly compute the components of the normal equation:

$$J_{icp}^\top J_{icp} = \sum_i J_i^\top \cdot J_i \quad , \quad J_{icp}^\top r_{icp} = \sum_i J_i^\top \cdot r_i \quad (22)$$

$$J_i = \begin{bmatrix} \mathbf{v}_i^{(t+1)} \times \hat{\mathbf{n}}_{c(i)}^{(t)} \\ \hat{\mathbf{n}}_{c(i)}^{(t)} \end{bmatrix}^\top \quad , \quad r_i = \langle \mathbf{v}_i^{(t+1)} - \hat{\mathbf{v}}_{c(i)}^{(t)} | \hat{\mathbf{n}}_{c(i)}^{(t)} \rangle \quad (23)$$

Each summand is thus be computed independently in parallel and the total sum is then be obtained by a tree reduction. Since the memory cost is only linear in the number of

correspondences, the total memory consumption is manageable. This property is very important since GPU's have only a very limited amount of memory compared to CPU's. Minimizing the footprint of such an algorithm is therefore a very important task and is also indirectly a criterion for being able to run in real-time.

In order to prepare for the next steps, we multiply both the Jacobian and residual by -1 . This does not change the result but results in a nice notation:

$$J_i = - \begin{bmatrix} \mathbf{v}_i^{(t+1)} \times \hat{\mathbf{n}}_{c(i)}^{(t)} \\ \hat{\mathbf{n}}_{c(i)}^{(t)} \end{bmatrix}^\top = (-\hat{\mathbf{n}}_{c(i)}^{(t)})^\top \cdot [-\mathbf{v}_i^{(t+1)}]_\times \quad I \quad (24)$$

$$r_i = - \langle \mathbf{v}_i^{(t+1)} - \hat{\mathbf{v}}_{c(i)}^{(t)} | \hat{\mathbf{n}}_{c(i)}^{(t)} \rangle = \langle \hat{\mathbf{v}}_{c(i)}^{(t)} - \mathbf{v}_i^{(t+1)} | \hat{\mathbf{n}}_{c(i)}^{(t)} \rangle \quad (25)$$

Later we will come back to this point and show what this reformulation actually means how it can be used.

The final step is to solve the equation system. Both components of the normal equation are only six-dimensional so this step is performed very efficiently on the CPU by a Cholesky decomposition. Finally, the estimated parameters ξ can be used to update the incremental transformation:

$$\Delta T \leftarrow \begin{bmatrix} \exp[\omega]_\times & t \\ \mathbf{0}^\top & 0 \end{bmatrix} \cdot \Delta T \quad (26)$$

As mentioned at the beginning, these steps must be performed iteratively until either the error falls below a threshold or a maximum number of iterations is reached. To improve the result, typically a three-level coarse-to-fine scheme is used where the images are step-wise filtered and down-sampled to lower resolutions. During the optimization, one starts at the coarsest level and then refines the solution until reaching the finest level. This helps to avoid local minima and guides the optimization towards the global optimum. However, there is still no guarantee to reach it.

3.2 Photometric ICP

The KinectFusion system was tested in several scenarios and while the accuracy is quite high in smaller scenes, it drops significantly on larger ones. Especially if the amount of characteristic features is low, e.g. on walls, only using the depth information is not sufficient to compute the camera pose. Steinbrücker et al., (2011) and Kerl et al., (2013b) used the color information to overcome this.

The main idea here is that if the camera has only moved a little bit like it is shown in Figure 3, the appearance of scene is the same on both images. Physically, this means the color of an object is the same and therefore its reflectance behavior does not change. This results in the assumption the whole scene mainly reflects perfectly diffuse and that specular parts can be neglected. For mirror-like objects, this assumption would of course fail, but usually walls, tables and most other objects can be approximately modeled

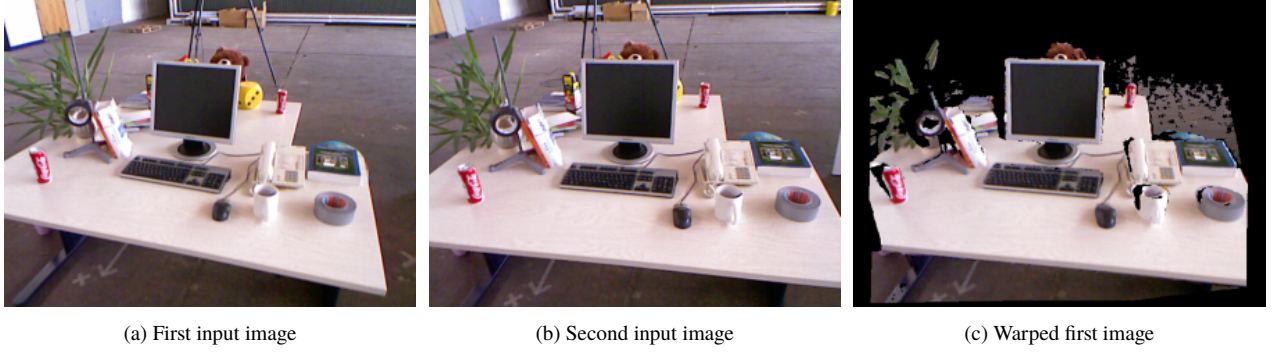


Figure 3: Example of warping, taken from Steinbrücker et al., (2011).

in this way. So based on this assumption, the so called photometric error can be formulated as

$$E_{rgb d} = \sum_i \left(I_g^{(t)}(w(\xi, p_i)) - I_g^{(t+1)}(p_i) \right)^2 \quad (27)$$

where $I_g^{(t)}, I_g^{(t+1)}$ are the intensity images computed from the RGB images at time steps t and $t + 1$. Special care about the choice of the coordinate system has to be taken here. As we will see later, we need to use the same system as in the KinectFusion ICP. So unlike in the original paper, we use this formulation to describe the error. The vector $w(\xi, p_i)$ is the warped pixel and defined according to the incremental transformation ξ :

$$w(\xi, p_i) = \Pi(K \cdot \tilde{\xi} \cdot \Delta T \cdot v^{(t+1)}(p_i)) \quad (28)$$

Intuitively, the warp takes a vertex from time step $t + 1$ and transforms it into the local camera coordinate system at time t . Then, it is moved according to the optimization parameters and finally projected to pixel coordinates. A correct alignment would move the vertex in such a way that after projection the intensity value at the corresponding pixel is the same as the one of the starting pixel.

Figures 3 and 4 show an example of this idea. Note that the RGB image of the model is not used here since it is not as precise as the captured one and therefore frame-to-frame tracking is performed for the color values. To prepare for the next step, the warped point is written as a mapping:

$$v(\xi, p_i) = \tilde{\xi} \cdot (\Delta T \cdot v^{(t+1)}(p_i)) = \tilde{\xi} \cdot v_i^{(t+1)} \quad (29)$$

One important note has to be mentioned at this point. $v_i^{(t+1)}$ is the same point as in the KinectFusion ICP error and the warping function w is essentially the mapping used to compute the correspondences. Knowing this, the Photometric ICP is in fact an ICP that operates on intensity images. The error can now be rewritten as

$$E_{rgb d} = \sum_i (d(\xi, p_i))^2 \quad (30)$$

$$d(\xi, p_i) = I_g^{(t)}(\Pi(K \cdot v(\xi, p_i))) - I_g^{(t+1)}(p_i) \quad (31)$$

Like in KinectFusion, one wants to apply the Gauss-Newton method to solve the problem iteratively. This requires that the error function needs to be linearized which is done by a Taylor series of the parameters ξ :

$$d(\xi, p_i) \approx d(0, p_i) + \frac{\partial d}{\partial \xi}(0, p_i) \cdot \xi \quad (32)$$

$$= \left(I_g^{(t)}(\Pi(K \cdot v_i^{(t+1)})) - I_g^{(t+1)}(p_i) \right) + \frac{\partial d}{\partial \xi}(0, p_i) \cdot \xi \quad (33)$$

