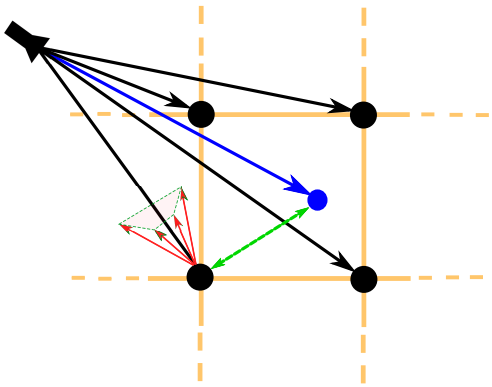
SIGRAD

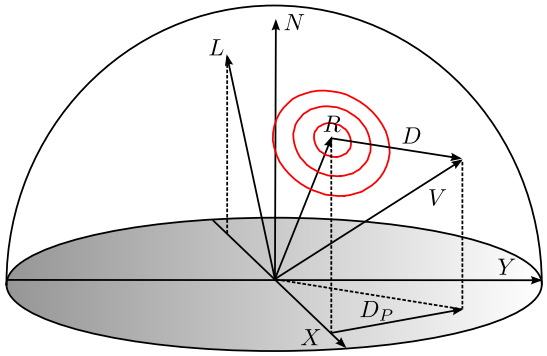
SIGRAD 2013 @ Norrköping

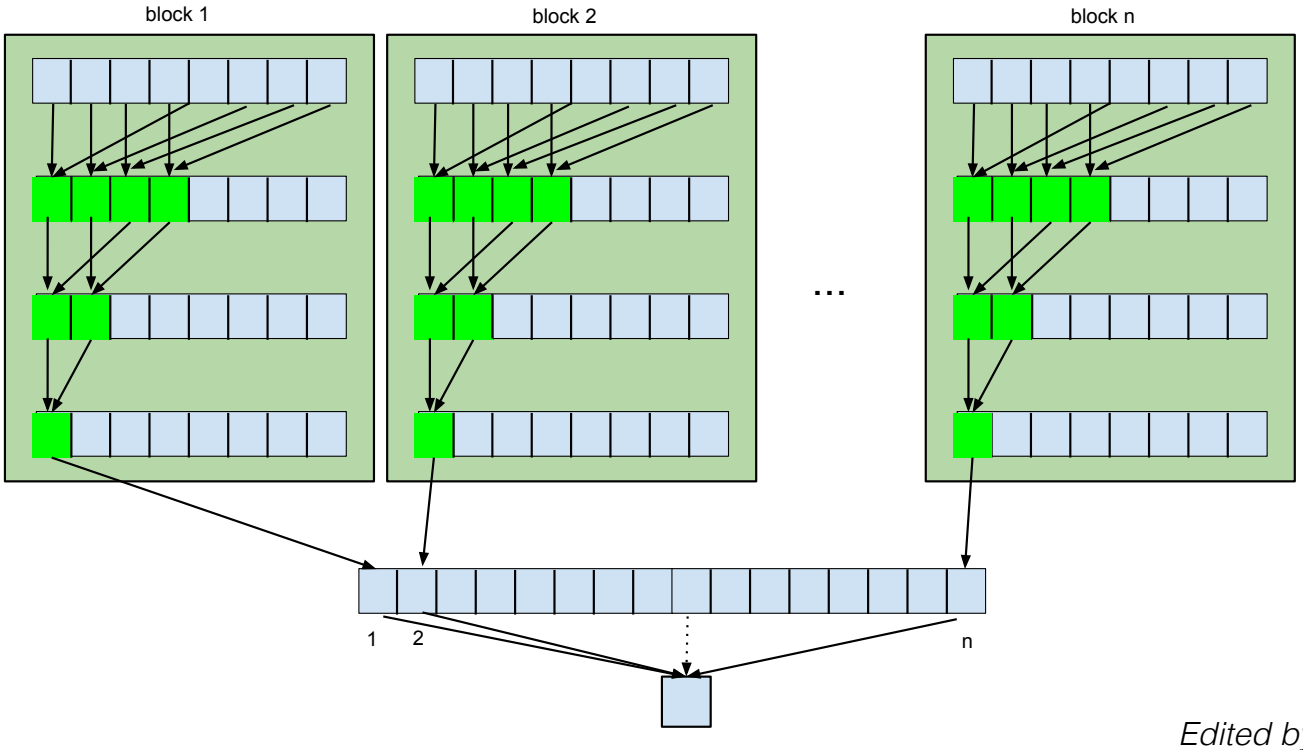
CRESEARCH

SIGRAD 2013

Visual computing







Edited by
Timo Ropinski & Jonas Unger

Proceedings of SIGRAD 2013

Visual Computing

June 13-14, 2013

Norrköping, Sweden

Edited by
Jonas Unger and Timo Ropinski

Linköping Electronic Conference Proceedings, No. 94
Linköping University Electronic Press
Linköping, Sweden, 2013
ISBN: 978-91-7519-455-4
ISSN: 1650-3686 (print)
ISSN: 1650-3740 (online)
http://www.ep.liu.se/ecp_home/index.en.aspx?issue=094

SIGRAD 2013

We are happy to announce the 12th SIGRAD Conference Proceedings. SIGRAD 2013 will be held in Norrköping, Sweden, on June 13 and 14, 2013. SIGRAD 2013 focuses on visual computing, and solicits the submission of original research papers that advance the state-of-the-art of one of the subareas of visual computing, ranging from computer graphics and visualization to human-computer-interaction.

SIGRAD 2013 is the premier Nordic forum for computer graphics and visualization advances for academia, and industry. This annual event brings together researchers and practitioners with interest in techniques, tools, and technology from various fields such as computer graphics,visualization, visual analytics, or human-computer interaction. Each paper in this conference proceedings was peerreviewed by at least three reviewers from the international program committee consisting of 26 experts listed below. Based on this set of reviews, the conference co-chairs accepted 9 papers in total and compiled the final program.