The first term is the current residual r_i and states the difference in the intensity values under the current alignment. The other term is the derivative of the energy with respect to ξ , an 1×6 matrix and the i -th Jacobian j_i . By using the chain rule the Jacobian can be is given as

$$\frac{\partial d}{\partial \xi}(0, p_i) = \frac{\partial I_g^{(t)}}{\partial \Pi} \cdot \frac{\partial \Pi}{\partial K} \cdot \frac{\partial K}{\partial v} \cdot \frac{\partial v}{\partial \xi}(0, p_i) \quad (34)$$

$$= \nabla I_g^{(t)} \cdot \frac{\partial \Pi}{\partial (x, y, z)^\top} \cdot K \cdot \frac{\partial v}{\partial \xi}(0, p_i) \quad (35)$$

Thus, one needs to compute the image gradient

$$\nabla I_g^{(t)} = \left(\frac{\partial I_g^{(t)}}{\partial x}, \frac{\partial I_g^{(t)}}{\partial y} \right) \in \mathbb{R}^{1 \times 2} \quad (36)$$

the derivative of the dehomogenization

$$\frac{\partial \Pi}{\partial (x, y, z)^\top} = \begin{pmatrix} \frac{1}{z} & 0 & -\frac{x}{z^2} \\ 0 & \frac{1}{z} & -\frac{y}{z^2} \end{pmatrix} \in \mathbb{R}^{2 \times 3} \quad (37)$$

and the derivative of the moving point $\frac{\partial v}{\partial \xi} \in \mathbb{R}^{3 \times 6}$ we are optimizing for. We found that there exists a compact formula for computing the last derivative. Gallego and Yezzi, (2015) showed a nice derivation of this formula and also considered the important case with $\xi = 0$:

$$\frac{\partial v}{\partial \omega_i}(0, p_i) = \frac{\partial \tilde{\xi}}{\partial \omega_i}(0) \cdot v_i^{(t+1)} \quad (38)$$

$$= \frac{\partial \exp[\omega]_\times}{\partial \omega_i}(0) \cdot v_i^{(t+1)} \quad (39)$$

$$= [e_i]_\times \cdot v_i^{(t+1)} \quad (40)$$

$$= [-v_i^{(t+1)}]_\times \cdot e_i \quad (41)$$

$$\frac{\partial \mathbf{v}}{\partial t_i}(\mathbf{0}, \mathbf{p}_i) = \frac{\partial \tilde{\boldsymbol{\xi}}}{\partial t_i}(\mathbf{0}) \cdot \mathbf{v}_i^{(t+1)} \quad (42)$$

$$= \frac{\partial \mathbf{t}}{\partial t_i} \quad (43)$$

$$= \mathbf{e}_i \quad (44)$$

Here, \mathbf{e}_i is the i -th basis vector of the standard basis of \mathbb{R}^3 . More compactly, these derivatives can be written in matrix form by

$$\frac{\partial \mathbf{v}}{\partial \boldsymbol{\xi}}(\mathbf{0}, \mathbf{p}_i) = \begin{bmatrix} [-\mathbf{v}_i^{(t+1)}]_{\times} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{3 \times 6} \quad (45)$$

And here we see the great similarity to the final formula for the normal equation components of the KinectFusion ICP given in equation (24). Also the residual term coincide with the one given in equation (25). In both cases, we were able to reformulate the problem such that it can be solved by the Gauss-Newton method and thus calculate a Jacobian. Here, the Jacobian is obtained directly by using the chain rule while in the KinectFusion ICP setting it is derived by using the small angles assumption. Now, we can also state what the last reformulation means. The derivative $\frac{\partial \mathbf{v}}{\partial \boldsymbol{\xi}}(\mathbf{0}, \mathbf{p}_i)$ of the moving point is also be present there and the term before it, namely the negative transposed normal $(-\hat{\mathbf{n}}_{c(i)}^{(t)})^\top$, is the derivative of the point-to-plane error.

To sum this up, the photometric energy can now also be written like the point-to-plane energy:

$$E_{rgb} = \|\mathbf{J}_{rgb} \boldsymbol{\xi} + \mathbf{r}_{rgb}\|_2^2 \quad (46)$$

So we can solve it by the Gauss-Newton method in exactly the same way as before. As mentioned before, Kerl et al., (2013b) use a slightly different approach in the first iteration of their DVO SLAM system. While the Jacobian matrix is the same as here, they additionally weight it using some statistical distributions. The weights come from the Maximum-A-Posteriori formulation and help to improve the results. In their approach, this is a t-distribution. Its covariance matrix is used for weighting but must be computed beforehand. In addition, a motion prior is used

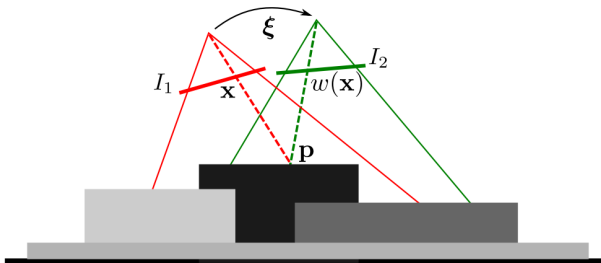


Figure 4: The warping function w shown here for a pixel \mathbf{x} is essentially a correspondence mapping, taken from Kerl et al., (2013b). Note that in our setting the roles of two images are changed meaning that red indicates the new and green the previous measurements.

guiding the optimization to prefer small motions. So while this variant can also be implemented efficiently, the authors already state that approximately twice the run-time is needed.

3.3 Combined ICP

So far, we showed two different approaches and their similarities. While the photometric variant of the ICP avoids the use of noisy depth data, it suffers from the assumption that the surfaces reflect ideally diffuse. Fortunately, both error functions use the same correspondences, are solved in the same way, defined in the same coordinate system and their limitations do not affect each other. Whelan et al., (2013) used these observations and combined the two error function so that all given information is used, the depth and the RGB image. Basically, the two error functions are weighted by a factor and then summarized to a combined Jacobian matrix and residual vector:

$$E_{combined} = E_{icp} + \lambda E_{rgb} \quad (47)$$

$$(\mathbf{J}_{icp}^\top \mathbf{J}_{icp} + \lambda \mathbf{J}_{rgb}^\top \mathbf{J}_{rgb}) \boldsymbol{\xi} = \mathbf{J}_{icp}^\top \mathbf{r}_{icp} + \sqrt{\lambda} \mathbf{J}_{rgb}^\top \mathbf{r}_{rgb} \quad (48)$$

The computation costs are of course higher than using only one of the error terms. However, both have linear complexity in time and memory consumption which means that the total cost increases only by a constant factor. On the other hand, one can exploit the similarities between the two errors and speed-up the computation of the Jacobians.

This idea was further used in Kintinuous (Whelan et al., 2015b) and ElasticFusion (Whelan et al., 2015a). Kintinuous is an extension of the KinectFusion system that use a moving volume to allow large scenes to be reconstructed. ElasticFusion drops the volumetric structure and uses surfels for reconstruction. However, it still fuses measurements and therefore combines the fusion principle from the KinectFusion world with the surfel structure that is typically used in the family of the Normal Distribution Transform. Kerl et al., (2013a) also tried to combine both error metrics. In the second and current iteration of their DVO SLAM system, the Maximum-A-Posteriori approach is again used to solve the problem. In a post-processing step, all three system perform loop closure to achieve global consistency of the reconstructed model.