Jonas Unger and Timo Ropinski

International Program Committee

Tomas Akenine Möller (Lund University, Sweden)
Ulf Assarsson (Chalmers University of Technology, Sweden)
Francesco Banterle (ISTI-CNR, Italy)
Alan Chalmers (Warwick University, UK)
Stefan Bruckner (TU Vienna, Austria)
Gerd Bruder (University of Würzburg, Germany)
Matt Cooper (Linköping University, Sweden)
Michael Doggett (Lund University, Sweden)
Thomas Ertl (University of Stuttgart, Germany)
Morten Fjeld (Chalmers University, Sweden)
Eduard Gröller (TU Vienna, Austria)
Diego Gutierrez (Universidad de Zaragoza, Spain)
Andreas Kerren (Linnaeus University, Sweden)
Lars Kjeldahl (KTH Stockholm, Sweden)
Tomas Larsson (Mälardalen University, Sweden)
Rafal Mantiuk (Bangor University, UK)
Tania Pouli (Max Planck Institut für Informatik, Germany)
Bernhard Preim (University of Magdeburg, Germany)
Stefan Seipel (University Gävle, Sweden)
Heidrun Schumann (University of Rostock, Germany)
Veronica Sundstedt (Blekinge Institute of Technology, Sweden)
Ivan Viola (TU Vienna, Austria)
Ruediger Westermann (TU Munich, Germany)
Thomas Wischgoll (Wright State University, US)
Burkhard Wuensche (University of Auckland, New Zealand)
Anders Ynnerman (Linköping University, Sweden)

Program

Thursday, June 13th

09:30-10:00 Registration

10:00-11:00 Paper Session: Visualization Applications

Peter Hoffmann, Markus Höferlin, Andreas Kirstädter, and Daniel Weiskopf
Dynamic Evacuation Planning by Visual Analytics—An Expert Survey

Khoa Tan Nguyen, Timo Ropinski
Feature Tracking in Time-Varying Volumetric Data through Scale Invariant Feature Transform

Alexander Steen, Marcus Widegren
3D Visualization of Pre-operative Planning for Orthopedic Surgery

11:00-11:15 Coffee Break

11:15-12:15 Keynote Thomas B Schön (Linköping University, Sweden)

12:15-13:30 Lunch (C-Cafe)

13:30-14:30 Industrial Keynote Martin Enthed (Ikea R&D, Sweden)

14:30-14:45 Coffee Break

14:45-15:45 Paper Session: Shading and Illumination

John David Olovsson, Michael Doggett
Octree Light Propagation Volumes

Neda Rostamzadeh, Daniel Jönsson and Timo Ropinski
Comparison of Volumetric Illumination Models

Stefan Gustavson
No more Texels, No more Facets: Emerging Trends in GPU Procedural Shading

16:00-17:00 Exhibition & Dome Show

18:00 Conference Dinner (Restaurant Laxholmen)

Friday, June 14th

9:00-10:00 Industrial Keynote Claes Lundström (Sectra AB, Sweden)

10:15-11:15 Paper Session: Volume Sampling and Segmentation

Stefan Lindholm, Alexander Bock
Poor Man's Rendering Of Segmented Data

Stefan Lindholm, Daniel Jönsson, Hans Knutsson, Anders Ynnerman
Towards Data Centric Sampling for Volume Rendering

Elhassan Abdou
Visual Planning and Verification of Deep Brain Stimulation Interventions

11:15-11:30 Coffee Break

11:30-12:30 Keynote Frank Steinicke (University of Wuerzburg, Germany)

12:30-12:45 Closing Remarks and Awards Ceremony

12:45 Lunch and Goodbye

Conference Sponsors

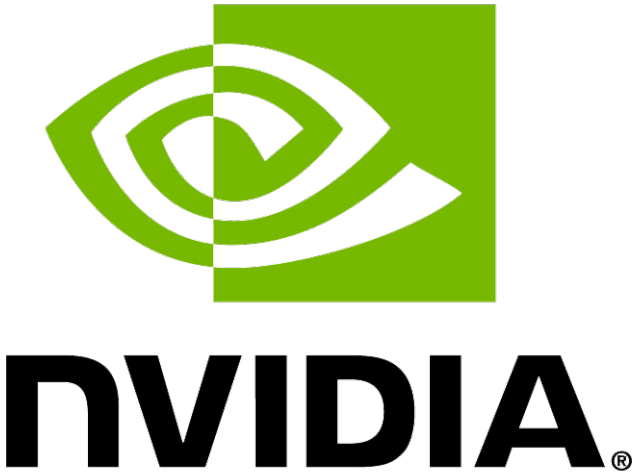


Table of Contents

Visualization Applications

SIGRAD 2013, pp. 3-10
T. Ropinski and J. Unger (Editors)

Dynamic Evacuation Planning by Visual Analytics—An Expert Survey

Peter Hoffmann[1,2] , Markus Höferlin[1], Andreas Kirstädter[2], and Daniel Weiskopf[1]

[1]Visualization Research Center, University of Stuttgart (VISUS), Germany
[2]Institute for Communication Networks and Computer Systems, University of Stuttgart (IKR), Germany

Abstract

In a formative user study, we evaluated the requirements of visual analytics for dynamic evacuation planning. To this end, we created a prototype that implements the visual analytics methodology and is able to compute, visualize, and interact with evacuation routes in emergency situations in buildings. Using this prototype, we conducted an expert survey including a psychologist, a consultant for building safety measures, and a building planner. The survey provides essential information needed to build a tool that is able to evacuate people inside a building safely. Additionally, we report on results of the survey regarding technical limitations in obtaining data such as the evacuees' position and their number to calculate the shortest routes during evacuation.

Categories and Subject Descriptors (according to ACM CCS):
F.2.2 [Nonnumerical Algorithms and Problems]: Sequencing and scheduling—Routing and layout
H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces (GUI)
K.4.1 [Computers and Society]: Public Policy Issues—Human safety

1. Introduction

In the last years, many tragedies occurred due to failed evacuations in emergency situations. Incidents at festivals, such as the Love Parade 2010[1] and the Madrid arena tragedy[2], as well as working places, such as the fires at a German workshop[3] and a Bangladesh textile factory[4] demonstrated that it can be disastrous if evacuations are not prepared thoroughly. Therefore, public spaces or buildings have to be planned carefully to create safe evacuation routes and avoid bottlenecks at all cost. Afterwards, the evacuation has to be supervised to lead the evacuees to safety and create secure routes without generating bottlenecks and avoiding blocked areas due to fire or smoke spreading.

For building planning as well as for evacuation supervision, a tool is needed to assist the users with visualization of the building and follow the progress of the evacuation. Interactivity is crucial to allow building planners to run multiple simulations with different evacuation scenarios or to intervene live evacuations if required.

To create such a tool, the knowledge and experience of experts in related fields needs to be incorporated. Psychologists provide knowledge about the human behavior in panic situations and can predict the actions of evacuees during evacuation. Their knowledge is also important to create good user interfaces that are intuitive and easy to use. Building planners and consultants know the guidelines that have to be taken into account for evacuation. They are also aware of the technical possibilities and familiar with the issues that arise when planning an evacuation. Therefore, the experts' knowledge is fundamental to create a tool that adheres to all requirements necessary to fulfill the legal regulations and to ensure safe evacuation.

[1] http://www.spiegel.de/international/germany/analysis-of-the-love-parade-tragedy-the-facts-behind-the-duisburg-disaster-a-708876.html
[2] http://elpais.com/elpais/2012/11/29/inenglish/1354205787_673115.html
[3] http://www.independent.co.uk/news/world/europe/fourteen-disabled-people-die-in-fire-at-german-workshop-8352798.html
[4] http://www.guardian.co.uk/world/2012/nov/25/bangladesh-textile-factory-fire

In this paper, a visual analytics tool is developed for building planning and evacuation supervision with the purpose to conduct a user study including a psychologist, a consultant for building safety measures, and a building planner. This survey provides important insights related to evacuation scenarios and will be relevant to future development of reliable tools that support evacuation.

2. Previous Work

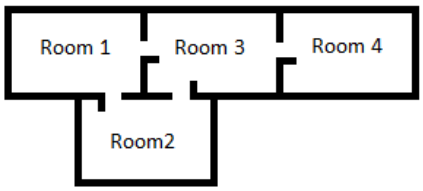
To create a tool for building evacuation, several steps are necessary. First, an algorithm is required to find the shortest paths for the evacuees to reach the exits. Then, we need a visualization to monitor the evacuation and detect critical situations. Interactivity is important for users to be able to switch between different visualizations and modify parameters to detect problems before they occur and solve them. This section discusses the related approaches of these topics.

2.1. Shortest Path Finding Algorithms

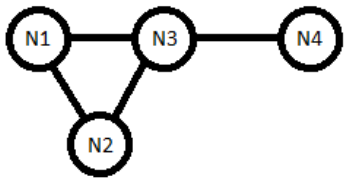
To find the shortest paths in an evacuation scenario, an algorithm is required that calculates the fastest routes from all occupied rooms to the exits. There are several algorithms that can be used for this purpose. Most of them first divide the structure of the building into nodes and edges and then calculate the fastest routes. A small example is shown in Figure 1. For instance, the *Capacity Constrained Route Planning* (CCRP) algorithm [LGS05] considers the capacities for each node and each edge whilst finding the shortest paths. There are also many advances to this approach, such as *Intelligent Load Reduction* [KGS07], which neglects bottlenecks found in previous iterations, or the *Incremental Data Structure* [KGS07], which reuses already calculated information to reduce the computation time. CCRP++ reduces unnecessary load in computation [Yin09]. There are also other approaches based on CCRP such as the *Hierarchical Route-Planning Algorithm* [YLJ*11], or the *Priority based Distributed Evacuation Routing* [RHDW12], which include priorities in their calculations. Since the purpose of the tool is the usage in the formative user study, the standard CCRP algorithm is adequate and, hence, used to calculate the shortest routes in the developed prototype.

2.2. Visualization Approaches for Building Evacuation

Two software tools that use visualization approaches for building evacuation are SIMULEX[5] and buildingEXODUS[6]. Both tools create simulations of building evacuations. For visualization, they calculate the paths for the evacuees to the nearest exits, first. To this end, they divide the building into a grid where the evacuees move from one square to the next. Individual evacuee parameters, such as body shape and size, walking speed, and time to respond to an alarm can be defined. The simulation starts after parameter definition. Additionally, parameters, such as walking speed, can be reduced when evacuees enter an area filled with smoke or fire.



(a) Floor plan.



(b) Nodes and edges representation.

Figure 1: The floor plan shown in (a) includes 4 rooms and is divided accordingly into 4 nodes (b) connected through edges, which represent the doors between the rooms.

After parameter definition, the evacuees go to the nearest exit. The evacuation is then presented as 2D or 3D visualization. The parameters can only be modified until the visualization starts, but specific data, such as general occupant flow or the individual movements, can be extracted after the simulation has finished. This allows the users to detect bottlenecks or other dangerous situations occurring during the evacuation.

2.3. Evacuation Utilizing Visual Analytics

Visual analytics is used to gain insight in complex scenarios by combining automatic processing, visualization, and human-machine interaction to take advantage of the strengths of both human and machine. In the context of this paper, a loop is used where an algorithm calculates the shortest paths and the visualization of the computed paths are presented to the users (Figure 2). Parameters can be modified to change the routes ensuring a safe evacuation. After adjusting the parameters, the background algorithm recalculates the data and shows the results in the visualization, again. This approach has been mainly applied to the evacuation of

[5] http://www.iesve.com/software/ve-pro/analysis-tools/egress/simulex
[6] http://www.fseg.gre.ac.uk/exodus/

cities or large areas using vehicles to create the scheduling for the evacuations (e.g., [AAB08b], [AAB08a]).

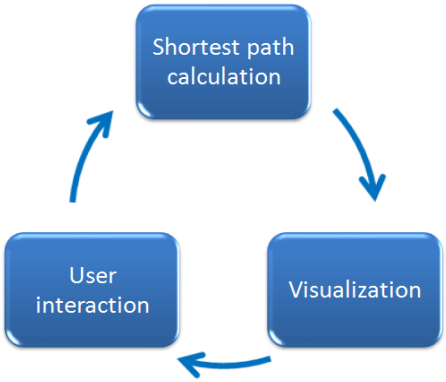


Figure 2: Visual analytics loop for dynamic evacuation planning. First, the shortest paths are calculated. A visualization communicates relevant information to the users, and the users can modify evacuation parameters and interact with the visualization. When parameters are modified, the loop is started from the beginning.

In building evacuation, a recent example of applying visual analytics is described by Reddy et al. [RHDW12]. The shortest paths to the exits are calculated and a visualization shows the building divided into a set of nodes connected by edges that represent the different rooms and their links. Each node includes a graph showing the occupancy rates inside the building during the whole evacuation. In our approach, we create a prototype based on this idea to receive feedback from experts related to building evacuation. This enables us to extract useful information to create a qualified tool for building evacuation in the future.