3.4 Generalized ICP

We continue our review of registration algorithms in the family of the ICP approaches by some probabilistic variants. As already introduced, Kerl et al., (2013b) converted the ICP energy into a Maximum-A-Posteriori problem. Similarly, Segal et al., (2009) presented a probabilistic variant which they called Generalized ICP. The basic idea is that the two sets of measurements \mathcal{A}, \mathcal{B} , which we want

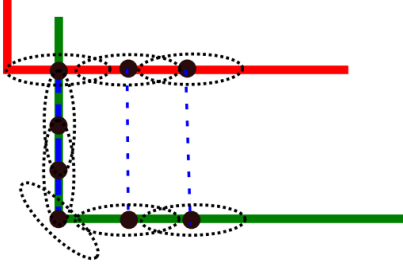


Figure 5: Influence of the covariance matrices on the plane-to-plane error, taken from Segal et al., (2009).

to align, are drawn from underlying sets $\hat{\mathcal{A}}, \hat{\mathcal{B}}$ probabilistically according to the normal distributions

$$\mathbf{v}_i^{(t+1)} \sim N(\mathbf{v}_{i_{opt}}^{(t+1)}, \Sigma_i^{(t+1)}) \quad (49)$$

$$\hat{\mathbf{p}}_j^{(t)} \sim N(\hat{\mathbf{p}}_{j_{opt}}^{(t)}, \Sigma_j^{(t)}) \quad (50)$$

This allows to model uncertainty in the measurements. Therefore instead of minimizing an energy, a Maximum-Likelihood estimate is computed:

$$\mathbf{T}^* = \arg \max_{\mathbf{T}} \prod_i p(d_i^{(T)}) = \arg \max_{\mathbf{T}} \sum_i \log(p(d_i^{(T)})) \quad (51)$$

with distances

$$d_i^{(T)} = \hat{\mathbf{p}}_{c(i)}^{(t)} - \mathbf{T} \cdot \mathbf{v}_i^{(t+1)} \quad (52)$$

Now it is assumed that both points are drawn from independent normal distributions, it follows that

$$d_i^{(T)} \sim N(0, \Sigma_{c(i)}^{(t)} + \mathbf{T} \Sigma_i^{(t+1)} \mathbf{T}) \quad (53)$$

Using this observation, the optimization problem can be formulated as

$$\mathbf{T}^* = \arg \min_{\mathbf{T}} \sum_i (d_i^{(T)})^\top \cdot (\Sigma_{c(i)}^{(t)} + \mathbf{T} \Sigma_i^{(t+1)} \mathbf{T})^{-1} \cdot d_i^{(T)} \quad (54)$$

To solve this, one first needs to know what the covariance matrices $\Sigma_{c(i)}^{(t)}, \Sigma_i^{(t+1)}$ are. If we use for example the configuration $\Sigma_{c(i)}^{(t)} = \mathbf{I}, \Sigma_i^{(t+1)} = \mathbf{0}$, the error functions reduces to the point-to-point distance where the square of the Euclidean distances between all point pairs is considered. In a similar way, also the point-to-plane distance can be constructed from this formulation. Therefore, this approach can be seen as a generalization of the standard ICP.

In order to define the so called plane-to-plane distance, the covariance matrices are chosen in a way such that the variance in the tangent plane is constant in every direction but small along the normal direction. For a normal that points along the x -axis, the covariance matrix would thus be

$$\Sigma_i = \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \epsilon \in [0, 1] \quad (55)$$

Other covariances can be created from this one by using rotation matrices that map the normal to the x -axis. This is also shown in Figure 5. Here, the covariance matrices are visualized using ellipses and indicate uncertainty that is high in the tangent plane but quite low in normal direction.

Generalized ICP only changes the energy formulation and keeps the other steps unchanged. This means that the correspondences can be computed in the same way as for the other variants. However, the optimal transformation must now be computed using the Conjugate Gradient algorithm. Fortunately, there exist GPU implementations of this algorithm and those only need to operate on the correspondence set. Therefore Generalized ICP should also be quite efficient and fast enough to be included in our reconstruction pipeline. Since its optimization method is different from the other ones, the performance in terms of accuracy will be interesting, so we will compare this later more carefully.

3.5 Direct Volume Matching

Another quite interesting variant of doing frame-to-model tracking is to use the volumetric data structure of our 3d reconstruction framework directly. So instead of ray-casting the model from the volume and matching the new RGB-D image to it, one directly matches the incoming image to the volume. Bylow et al., (2013) and Canelhas et al., (2013) investigated this possibility and adjusted the Gauss-Newton solver to operate on the implicitly stored surface.

This surface is stored in the volumetric data structure and represented by a function

$$D_t: \mathbb{Z}^3 \rightarrow [-1, 1] \quad (56)$$

that maps voxels to truncated signed distance values. The three-dimensional space is therefore discretized and a signed distance field is constructed by this function. The surface itself is encoded as the zero-set of this function. Every time a new depth map gets integrated into the volume, the signed distance function D_t is updated:

$$D_{t+1}(\mathbf{x}) = \frac{D_t(\mathbf{x}) W_t(\mathbf{x}) + d_{t+1}(\mathbf{x}) w_{t+1}(\mathbf{x})}{W_t(\mathbf{x}) + w_{t+1}(\mathbf{x})} \quad (57)$$

$$W_{t+1}(\mathbf{x}) = W_t(\mathbf{x}) + w_{t+1}(\mathbf{x}) \quad (58)$$

To align the new data, an energy based on the signed distance can now be formulated:

$$E_{volume} = \sum_i (D_t(\tilde{\xi} \cdot \mathbf{v}_i^{(t+1)}))^2 \quad (59)$$

Here, the fact is exploited that the signed distance not only gives a distance to the surface but also a direction. Thus, one is able to distinguish whether the current voxel is inside or outside the object. Like in the Photometric ICP this distance can now be linearized by a Taylor expansion

$$D_t(\tilde{\xi} \cdot \mathbf{v}_i^{(t+1)}) \approx D_t(\mathbf{v}_i^{(t+1)}) + \nabla D_t(\mathbf{v}_i^{(t+1)}) \cdot \tilde{\xi} \quad (60)$$

So we again can express the error in terms of a Jacobian and a residual:

$$E_{volume} = \|\mathbf{J}_{volume} \boldsymbol{\xi} + \mathbf{r}_{volume}\|_2^2 \quad (61)$$

Unfortunately, this formulation has several drawbacks in contrast to the previous ones. It is of course directly designed for our pipeline and exploits it but from the computational point-of-view it is not very efficient.

First, we consider the formulation more carefully. The signed distance is defined on discrete coordinate while in the error function, the actual 3D position $\mathbf{v}_i^{(t+1)}$ of a point is inserted. This means that the signed distance must be estimated using trilinear interpolation which requires eight look-ups in the volume. In a full volume, this operation would be very fast at the cost of the infeasible memory requirements. To do this in our scenario, we need eight hash table look-ups followed by eight access to the volume. One could sacrifice accuracy by speed when only the distance of the nearest voxel is used. However since drift is very critical in this application, this is also not an option. Even worse, the gradient of the signed distance function has to be computed. Usually, this is done by finite differences. But here, these are differences of trilinear interpolated values so the cost is even higher.

We also see a general limitation of this approach. It can only operate perfectly on full volumes. However, our pipeline exploits the facts the empty space is not needed for reconstruction and thus only stores a small region round the actual surface, called the truncation region. Only if the gradient and the residual are evaluated inside this region, they are correct. Otherwise, we would have to prune them. If the camera however moves quite fast, some important parts may get out of this region and the performance of the tracker would decrease rapidly. Therefore, we believe that this approach will not give the best performance neither in accuracy nor in run time.

4 Normal Distribution Transform

Up to now, we showed several approaches in the family of the ICP algorithm. In the following, we will consider another family of algorithms, the Normal Distribution Transform (NDT) invented by Biber and Straßer, (2003). While ICP algorithms directly align the two point sets, NDT changes the representation of one point set to a mixture of normal distributions and then tries aligns the other set to this model.