3. Evacuation Tool Prototype

Our visual analytics system creates an interactive loop that calculates the shortest routes first and then visualizes the information on screen. The user is then able to interact with the visualization: modifications lead to the recalculation of the shortest paths and the visualization is updated (see Figure 2). This offers the possibility to filter and extract data regarding the building evacuation, while the simulation is running. This represents the main novelty compared to previous approaches (Sections 2.2 and 2.3), where data can only be extracted after a simulation has been completed. These approaches can only be used for building planning or to analyze an evacuation after it took place. With our tool an evacuation can also be followed while it takes place to help the evacuees get out of the building safely.

3.1. Path Finding Algorithm

The CCRP algorithm is applied to calculate the shortest paths considering the capacities inside the building. Dijkstra's algorithm is used to determine the shortest routes from the nodes sheltering evacuees to the exits. Afterwards, CCRP reserves the capacities throughout the calculated escape routes to ensure that the evacuees will not be blocked once the evacuation has started. Using this process, bottlenecks are avoided in advance. Several routes can be taken to the exits instead of only one static evacuation route for all people inside the building. The algorithm runs in the background to recalculate the paths if modifications are made in the visualization.

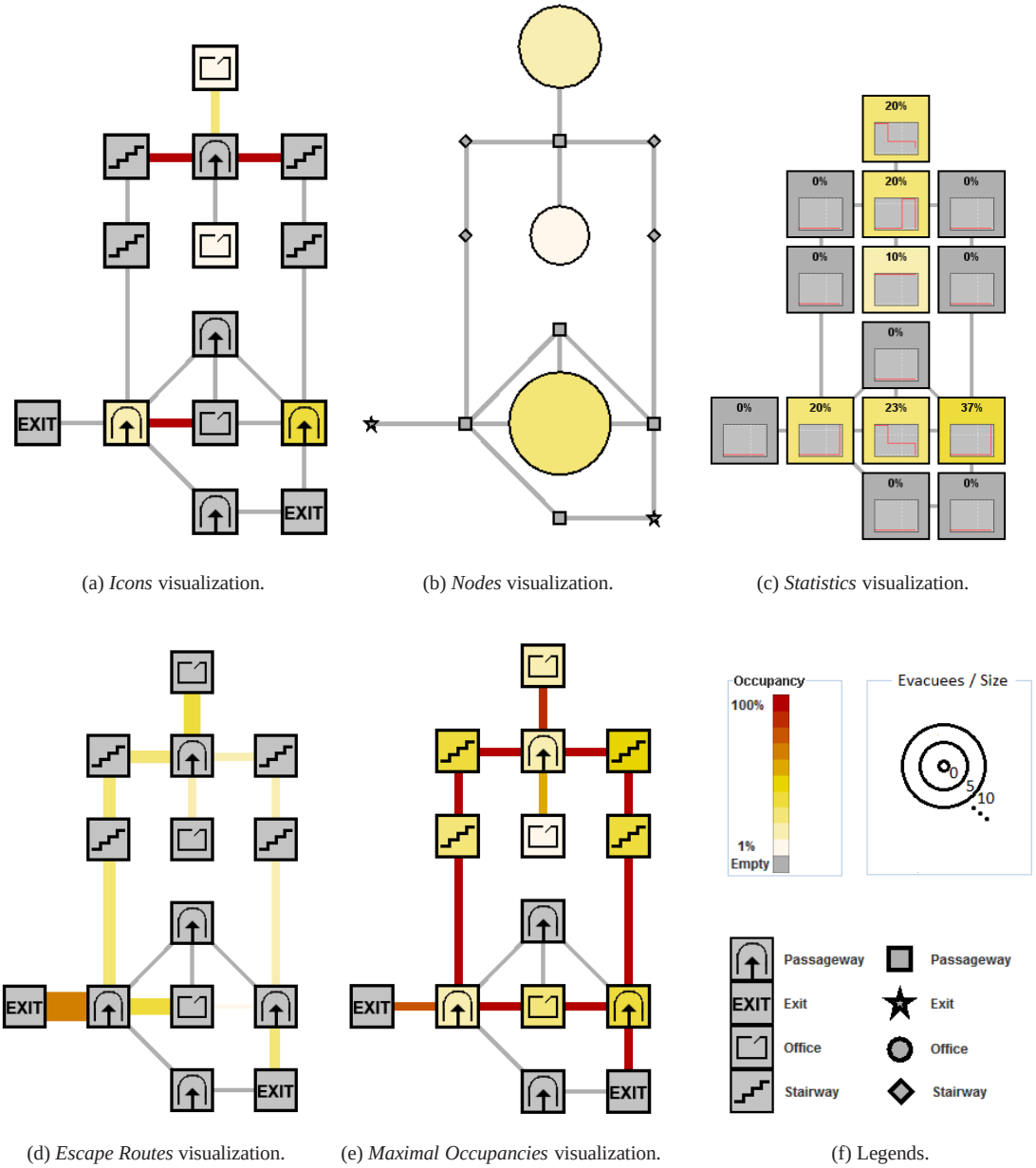
3.2. Visualization

The visualization includes several types of coordinated views (*Icons*, *Nodes*, *Statistics*, *Escape Routes*, and *Maximal Occupancies*) that display different kinds of information needed for building planners or in live evacuations (see Figure 3). The *Icons* visualization shows the rooms in a building represented by nodes that are connected with edges wherever there is a door in the building (see Figure 3 (a)). A different icon is used for each type of room (office, stairway, passageway, and exit) and colors are used to display the occupancy level of each room and edge. The movement of the evacuees inside the building is represented by the changing colors of the different items (nodes/edges). Rooms getting filled tend to have a reddish color while rooms where evacuees are leaving get white. Grey color is used for empty items.

In the *Nodes* visualization, simple shapes such as circles or squares are used to represent the nodes (see Figure 3 (b)). In this case, the size of the nodes changes depending on the number of evacuees inside. This visualization provides an overview of the number of evacuees in each node whereas the node colors still represent the occupancy level of the items.

The *Statistics* visualization is based on the approach used in [RHDW12] (see Figure 3 (c)). Each node is represented by a graph that shows the number of evacuees that have passed through this node until the current time instant. Using this visualization, each node can be analyzed independently looking at each graph. The color coding used here depends on the maximal occupancy level of a node from the beginning of the evacuation until the current point in time. The exact value is also displayed above every graph.

The escape routes that were taken during the evacuation are displayed in the *Escape Routes* visualization. Here, the size and colors of the edges change both according to the number of evacuees that passed through them in relation to the total number of evacuees in the building at the starting point (see Figure 3 (d)). Hence, the more evacuees crossed an edge during the whole evacuation, the bigger its size and darker its color. Using this visualization the overall paths



(a) *Icons* visualization. (b) *Nodes* visualization. (c) *Statistics* visualization.

(d) *Escape Routes* visualization. (e) *Maximal Occupancies* visualization. (f) Legends.

Figure 3: Visualization types of the prototype for evacuation planning. The *Icons* visualization (a) includes figures representing the nodes that show the room type (office, passageway, stairway, or exit). The colors are linked to the occupancy level of each node/edge. The *Nodes* visualization (b) uses simple shapes, such as circles or squares, for the nodes. The size of the nodes indicates the number of evacuees. The larger the node, the more evacuees inside. In the *Statistics* visualization (c), each node is represented by a small graph that shows how many evacuees are inside from the starting point until the current time instant. The percentage as well as the background color of each node represents the maximal value reached inside a graph. In the *Escape Routes* visualization (d), information aggregated through time is represented. The size as well as the color of an edge show how many evacuees went through this path with respect to the total count of evacuees. The thicker an edge, the more evacuees crossed it during the whole evacuation. In the *Maximal Occupancies* visualization (e), the color of each item (node/edge) shows the maximal occupancy level of this item. The more a color tends to red, the more evacuees have been inside simultaneously in one time instant. Figure (f) shows the legends related to the colors, the sizes of the nodes in (b), and the icons used.

taken by the evacuees can be followed and analyzed with the purpose of optimizing the number of paths in a building and their capacities.

Finally, the *Maximal Occupancies* visualization shows the maximal occupancy level of each item inside the building throughout the whole evacuation (see Figure 3 (e)). This allows the users to see which nodes or edges reached their limit and detect bottlenecks inside the building swiftly. The time each node or edge reached its maximal occupancy level can also be reviewed in this visualization.

3.3. Interaction

It is important to enable the users to modify the parameters of the visualization dynamically for building planning as well as for live evacuations. Therefore, the tool includes several interaction possibilities to modify the simulations, which can be used to test different simulations to create safe escape routes in a building. Different evacuation scenarios have to be examined to rule out any problems that can emerge. In the following, the included interaction possibilities are explained.

First, users can obtain detailed information of items by clicking. The other options can be found in the user interface depicted in Figure 4: a timeline allows the users to move backward or forward in time showing the evacuation situation at each time step. The different evacuation routes of the evacuees can be highlighted to see which paths the people are currently following. Finally, users can modify different parameters of the simulation, such as the evacuee number, the capacity, and the travel times of items. Nodes and edges can also be added or deleted to create new evacuation routes for the evacuees. The tool visualizes the modified evacuation routes and new situations directly.

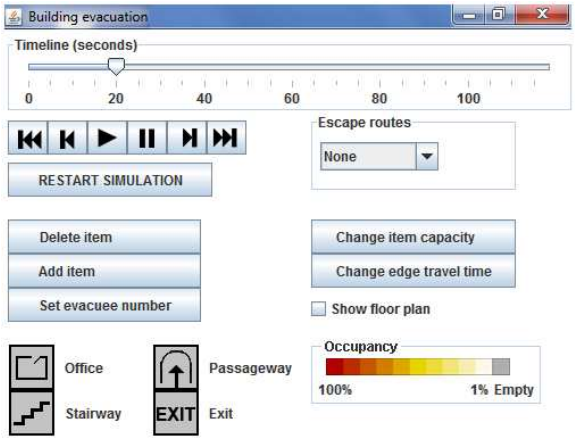


Figure 4: Dialog for user interaction. The timeline can be modified with the buttons below to show other time steps. The calculated escape routes can be highlighted using the *Escape routes* drop-down list. The buttons in the middle are used to change different parameters of the evacuation, such as deleting or adding an item, changing the number of evacuees in one node, changing the maximal capacity of an item, or changing the travel time of an edge. The building plan can be visualized by the checkbox on the lower right. The legends of the visual mappings for each visualization are shown on the lower part of the screen.

4. Expert Survey

The discussed prototype was developed to collect user feedback for improvements of the system in a formative process and to guide the design of future applications. Some examples demonstrating the performance of the tool in different situations were shown to experts (a psychologist, a consultant for building safety measures, and a building planner) to obtain their opinion about the tool and additional aspects that have to be considered further. Moreover, technical limitations that may cause problems during implementation of such tools in real evacuation scenarios were discussed. In the following, the conclusions of this surveys are presented. As every expert provided information related to different fields (psychological aspects, technical limitations, etc.) we decided to divide the areas in which the feedback was gathered into four main groups: conceptual issues, psychological aspects, important features, and technical limitations. Therefore, general ideas and concepts will be explained in the first two groups, and then the more specific and technical problems discussed in the surveys will be outlined in the last two subsections.

4.1. Participants

The participants were chosen from different fields of research, related to building evacuation. The psychologist as well as the consultant for building safety measures work in an international company that is involved in different fields of research including building safety planning. The building planner works in his own company and has different projects regarding the planning and implementation of buildings.

The psychologist works in a group related to human-machine interaction. His task was to analyze the interface and the building representations of the tool to improve the overall visualization in further versions. The consultant for building safety measures was invited to analyze the feasibility of the features included in the tool, as well as the problems concerning the technology available to get data necessary for the building evacuation (evacuee count, position, etc.). The building planner's task was to analyze the features as a potential target user and give recommendations regarding the features of the tool.

4.2. Study Procedure

The prototype features were presented to the experts independently, simulating several evacuation scenarios including the different representations available. They did not interact with the tool themselves, but observed and stated their questions during the presentation. These questions led to several discussions related to the areas that will be presented in the next subsections. The length of each study sessions was about two hours inside private conference rooms. After the presentation, the experts gave their overall opinion about the tool. The results include the pending problems that will have to be solved in future versions.

4.3. Conceptual Issues

First, the purpose of visual analytics for evacuation planning was discussed with the experts. They insisted that the goals have to be clear to adapt the different visualizations and interactive options. In this case, two goals for an evacuation tool have to be distinguished: live evacuation, and building planning.

For live evacuation, users can intervene in evacuations when required. Blocked paths due to collapsed ceilings, fire, or smoke spreading can be dangerous when evacuating people; here the tool can help recalculate the shortest routes and lead the evacuees through safer paths. As important decisions need to be made in time critical situations, it is important to keep the visualizations as well as interaction as simple as possible. The experts also emphasized that users need to be able to recognize dangerous situations as well as bottlenecks swiftly to intervene in a fast and efficient way.