4.1 3D-NDT

Whereas the original NDT algorithm only operated on 2D data, Magnusson et al., (2007) extended the framework to 3D. Like in the 2D version, the data is subdivided into smaller sets by a regular volumetric grid. In each cell C_i ,

the points falling in it are modeled by a normal distribution $N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$:

$$\boldsymbol{\mu}_i = \frac{1}{K} \sum_{k=1}^K \hat{\mathbf{v}}_k^{(t)} \quad (62)$$

$$\boldsymbol{\Sigma}_i = \frac{1}{K-1} \sum_{k=1}^K (\hat{\mathbf{v}}_k^{(t)} - \boldsymbol{\mu}_i) (\hat{\mathbf{v}}_k^{(t)} - \boldsymbol{\mu}_i)^\top \quad (63)$$

where K is the number of points in the current cell C_i . For a given point, the likelihood that it is observed in the model can be expressed by the probability density function

$$p_{c(i)}(\mathbf{x}_i) \propto \exp\left(-\frac{(\mathbf{x}_i - \boldsymbol{\mu}_{c(i)})^\top \boldsymbol{\Sigma}_{c(i)}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_{c(i)})}{2}\right) \quad (64)$$

Since the task is to align the two point sets, the likelihood of all points of the new measurement should be high. Thus a score function can be defined in the following way:

$$s_{p2d}(\boldsymbol{\xi}) = - \sum_{i=1}^n p_{c(i)}(\tilde{\boldsymbol{\xi}} \cdot \mathbf{v}_i^{(t+1)}) \quad (65)$$

The advantage of this formulation is that analytic derivatives of the score function exist. Finding the optimal pose $\boldsymbol{\xi}$ that minimizes this function is actually finding a pose where the first derivative of the score function is zero. Like in the original paper, Magnusson et al., (2007) also suggest to use the Newton method and solve the following equation system:

$$\mathbf{H} \boldsymbol{\xi} = -\mathbf{g} \quad (66)$$

Here, $\mathbf{H} \in \mathbb{R}^{6 \times 6}$ denotes the Hessian and $\mathbf{g} \in \mathbb{R}^6$ the gradient of s_{p2d} . In their paper, Magnusson et al., (2007) used the axis-angle representation of a rotation where the angle is considered separately and the axis is always normalized. In this case, they optimized seven instead of six parameters. However, this only leads to a slightly different Hessian and gradient. To be consistent with the other approaches, we use this formulation.

Now, we discuss the strengths and weakness of the 3D-NDT. As a preprocessing step, the normal distributions have to be constructed. Thus, all points need to be associated to their corresponding cells in the regular grid and the mean and covariance of each cell must be calculated. Computing nearest neighbors might be problematic since building search structures such as KD-Trees on the GPU is complicated. Furthermore, the projective behavior of the points is not further considered or exploited which makes the model more general. On the other hand, some optimization potential is lost making the approach possibly slower than others.

As stated in their paper, this representation is much more compact than storing and using all points. So from the computational point of view, the change in the representation introduces additional complexity but fortunately this only

has to be performed on the fixed data set that is not moving. Nevertheless, we have to take this preprocessing into account. Most computation steps including preprocessing and matching can be performed in parallel on the GPU. Like in the ICP approaches, a tree reduction can be used to compute the Hessian and the gradient.

4.2 Color 3D-NDT

One big advantage of using normal distributions is that they can be modeled in any dimension. That was the reason for the extension to 3D and it is also the reason to extend it to incorporate colors. Huhle et al., (2008) developed this extension. A straightforward extension would be to convert colors to the $L^*a^*b^*$ color space, where perceptual differences coincide to the L2-distance, and use a 6D feature vector consisting of the 3D position and the color. Then, only the Hessian and the gradient have to be adjusted and the solution is again found by the Newton method. However, Huhle et al., (2008) observed that a single normal distribution per cell is not able to model the color distribution accurately.

To construct a better model, the authors suggest to use a Gaussian Mixture Model (GMM):

$$p_i(\mathbf{x}^*) = \sum_{j=1}^M \alpha_j N(\mu_{i,j}^*, \Sigma_{i,j}^*) \quad (67)$$

Here, \mathbf{x}^* refers to the color and \mathbf{x} to the 3D position of a point. So for each cell C_i , M normal distributions are computed using Expectation Maximization (EM) with an initial guess calculated from k-means. These color distributions are then used as weighting functions to the 3D-NDT algorithm:

$$w_{ij}(\mathbf{x}_i^*) = \exp\left(-\frac{(\mathbf{x}_i^* - \mu_{c(i),j}^*)^\top \Sigma_{c(i),j}^{*-1} (\mathbf{x}_i^* - \mu_{c(i),j}^*)}{2}\right) \quad (68)$$

Having these weights, the mean and the covariance of the 3D positions of each color distribution are computed. Note that in contrast to the standard 3D-NDT algorithm M normal distributions $N(\mathbf{q}_{ij}, \Sigma_{ij})$ are computed for each cell C_i . The score function is then given as

$$s_{p2d,color}(\xi) = - \sum_{i=1}^n \sum_{j=1}^M w_{ij} p_{c(i),j}(\tilde{\xi} \cdot \mathbf{v}_i^{(t+1)}) \quad (69)$$

where the probability density function is now defined as

$$p_{c(i),j}(\mathbf{x}_i) \propto \exp\left(-\frac{(\mathbf{x}_i - \mu_{c(i),j})^\top \Sigma_{c(i),j}^{-1} (\mathbf{x}_i - \mu_{c(i),j})}{2}\right) \quad (70)$$

As in 3D-NDT, the score can be optimized by the Newton method.

This approach inherits most of the properties of the underlying 3D-NDT algorithm. However, the amount of work per cell is now much higher than before. Huhle et al., (2008) suggest to use $M = 3$ mixture models to guarantee unique poses which requires $2 \cdot 3 = 6$ normal distributions per cell. However, these normal distributions have to be computed by Expectation Maximization and k-means which both can run in parallel on subsets of the data on the GPU. Since the number of iterations varies for each subset, some performance of the GPU might be lost. In addition, the preprocessing step is much more costly and the compactness of the model is also lower than before because now 6 normal distributions per cell are used.

4.3 D2D 3D-NDT

Some years later, Stoyanov et al., (2012) came up with an idea that is similar to the Generalized ICP. In the standard ICP, the point-to-plane distance between two point clouds is minimized and this was generalized to what Segal et al., (2009) called the plane-to-plane distance. Consequently, the question arose whether it was possible to extend the 3D-NDT in the same way meaning that two sets of normal distributions should be registered. The task is therefore to find a transformation that aligns the two models

$$\mathbf{T}(M_{NDT}^{(t+1)}) = \{N(\mathbf{T}(\mu_i), \mathbf{T}(\Sigma_i))\} \quad (71)$$

$$M_{NDT}^{(t)} = \{N(\mu_j, \Sigma_j)\} \quad (72)$$

Distances between two probability density functions can be defined in different ways. However, the standard L2-distance between all possible points is a simple but reasonable choice:

$$d_{L2} = \int \left(p(\mathbf{x} | M_{NDT}^{(t)}) - p(\mathbf{x} | \mathbf{T}(M_{NDT}^{(t+1)})) \right)^2 d\mathbf{x} \quad (73)$$

After rearranging the terms and applying some identities of normal distributions, the L2-distance simplifies to

$$d_{L2} \sim \sum_i \sum_j p(\mathbf{0} | \mathbf{T}(\mu_i) - \mu_j, \mathbf{T}(\Sigma_i) + \Sigma_j) \quad (74)$$

A new score function can now be defined:

$$s_{d2d}(\xi) = - \sum_i \sum_j p(\mathbf{0} | \mathbf{T}(\mu_i) - \mu_j, \mathbf{T}(\Sigma_i) + \Sigma_j) \quad (75)$$

As a consequence of this formulation, the optimization can be performed exactly in the same way as for the original 3D-NDT algorithm. Only the mean and the covariance differ. Like Magnusson et al., (2007), the score function is approximated by only considering the nearest normal distribution for each of the transposed ones. Thus, the matching here is faster than the standard 3D-NDT if the size of the model is significantly smaller than the size of the scan data. On the other hand, two models have to be constructed in a preprocessing step resulting in some

overhead. Similar to the ICP optimization, a coarse-to-fine scheme improves the results by optimizing in different levels of details to hopefully avoid local minima. While in the ICP setting this can be done cheaply by calculating the first levels of the Gaussian pyramid of the RGB-D images, here all points have to be considered in each level making the construction slower. Stoyanov et al., (2012) already recognized that this step is more costly than the actual registration so we have to consider this in our comparison.