The objectives for building planners are different. They need a tool to plan buildings as safe as possible. Therefore, they need to do different simulations using several scenarios. Parameters, such as door widths, room properties, or travel times need to be changed to find optimal solutions, which requires a lot of simulations to compare different solutions.

Hence, according to the experts, the interaction possibilities should be adapted according to the purpose of the tool: live evacuation or building planning. The developed tool is a prototype including options suited for both scenarios. Consequently, the different visualizations and interactive options have to be planned thoroughly to concentrate only on one of these purposes. Ideally, two separate tools have to be developed to achieve both objectives.

4.4. Psychological Aspects

Psychological aspects have to be considered in an evacuation tool to understand the evacuees' behavior and predict their actions. The key issues that became apparent in the discussion are listed below:

- *Family members tend to stay together* instead of escaping the building as fast as possible, also see Kobes et al. [KHdVP10].
- *Evacuees cannot be stopped once the evacuation started*, even if bottlenecks were avoided this way. The only option is to start the alarm signal in each room at a different time according to the output of the developed evacuation tool.
- *Group building* is important in an evacuation. Evacuees can be grouped together, but they cannot be split afterwards. The splitting would take too much time and there would be nobody to decide on how to split the group.
- *Response time* after the alarm starts has to be considered. When an alarm sounds, the evacuees need a certain time to realize that the situation is real and that they have to leave the building. This response time has to be included in the calculation of the evacuation schedule. In [Hos09], the effects and reactions influencing the so-called premovement time are summarized. Mainly the time to perceive the alarm, the interpretation time of the perceived alarm, and the time used for actions that are not directly related to the evacuation itself have to be considered here.

4.5. Important Features

The features that emerged to be important during the survey are discussed as follows starting with the primary issues that have to be modified, followed by secondary problems related to more specific aspects of building evacuation:

- *Including priorities* in the calculation of the shortest paths. Applying different priorities on the evacuees depending on the degree of danger of their location can ensure the safety of all the evacuees. In [RHDW12], the priority-based distributed evacuation routing algorithm is developed to include such priorities in the calculations of the shortest paths. The evacuees with high priorities are evacuated to safe areas first, to ensure their safety. The other evacuees have to wait for the paths to be empty to start with their evacuation. Therefore, the alarms in the areas with high priority have to start earlier than the others to make sure the evacuation routes are still free.
- A *large number of simulations* is required with changing parameters automatically to identify worst case scenarios and difficulties that arise during an evacuation. Tools, such as PedGo[7], compare up to 500 different variants that are analyzed afterwards. The large number of variants enables one to achieve reliable results. Preferably, a tool communicates the results in predefined visualizations and alerts if critical locations in the building were found.
- The *travel times* estimated for the evacuees to move through the rooms in a building were researched: Predtetschenski and Milinski [PM71], for example, analyzed the traffic flow of people inside a building in normal as well as in panic situations. This data has to be used to

[7] http://www.traffgo-ht.com/de/pedestrians/products/pedgo/index.html

calculate the travel times in the evacuation tool to have a legal basis for future applications.

- *Waiting times* at bottlenecks have to fulfill certain requirements, too. Although the tool tries to prevent bottlenecks at all costs, it is sometimes impossible to cope with this condition. By reconstructing the calculations of [PM71] for the venue regulations, the maximal waiting time used for traffic jams was one minute. To reach this goal, the minimal stipulated door width has to be 1.2 meters. As there is also a prescribed width for a person crossing a door (60 cm), the minimal amount of people that can cross a door simultaneously is 2.
- A tool should also consider the guideline that stipulates that the *maximal evacuation time for a fire area* is around 10 minutes. For this purpose, the maximal length for an evacuation route must not be exceeded. The program should also be able to alert the users if any of the prescribed guidelines or rules are broken.
- *Include an alarm*, signaling the users of the tool that a dangerous situation emerged during evacuation. The alarm has to ensure that every critical situation is detected. For large buildings, this feature is mandatory, as it is impossible to have an overview of the whole building at once.
- To use a tool that supports evacuation in *buildings with large halls*, such as theatres, cinemas, etc., it is important to review the concept of a node. Therefore, a big hall should not be regarded as single node, but as separate graph including a set of nodes and edges.
- The *use of elevators* in evacuations. Nowadays, some elevators can be used during fire emergency situations. These elevators have to be considered when calculating the shortest escape paths. Nevertheless, these elevators should only be used for people with disabilities to avoid chaos due to capacity limitations.

4.6. Technical Limitations

Now, the technical limitations related to an evacuation tool are summarized. The more critical problems will be explained first:

- *Number and position of the evacuees inside a building*. It is difficult to assess these parameters for shortest path calculations in real evacuation scenarios. A possible solution is to use microchips inside badges in office buildings. The problem is that it cannot be assured that the workers always carry their ID cards. If the exact locations is unknown, another option is to count the number of people inside the building. Turnstiles are a possible solution for this. Another option is to use video surveillance. Various methods exist to count people in video. In [HBD05], for example, people are recognized using a skin color model and a probabilistic shape model of faces, while in [ZC07] a method is used to count the number of people in groups through the combination of human segmentation and group tracking. So-called intelligent buildings are able to incorporate these video surveillance systems to make an approximation of the evacuees inside the building and then calculate the escape routes with the evacuation supporting tool.
- The *guidance of the evacuees* through the building. Nowadays, the only suitable option is to use LEDs on the floor to guide the people dynamically through buildings. This method is used in air planes and is applicable even with smoke. Right now, this method seems to be the easiest and safest one to use in large buildings. In [Pet04], an example of remotely controlled signaling can be found. There, the signs are used to prevent traffic jams by signaling the drivers that there is a traffic jam ahead and lead them through alternative routes. These signs are remote controlled and the information displayed can be modified at any time. This technology can be used to control the signs or LEDs inside buildings, too.
- *Visualization of large buildings* suffers from their complexity. Only small parts of the building can be displayed simultaneously to keep track of the evacuation. A solution would be to display only one floor at once. Therefore, the software has to be able to switch between floors easily so that the visualization is kept simple. Another interesting feature would be a 3D view of the building to gain insight of the construction.
- *Visualization of the evacuees' movement*. The prototype shows that nodes and edges that change their colors and sizes abruptly are difficult to understand. A smooth transition between the different states of the items is important and ease to understanding of what is happening during the evacuation. Another option would be to represent groups of evacuees by small symbols that move through the building to show the positions of the evacuees clearly.