4.4 Multi-Resolution Surfel Maps

More work was done to develop a model that allows cheap matching like the D2D 3D-NDT and also incorporate color information as in Color 3D-NDT. The result of this work were Multi-Resolution Surfel Maps invented by Stückler and Behnke, (2014). In their work, the authors extend the probabilistic framework of the Normal Distribution Transform by adding additional information.

First, a mean $\mu \in \mathbb{R}^6$ and a covariance matrix $\Sigma \in \mathbb{R}^{6 \times 6}$ on the shape and color are calculated for each cell. Inspired by the Fast Point Feature Histograms (FPFH) by Rusu et al., (2009), a descriptor $h \in \mathbb{R}^{12}$ is constructed and used as an additional source of information for correspondence finding. To achieve high frame-rates, the authors state that the neighborhood relations between the cells are stored explicitly, so the 26 neighbors can be found in constant time. The surfel maps also provide a multiple resolution structure by using octrees to implicitly support a coarse-to-fine scheme.

In the registration stage, a score function is optimized by first using the Levenberg-Marquardt method to get an initial coarse pose estimate and then refining this estimate by the Newton method. In particular, Stückler and Behnke, (2014) used the same score function as in D2D 3D-NDT but only considered correspondences with sufficiently small L2-distance between the shape-texture descriptors. In a post-processing step, they detect loop closures to further reduce accumulated drift and improve results. We will later discuss this concept more carefully later.

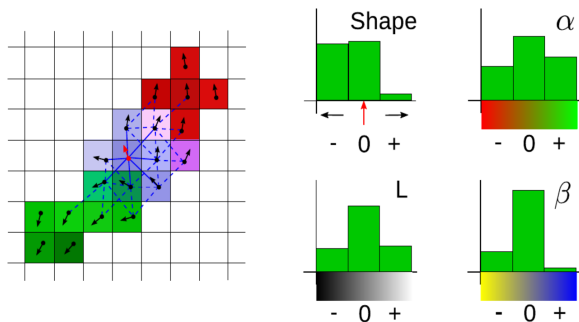


Figure 6: 2D illustration of the shape-texture descriptor, taken from Stückler and Behnke, (2014).

So in principle, Multi-Resolution Surfel Maps are a consequent evolution in the family of the Normal Distribution Transform. Since they inherit most properties from their predecessors, we only need to evaluate how efficient their construction on the GPU can be. Originally, this approach was developed and implemented on the CPU and the authors already argue that they gained significant speed-ups by using all cores of the CPU. However, to use the full potential of GPU's the whole process must run fully in parallel. The only challenging part is the octree structure. In the literature, there are many papers on GPU octrees but the explicitly stored neighbors here are somehow problematic. While they are allowing us to get a constant time lookup, they introduce a randomness in the memory lookups and additional memory costs. And since GPU memory is still quite valuable and GPU's typically operate on coherent data, it is not directly clear how well a GPU implementation of Multi-Resolution Surfel Maps will perform. However, we highlight again that this approach introduces a lot of complexity. That is probably because this family of algorithms can be easily extended to a full reconstruction algorithm and is therefore more than a simple registration algorithm.

5 Sparse Methods

Algorithms that are more oriented in the field of Computer Vision use features to estimate the camera pose. So instead of using all the data and minimizing an error function, one only considers parts of the scene which are informative and characteristic. This is reasonable since only very few points in an image are characteristic enough to describe the transformation like border points or points with a very strong gradient. More generally, these points can be seen as features and the task is to detect enough of them to robustly estimate the camera pose. An example of detected features is shown in Figure 7.

In the following, we will consider the approaches of Endres et al., (2012) and Huang et al., (2011). While the basic idea is the same in both systems, they made different decisions on the kind of features and how they are matched. In particular, the RGB-D SLAM system by Endres et al., (2012) use the combination of SIFT, SURF and ORB features. Since they argue that especially the SIFT features are computationally demanding, a GPU-accelerated version is used. After the features are matched, RANSAC is used to compute the best rigid transformation. To reduce accidental accumulation of drift, the registration is performed on the last three recent frame and seventeen randomly sampled earlier frames. Finally, they detect loop closures and optimize the pose graph to further reduce drift.

A similar technique is used by the foveis algorithm of Huang et al., (2011). But they only rely on FAST features and use a different inlier detection method. Instead of RANSAC, the problem is formulated in a graph of consistent features. To get the best transformation, a



Figure 7: Extracted features shown on the RGB image and the 3D point cloud, taken from Henry et al., (2012).

Maximal-Clique-Problem needs to be solved. Unfortunately, Maximal-Clique is NP-complete, so the exact solution can not be determined efficiently. The authors therefore suggest to use a greedy approximation to solve the problem in less time. To reduce drift, the registration is performed against a reference key-frame instead of the previous frame. It is a common technique and also used in other approaches like in Multi-Resolution Surfel Maps. For global consistency, loop closures are detected.

Unlike all previously discussed algorithms, sparse methods can not so easily be implemented on the GPU. Basically, that class of algorithms can be split into three steps. First, features are computed. This can be done on the CPU if they are fast to evaluate. For computationally more demanding descriptors, like the SIFT descriptor, a GPU version has to be used. In case of the SIFT descriptor, there is such an implementation publicly available (Wu, 2007). Second, the features have to be matched to get a list of pairs. Since the features are independent of each other, this can be parallelized and speeded-up by only considering a reasonably small search window. Third, the best transformation from the pairs has to be found. While RANSAC search the solution by randomly testing candidates, the greedy approximation of Maximal-Clique iteratively improve the found solution. Furthermore, the solver for the transformation parameters ξ must be implemented on the GPU in this case. This is non-trivial and special care has to be taken here to avoid numerical instabilities. But since this method is sparse, CPU versions can also be very fast.

6 Loop Closure

The last point we want to discuss are loop closures. In a few previously considered algorithms, the authors try to achieve global consistency by explicitly modeling this as a loop closure detection problem. This is rather important to prevent implausible reconstructions like it is shown in Figure 8. The basic idea is to build a graph where the nodes represent the camera poses ξ_i of some selected key-frames

and edges between them the incremental transformations ΔT_{ij} . In order to optimize the graph, the corresponding reconstruction representations of the key-frames are attached to them. The optimization itself is usually performed by the generic graph optimization framework of Kümmerle et al., (2011) by minimizing a cost function based on the key-frames.

In the DVO SLAM system of Kerl et al., (2013a) this framework is used. Stückler and Behnke, (2014) also considered loop closures to improve their results. Fusion systems using loop closures that are closer to our pipeline are the Kintinuous (Whelan et al., 2015b) and ElasticFusion (Whelan et al., 2015a) system. While in Kintinuous a moving volume approach is used and the pose graph consisting of cloud slices of this volume is optimized after detecting a loop closure, ElasticFusion uses a surfel-based map representation and also overcomes the need of a pose graph.

However, our VoxelHashing pipeline strongly exploits the fact that the scene is static and thus culls all empty parts of the volume away. This assumption is very restrictive and even the loop closure system of Kintinuous can not be easily used since it would require to deform at least a part of the volume. Therefore, large parts of the hash table have to be changed since voxel blocks are moving. This is a drawback of our pipeline and nevertheless we include this strategy into our evaluation to see how much results improve when considering global consistency.