5. Conclusion

We developed a prototype to receive expert feedback. The result of the conducted expert survey can be used to develop efficient tools to support building evacuations. Beside legal aspects that have to be taken into account, the evacuees behavior and several technical limitations have to be considered. The survey also showed that some issues cannot be solved easily with the technology available nowadays. Knowing the exact number of people and their positions inside a building and guiding them to the nearest exit are problems that could be solved using different sensors that will be available in future. The survey shows useful insights to improve the tool and add features that will be essential in live evacuations or for building planning.

Acknowledgments

This work was partially funded by German Research Foundation (DFG) as part of the Priority Program "Scalable Visual Analytics" (SPP 1335).

References

[AAB08a] ANDRIENKO G., ANDRIENKO N., BARTLING U.: Interactive visual interfaces for evacuation planning. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (2008), pp. 472–473. 3

[AAB08b] ANDRIENKO G., ANDRIENKO N., BARTLING U.: Visual analytics approach to user-controlled evacuation scheduling. *Information Visualization 7*, 1 (2008), 89–103. 3

[HBD05] HARASSE S., BONNAUD L., DESVIGNES M.: Finding people in video streams by statistical modeling. In *Pattern Recognition and Image Analysis*, Singh S., Singh M., Apte C., Perner P., (Eds.), Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, pp. 608–617. 7

[Hos09] HOSSER D.: *Leitfaden Ingenieurmethoden des Brandschutzes*. Tech. rep., Vereinigung zur Förderung des Deutschen Brandschutzes e. V. (vfdb), Technisch-Wissenschaftlicher Beirat (TWB), Referat 4, Dietmar Hosser, 2009. 6

[KGS07] KIM S., GEORGE B., SHEKHAR S.: Evacuation route planning: scalable heuristics. In *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems* (2007), pp. 20:1–20:8. 2

[KHdVP10] KOBES M., HELSLOOT I., DE VRIES B., POST J. G.: Building safety and human behaviour in fire: A literature review. *Fire Safety Journal 45*, 1 (2010), 1–11. 6

[LGS05] LU Q., GEORGE B., SHEKHAR S.: Capacity constrained routing algorithms for evacuation planning: A summary of results. In *Advances in Spatial and Temporal Databases*, Bauzer Medeiros C., Egenhofer M., Bertino E., (Eds.), Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, pp. 291–307. 2

[Pet04] PETERS H.: Road warning system, Patent: DE102004013566. URL: http://www.freepatentsonline.com/DE102004013566.html, November 2004. 7

[PM71] PREDTETSCHENSKI W. M., MILINSKI A. I.: *Personenströme in Gebäuden*. Verlagsgesellschaft Rudolf Müller, 1971. 6, 7

[RHDW12] REDDY R., HÖFERLIN M., DAMBIER M., WEISKOPF D.: Visual analytics for dynamic evacuation planning. In *Proceedings of EuroVA 2012: International Workshop on Visual Analytics* (2012), pp. 13–17. 2, 3, 6

[Yin09] YIN D.: A scalable heuristic for evacuation planning in large road network. In *Proceedings of the Second International Workshop on Computational Transportation Science* (2009), pp. 19–24. 2

[YLJ*11] YANHUI W., LIQIANG Z., JINGTAO M., LIU L., DONGQIN Y., LIXIN Z.: Combining building and behavior models for evacuation planning. *IEEE Computer Graphics and Applications 31*, 3 (2011), 42–55. 2

[ZC07] ZHANG E., CHEN F.: A fast and robust people counting method in video surveillance. In *International Conference on Computational Intelligence and Security* (2007), pp. 339–343. 7

SIGRAD 2013, pp. 11–16
T. Ropinski and J. Unger (Editors)

Feature Tracking in Time-Varying Volumetric Data through Scale Invariant Feature Transform

Khoa Tan Nguyen[1] and Timo Ropinski[1]

[1]Scientific Visualization Group, Linköping University, Sweden

Abstract

Recent advances in medical imaging technology enable dynamic acquisitions of objects under movement. The acquired dynamic data has shown to be useful in different application scenarios. However, the vast amount of time-varying data put a great demand on robust and efficient algorithms for extracting and interpreting the underlying information. In this paper, we present a gpu-based approach for feature tracking in time-varying volumetric data set based on the Scale Invariant Feature Transform (SIFT) algorithm. Besides, the improved performance, this enables us to robustly and efficiently track features of interest in the volumetric data over the time domain. As a result, the proposed approach can serve as a foundation for more advanced analysis on the features of interest in dynamic data sets. We demonstrate our approach using a time-varying data set for the analysis of internal motion of breathing lungs.

1. Introduction

As the major focus of medical imaging has been the understanding of anatomical structures, vast research efforts has been dedicated to the acquisition and interpretation of anatomical modalities. While these techniques enable interpretation of high-resolution static anatomical images, time-varying data is now becoming more important, as the diagnostic workflow can be significantly improved by better understanding of organ function. This has led to the emergence of many functional modalities, which allow multimodal imaging of physiological processes alongside the anatomical image data serving as a context. In some cases the functional information is extracted from originally anatomical modalities,. The most prominent case of this development is probably fMRI (functional magnetic resonance imaging) that enables imaging of brain activity by detecting oxygen level changes in the blood flow.

The latest exploitation of anatomical modalities in a functional context arose with the recent advances in CT (computed tomography) imaging. Driven by the demands of imaging the beating heart, the scanning times of modern CT scanners nowadays enable a dynamic acquisition under movement. Through this technological advancement, new application cases become possible. For instance, 4D CT and 4D MRI can depict the breathing motion of the internal organs. As the amount of acquired data increases, there has been a great demand on new techniques that enable robust and efficient ways to interpret the vast amount of data under investigation.

In this work, we present a GPU-based implementation of the scale invariant feature transform (SIFT) algorithm [Low99, Low04] applied to the analysis time-varying volumetric data. The advantage of the proposed approach is two fold. First, it supports interactive feature detection. Second, it enables us to robustly and efficiently track features of interest in volumetric data over the time domain.