7 Comparison

In this part, we compare the discussed algorithms. Fortunately, most of them use the RGB-D SLAM benchmark by Sturm et al., (2012) to show their performance. The measurements of the scenes contained in this benchmark are real-world data and also ground truth trajectories are available. In addition, they also define useful error metrics like the absolute and relative pose error and provide a tool for evaluation.

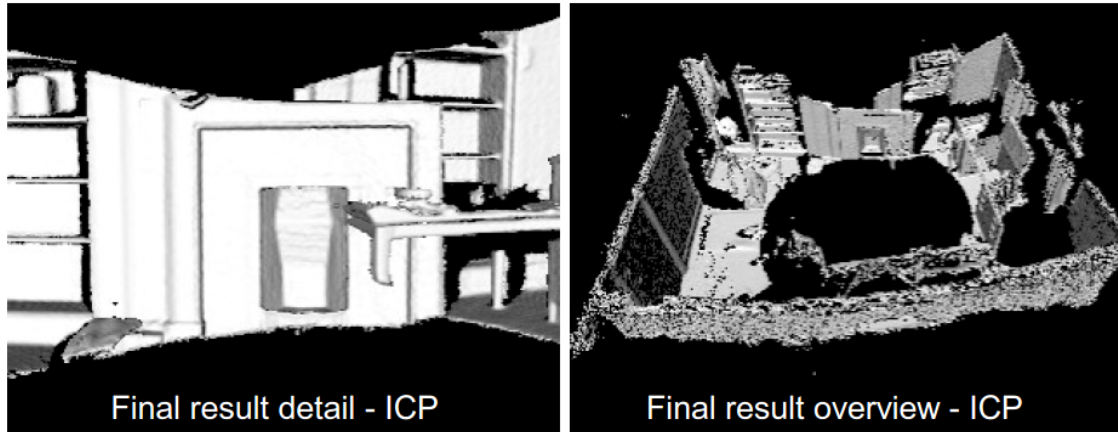


Figure 8: Drift accumulates and gets visible if a loop is closed, taken from Kähler et al., (2015).

However since the benchmark contains many different test scenes, each author of the presented papers have chosen different subsets of the provided scenes making a direct comparison impossible. Thus, we decided to compress the collected results. So instead of simply listing and comparing the error values measured by the authors directly, we use Won-Lost-Tables which are commonly used in the field of Machine Learning to compare the performance of learning algorithms. The main advantage is that they are abstracting from the absolute results and only measure the relative performance between two algorithms for the whole dataset. In particular, for each paper we construct a table that summarizes the results in a way that we can compare each pair of algorithms and decide which one performs better on the test set. Note that we have only done this for papers where the respective authors have tested their approach against more than one of the other approaches we have mentioned in this report and also used enough test scenes.

Constructing these tables is done in the following way. First, we compute the mean error of each algorithm L :

$$\text{ME}(L) = \frac{1}{n} \sum_{i=1}^n e_i(L) \quad (76)$$

From these means, one can now compute the quotient $\text{ME}(L_1) / \text{ME}(L_2)$ which states the relative mean error of algorithm L_1 compared to L_2 . This is our first measurement and it is built for all pairs of algorithms. It compresses the performance across all test cases and tells us how much L_1 is better or worse than L_2 .

The second measurement is constructed by evaluating which algorithms was better at which test case. This means that for each scene one determines if L_1 has won, lost or was equally good against L_2 and accumulates this information. As a result, one gets the number of wins, losses and ties of L_1 against L_2 in the complete test set. Using that information helps to interpret the relative mean error since one large deviation can influence this value quite significantly. On the other hand, the won/lost/ties statistic is

not affected from such deviations and thus helps us to give better interpretations. The number of scenes on which the approaches are evaluated is also implicitly encoded in these tables by summing the wins, losses and ties.

In Table 1 we show the results of our comparison. Here, the ICP variants KinectFusion ICP (Izadi et al., 2011; Newcombe et al., 2011; Nießner et al., 2013), Photometric ICP (Steinbrücker et al., 2011), Combined ICP (Whelan et al., 2013), Kintinuous (Whelan et al., (2015b)), ElasticFusion (Whelan et al., (2015a)), Generalized ICP (Segal et al., 2009) and DVO SLAM (Kerl et al., 2013a) are compared. The NDT approaches are D2D 3D-NDT (Stoyanov et al., 2012) and Multi-Resolution Surfel Maps (Stückler and Behnke, 2014) and from the sparse methods foveis (Huang et al., 2011) and RGB-D SLAM (Endres et al., 2012) are chosen. To ensure an objective comparison, all results are weighted equally.

First, we focus on the results of Whelan et al., (2013). They introduced the Combined ICP and compared its performance against the geometric and the photometric variant. In addition, they also evaluated the foveis system. From the table, one can infer that the Combined ICP is much stronger than the KinectFusion ICP and Photometric ICP. This result is quite intuitive since the motivation of it was that the strengths and the weaknesses of the two approaches were different and a combination would help both parts to perform better. Furthermore, Photometric ICP seems to be also much better than the standard KinectFusion ICP approach. Here, the much lower noise in the RGB images allows the algorithm to be more precise. foveis performs also very well and is on par with the Combined ICP while having a slightly lower mean error. Experiments of Whelan et al., (2013) also showed that especially in cases where only few or no visual and geometric features are available, like in corridor scenes shown in Figure 9, the combination of depth and color data stabilizes the camera pose estimation process and makes it very robust in such scenarios.

Next, we analyze the results of Multi-Resolution Surfel Maps. Interestingly, this approach is considered to be

Table 1: Won-Lost-Tables. If L_c and L_r indicate the column and row algorithms, then the lower triangle shows the number of win-loss-ties of L_c versus L_r whereas the upper triangle indicates the quotient of the average errors $ME(L_r) / ME(L_c)$.

(a) Results built from Stückler and Behnke, (2014) using relative pose error (RPE) median

	Multi-Resolution Surfel Maps	Photometric ICP	Generalized ICP	D2D 3D-NDT	fovis
Multi-Resolution Surfel Maps		0.667	0.646	0.699	0.859
Photometric ICP	18-3-1		0.968	1.049	1.288
Generalized ICP	19-3-0	12-10-0		1.083	1.331
D2D 3D-NDT	20-2-0	15-7-0	9-13-0		1.229
fovis	18-4-0	17-4-1	9-13-0	5-17-0	

(b) Results built from Kerl et al., (2013a) using absolute trajectory error (ATE) root-mean-square error (RMSE)

	DVO SLAM	RGB-D SLAM	Multi-Resolution Surfel Maps	KinectFusion ICP
DVO SLAM		0.629	0.791	0.114
RGB-D SLAM	8-2-0		1.256	0.182
Multi-Resolution Surfel Maps	8-2-0	3-7-0		0.145
KinectFusion ICP	10-0-0	9-1-0	9-1-0	

(c) Results built from Whelan et al., (2013) using relative pose error (RPE) root-mean-square error (RMSE)

	Combined ICP	KinectFusion ICP	Photometric ICP	fovis
Combined ICP		0.233	0.726	1.289
KinectFusion ICP	4-0-0		3.117	5.537
Photometric ICP	4-0-0	0-4-0		1.224
fovis	2-2-0	0-4-0	1-3-0	

(d) Results built from Whelan et al., (2015b) using absolute trajectory error (ATE) root-mean-square error (RMSE)

	Kintinuous	DVO SLAM	RGB-D SLAM	Multi-Resolution Surfel Maps
Kintinuous		1.494	1.010	1.024
DVO SLAM	0-10-0		0.676	0.685
RGB-D SLAM	4-6-0	6-4-0		1.014
Multi-Resolution Surfel Maps	5-5-0	10-0-0	6-4-0	

(e) Results built from Whelan et al., (2015a) using absolute trajectory error (ATE) root-mean-square error (RMSE)

	ElasticFusion	ElasticFusion (no deformation)	Kintinuous	Multi-Resolution Surfel Maps	RGB-D SLAM	DVO SLAM
ElasticFusion		0.236	0.355	0.053	0.358	0.356
ElasticFusion (no deformation)	7-0-1		1.506	0.224	1.517	1.509
Kintinuous	6-2-0	5-3-0		0.148	1.007	1.002
Multi-Resolution Surfel Maps	8-0-0	7-1-0	7-1-0		6.786	6.750
RGB-D SLAM	6-2-0	3-5-0	4-4-0	0-8-0		0.995
DVO SLAM	8-0-0	4-4-0	4-4-0	1-7-0	6-2-0	

a reference algorithm in the literature since most of the papers evaluated their approaches against it. Here, this has the great advantage that if we assume that its performance is the same across the papers, we are able to compare the

algorithms across this boundary. Stückler and Behnke, (2014), the inventors of the Multi-Resolution Surfel Maps, compared it to the Generalized ICP, the Photometric ICP, fofis and D2D 3D-NDT on which it is based on. It is

also not very surprising that Multi-Resolution Surfel Maps outperform its competitors quite significantly in the number of wins and also in relative accuracy. The main reason for this is that none of the other algorithms explicitly model a loop closure. So indeed the performance is better but the algorithm is not a pure registration algorithm anymore. We can also see that the Photometric ICP achieves very good results and is most of the time better than the other algorithms. This is again due to its design, it is the only one except foveis and Multi-Resolution Surfel Mapsthat uses color information and is therefore better than the non-color registration algorithms.

If we now compare Multi-Resolution Surfel Maps against other algorithms implementing a loop closure system, we see that its dominance is gone. A bit surprising are the results from Whelan et al., (2015a). Here, the approach of Stückler and Behnke, (2014) totally fails against all other algorithms by at least one order of magnitude. This is simply caused by two of the eight tests where Multi-Resolution Surfel Maps probably computed totally wrong poses resulting in a very high error. But even if we only count the other six tests, the results are still not very good. On the other hand, DVO SLAM seems to be very robust and is either on par with or better than the other approaches. Its main advantage is the usage of both color and depth data combined in a robust Maximum-A-Posteriori formulation. However Kintinuous, which also used all the available data, is accurate enough to be on par with it. Here, we see another interesting effect. Even though the Maximum-A-Posteriori model is much more general and allows to weight each Jacobian and residual to be weighted individually, this advantage can somehow be compensated if a fused model that is much smoother and contain less

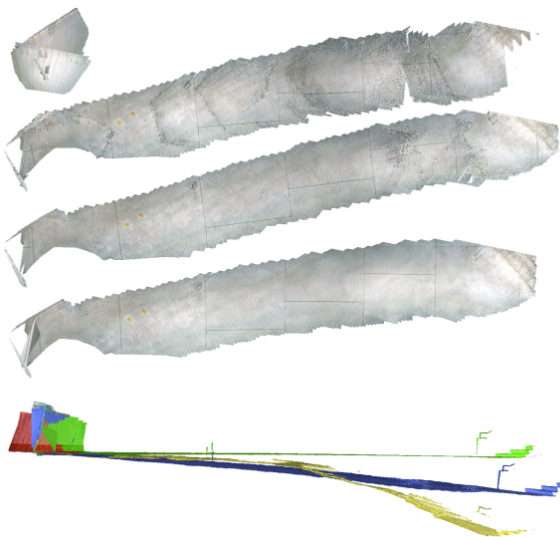


Figure 9: Reconstruction results of another data set, taken from Whelan et al., (2013). From top to bottom and in colors red, yellow, blue, green: KinectFusion ICP, Photometric ICP, foveis, Combined ICP.

noise is used. While in KinectFusion, this seems to have not a great impact, Kintinuous uses the advanced Combined ICP and also models loop closures by deforming the reconstruction. In that way, its performance is quite well. However, further tests by Whelan et al., (2015b) showed that DVO SLAM is in fact better and achieves better results. Nevertheless, it is interesting to see how a fused model helps to find better poses.

Across all papers, KinectFusion ICP performs very bad and is outperformed by all other algorithms. This is not very surprising because most of its competitors have models that are more complex and use the data in a better way. The sparse methods RGB-D SLAM and foveis also achieve very good results and especially foveis is more accurate than Generalized ICP, D2D 3D-NDT and the Photometric ICP. ElasticFusion, probably the most advanced approach, beats every other algorithm quite significantly and even with disabled loop closure mechanism its performance is still very good. This is in fact very surprising. Disabling loop closure detection removes the possibility to correct drift that was accumulated during registration. Only if the estimation of the camera pose is very accurate, loop closures get less important. But in our scenario, the noise produced by depth sensors is really problematic and thus we don not expected such impressive results. The hybrid model that is somehow in-between the classic volumetric approach and the probabilistic NDT model seems to perform very well and combine the strengths of both worlds.

7.1 Conclusion

Summarizing all results we can conclude that the most powerful pure registration algorithm for our pipeline is the Combined ICP approach. It is fast, accurate and achieve very good results. If we consider all approaches, ElasticFusion achieves the overall best performance and is still good even if deformations are disabled. Its model also seem to be better and more flexible than ours.

The main drawback of our pipeline is the dependence on static scenes which makes it inflexible against deformations. This includes the mentioned loop closures and also dynamic scenes. Recently, Newcombe et al., (2015) extended KinectFusion in this way to reconstruct moving and deformable objects. ElasticFusion is also an example of a more flexible system. All in all, we have discovered interesting registration and reconstruction algorithms and we hope that the report will help researchers to find new ways to solve this hard problem.

References

- [1] P. J. Besl and N. D. McKay. “A Method for Registration of 3-D Shapes”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14.2 (1992), pp. 239–256.

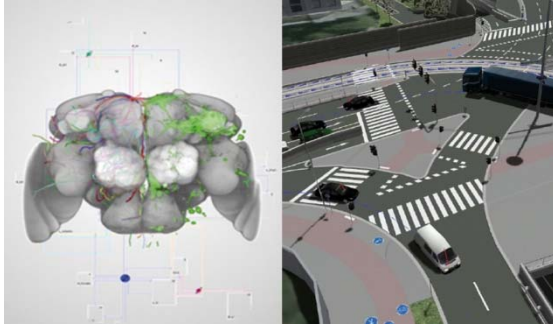
- [2] P. Biber and W. Straßer. “The normal distributions transform: A new approach to laser scan matching”. In: *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. Vol. 3. IEEE. 2003, pp. 2743–2748.
- [3] G. Blais and M. D. Levine. “Registering multi-view range data to create 3D computer objects”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17.8 (1995), pp. 820–824.
- [4] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. “Real-time camera tracking and 3d reconstruction using signed distance functions”. In: *Robotics: Science and Systems (RSS) Conference 2013*. Vol. 9. 2013.
- [5] D. R. Canelhas, T. Stoyanov, and A. J. Lilienthal. “SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 3671–3676.
- [6] J. Chen, D. Bautembach, and S. Izadi. “Scalable Real-time Volumetric Surface Reconstruction”. In: *ACM Trans. Graph.* 32 (2013), 113:1–113:16.
- [7] Y. Chen and G. Medioni. “Object Modelling by Registration of Multiple Range Images”. In: *Image Vision Computing (IVC)* 10.3 (1992), pp. 145–155.
- [8] B. Curless and M. Levoy. “A Volumetric Method for Building Complex Models from Range Images”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '96*. 1996, pp. 303–312.
- [9] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. “An evaluation of the RGB-D SLAM system”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 1691–1696.
- [10] G. Gallego and A. Yezzi. “A compact formula for the derivative of a 3-D rotation in exponential coordinates”. In: *Journal of Mathematical Imaging and Vision* 51.3 (2015), pp. 378–384.
- [11] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. “RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments”. In: *The International Journal of Robotics Research* 31.5 (2012), pp. 647–663.
- [12] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. “Visual odometry and mapping for autonomous flight using an RGB-D camera”. In: *International Symposium on Robotics Research (ISRR)*. 2011, pp. 1–16.
- [13] B. Huhle, M. Magnusson, W. Straßer, and A. J. Lilienthal. “Registration of colored 3d point clouds with a kernel-based extension to the normal distributions transform”. In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, pp. 4025–4030.
- [14] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. “KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. ACM, 2011, pp. 559–568.
- [15] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. Torr, and D. Murray. “Very high frame rate volumetric integration of depth images on mobile devices”. In: *Visualization and Computer Graphics, IEEE Transactions on* 21.11 (2015), pp. 1241–1250.
- [16] C. Kerl, J. Sturm, and D. Cremers. “Dense visual slam for rgb-d cameras”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 2100–2106.
- [17] C. Kerl, J. Sturm, and D. Cremers. “Robust odometry estimation for RGB-D cameras”. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3748–3754.
- [18] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. “g2o: A general framework for graph optimization”. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 3607–3613.
- [19] S. Laine and T. Karras. “Efficient Sparse Voxel Octrees”. In: *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10. 2010, pp. 55–63.
- [20] K.-L. Low. *Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration*. Tech. rep. Chapel Hill, University of North Carolina, 2004.
- [21] M. Magnusson, A. Lilienthal, and T. Duckett. “Scan registration for autonomous mining vehicles using 3D-NDT”. In: *Journal of Field Robotics* 24.10 (2007), pp. 803–827.
- [22] R. A. Newcombe, D. Fox, and S. M. Seitz. “DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 343–352.
- [23] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. “KinectFusion: Real-Time Dense Surface Mapping and Tracking”. In: *IEEE ISMAR*. IEEE, 2011.

- [24] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. “Real-time 3D Reconstruction at Scale Using Voxel Hashing”. In: *ACM Trans. Graph.* 32.6 (2013), 169:1–169:11.
- [25] H. Roth and M. Vona. “Moving Volume KinectFusion”. In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2012, pp. 112.1–112.11.
- [26] R. B. Rusu, N. Blodow, and M. Beetz. “Fast point feature histograms (FPFH) for 3D registration”. In: *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*. IEEE. 2009, pp. 3212–3217.
- [27] A. Segal, D. Haehnel, and S. Thrun. “Generalized-ICP.” In: *Robotics: Science and Systems*. Vol. 2. 4. 2009.
- [28] F. Steinbrücker, J. Sturm, and D. Cremers. “Real-time visual odometry from dense RGB-D images”. In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE. 2011, pp. 719–722.
- [29] P. Stotko and T. Golla. “Improved 3D Reconstruction using Combined Weighting Strategies”. In: *Proceedings of CESC G 2015: The 19th Central European Seminar on Computer Graphics*. 2015, pp. 135–142.
- [30] T. D. Stoyanov, M. Magnusson, H. Andreasson, and A. Lilienthal. “Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations”. In: *The International Journal of Robotics Research* (2012), p. 0278364912460895.
- [31] J. Stückler and S. Behnke. “Multi-resolution surfel maps for efficient dense 3D modeling and tracking”. In: *Journal of Visual Communication and Image Representation* 25.1 (2014), pp. 137–147.
- [32] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 573–580.
- [33] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. “Kintinuous: Spatially Extended KinectFusion”. In: *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*. Sydney, Australia, 2012.
- [34] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. “ElasticFusion: Dense SLAM without a pose graph”. In: *Proceedings of Robotics: Science and Systems (RSS)*. 2015.
- [35] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. “Real-time large-scale dense RGB-D SLAM with volumetric fusion”. In: *The International Journal of Robotics Research* 34.4-5 (2015), pp. 598–626.
- [36] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald. “Robust real-time visual odometry for dense RGB-D mapping”. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 5724–5731.
- [37] C. Wu. “SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)”. In: (2007). URL: <http://cs.unc.edu/~ccwu/siftgpu/>.
- [38] M. Zeng, F. Zhao, J. Zheng, and X. Liu. “A Memory-Efficient KinectFusion Using Octree”. In: *Computational Visual Media*. Springer, 2012, pp. 234–241.
- [39] M. Zeng, F. Zhao, J. Zheng, and X. Liu. “Octree-based Fusion for Realtime 3D Reconstruction”. In: *Graph. Models* 75.3 (2013), pp. 126–136.

Sponsors of CESC 2016



zentrum für
virtual reality und visualisierung
forschungs-gmbh



VRVis Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH

The VRVis Research Center is a joint venture in research and development for virtual reality and visualization. VRVis was founded in 2000 as part of the Austrian Kplus program to bridge the gap between academic research and commercial development as well as to supply the necessary transfer of knowledge between the academic community and industry. The competence center VRVis is funded by BMVIT, BMWFW, and the Vienna Business Agency within the scope of COMET – Competence Centers for Excellent Technologies. The program COMET is managed by FFG.

This mission is mirrored in a variety of academic and industrial partners. The research center is currently conducted by five academic institutes and numerous industrial partners. Leading-edge innovations and down-to-earth business style characterizes VRVis as a valued partner for high-level research.

The company is located in Vienna, Austria. Today, around 60 researchers together with about 20 students do high-level applied and basic research in three different areas.

The Team

VRVis consists of internationally experienced researchers in the areas of visualization, rendering and visual analysis. Their outstanding experience and knowledge in these topics qualify them for the innovative research they are performing. The research areas are headed by key researchers who manage these areas, define goals and projects for this area, and conduct the defined research together with their staff. Most members of the research teams are young researchers, whose creativity and ingenuity is the key to the success. VRVis is always looking for young, talented, and motivated researches in the fields of research to extend its research work or to support partner companies.

Research Program

The scientific research program consists of three research areas (Visualization, Rendering and Visual Analysis) in which thematically matching research projects are conducted. Each research area realizes application projects on the one hand and basic research for these application projects on the other hand.

Working at VRVis

VRVis is always looking for students, junior and senior researchers who want to join the VRVis team. VRVis is offering internships, diploma theses and PhD theses in cooperation with the TU Wien and regular positions. For more information or search for job opportunities in the field of Visual Computing visit our webpage at www.vrvis.at.

Selection of Partners

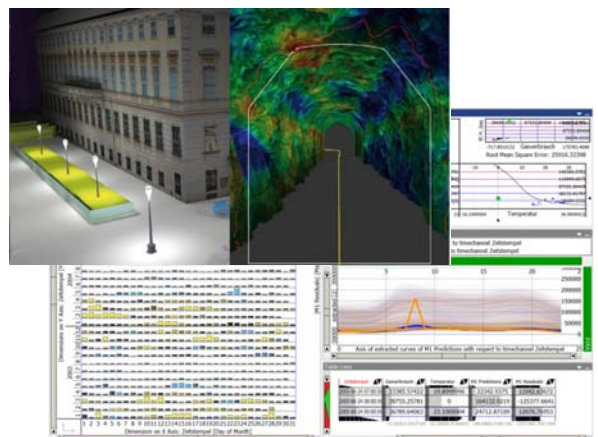
Scientific Partners:

- Vienna University of Technology
- Graz University of Technology
- University of Vienna

Industrial Partners:

- AVL List GmbH
- AGFA Healthcare GesmbH
- Austria Power Grid AG
- Geodata Ziviltechniker GmbH
- Imagination Computer Services GesmbH
- ÖBB-Infrastruktur AG
- Zumtobel Lighting GmbH
- and many more

Currently, VRVis is again extending its industrial base with new partners from several new fields.



Additional Information and Contact

Please visit our webpage for detailed information about the research program or current projects at www.vrvis.at or contact us at office@vrvis.at or via phone +43 (1) 20501 / 30100.