The remainder of the paper is structured as follows. In the next section, we review works that are related to our approach. In Section 3, we present an overview of the SIFT algorithm. We then present our GPU-based implementation in Section 4. We report the result of the proposed approach applied to the analysis of the internal motion of a time-varying volumetric data set of the lung in Section 5, and conclude the paper in Section 6

2. Related Work

In order to track features throughout a time-varying volumetric data sets as well as across different acquisitions, a robust feature tracking approach is mandatory. The SIFT algorithm proposed by Lowe fulfills this criterion [Low99]. SIFT focuses on extracting points of interest with a high saliency,

and that are stable across different scales. These points of interest are then represented by feature descriptors, which are invariant with respect to scaling, translation, and orientation. Since its introduction, SIFT has been widely used in the field of computer vision for image matching. While initially proposed for 2D images, SIFT has been extended to work with higher dimensional data and has been applied to different applications involving salient feature localization and matching such as motion tracking [SAS07, AKB*08], group-related studies [TWICA10], volumetric ultrasound panoramas [NQY*08], and complex object recognition [FBMB]. As an extension to the standard SIFT algorithm, Lowe proposed a guideline for optimal parameter settings that improve the accuracy as well as the performance [Low04]. To further improve its performance and make it interactively applicable, Heyman et al. proposed a GPU-based implementation of the SIFT algorithm enabling real-time feature detection and matching between images [HMS*07]. Although this algorithm was designed for, and tested on, 2D images of 3D objects, the underlying mathematical theory does not limit its extension to handle higher dimensional data. Scovanner et al. proposed a new approach to the creation of SIFT descriptors for the application of action recognition in video (2D images + time domain) [SAS07]. Cheung et al. generalized the scale space principle and applied SIFT to n-dimensional dataset [CH07, CH09]. Their extension has been applied to 3D MR images of the brain and 4D CT of a beating heart. In order to extend SIFT to handle high dimensional data set, they proposed the use of hyperspherical coordinate representations for volume gradients as well as multi-dimensional histograms to capture the distribution of gradient orientations in the neighborhood of detected feature locations. To improve the quality of the detected feature locations, Allaire et al. made use of the 3×3 Hessian matrix to compute the principle curvature at the detected feature locations [AKB*08]. This enables the filtering of features that are of less interest in medical data, such as non-blob and edge-like locations. In addition, the authors presented a technique that takes into account the tilt angle at the detected feature locations during the construction of SIFT descriptors to achieve full rotation invariance. The proposed extensions have been applied to complex object recognition in 3D volumetric CT data [FBMB]. Paganelli et al. further reported the result of the preliminary feasibility study on the application of SIFT to feature tracking in time-varying data sets [PPP*12]. Recently, Yu et al. compared SIFT to other feature detection algorithms and showed that SIFT achieve a balanced result between stability and performance [YWC12].

As we are interested in interactive feature tracking, besides the robustness, also the performance of the feature tracking is of interest. In comparison to previous works, we exploit the computing performance of the GPU through a GPU-based implementation applied to 4D data sets (3D volume + time domain).

3. 3D SIFT

While the SIFT algorithm has been initially proposed for 2D data, our work is based on recent extensions which have generalized it to 3D [CH07, SAS07, CH09]. The algorithm is performed in three successive stages: feature location detection, feature descriptor construction, and feature identification.

Feature location detection. In the first stage, the volumetric input data, $I(x,y,z)$, is convoluted with variable-scale Gaussian functions, $G(x,y,z,k\sigma)$, to generate a scale space, $L(x,y,z,k\sigma)$, as follows:

$$L(x,y,z,k\sigma) = G(x,y,z,k\sigma) * I(x,y,z) \tag{1}$$

where k is a constant multiplicative factor for separating scales in the scale-space.

The local extrema of the difference-of-Gaussian functions applied to this scale space are considered to be potential local features in the original volumetric data:

$$D(x,y,z,k^i\sigma) = L(x,y,z,k^{i+1}\sigma) - L(x,y,z,k^i\sigma) \tag{2}$$

Lindeberg and colleagues could show that these local extrema are a close approximation to the scale normalized Laplacian-of-Gaussian [Lin94], $\sigma^2 \bigtriangledown^2 G$, which are the most stable features in the input image [MTS*05].

In order to improve the stability of the detected feature location in volumetric data sets, Allaire et al [AKB*08] proposed the principal curvature thresholding technique to filter out the blob-like features, which are usually of no interest. The proposed technique is based on the analysis of the Hessian matrix, which describes the local curvature at a detected feature location:

$$H = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{xy} & D_{yy} & D_{yz} \\ D_{xz} & D_{yz} & D_{zz} \end{bmatrix}$$

The elements of H are computed using finite differences at the detected feature location in the corresponding scale, taking the anisotropy of the image into account. Let t_{max} be the curvature threshold, which is the ratio between the largest magnitude eigenvalue and the smaller one, the following conditions help to filter out blob-like features and improve the stability of the detected feature locations:

$$\begin{aligned} &(1) \quad tr(H)det(H) > 0 \quad \text{and} \quad \textstyle\sum det_2^P(H) > 0 \\ &(2) \quad \frac{tr(H)^3}{det(H)} < \frac{(2t_{max}+1)^3}{t_{max}^2} \end{aligned}$$

where $det_2^P(H)$ is the sum of second-order principal minors of H, $tr(H)$, and $det(H)$ are the trace and the determinant of the Hessian matrix, H, respectively.

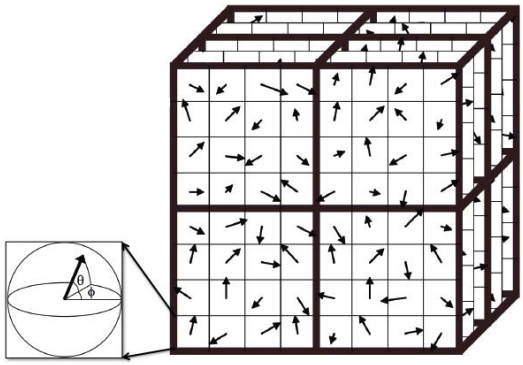


Figure 1: *The formulation of a 3D SIFT descriptor with its corresponding sub-volume.*

Feature descriptor construction. The aim of the second stage is to construct a unique descriptor to represent the detected feature location in such a way that it is most invariant with respect to rotation, scaling, and translation. The construction of such a descriptor is based on the gradient orientations in the neighborhood of the detected feature locations and presented as a histogram of gradient orientations. In 3D volumetric images, there are three angles that need to be handled during the construction of a descriptor: azimuth, elevation, and tilt angles. Consequently, while in 2D a 1D histogram can be used to describe a feature descriptor, in 3D a 2D histogram is required to capture the distribution of the azimuth and elevation angles. It is worth pointing out that while the azimuth and elevation angles can be derived directly from the orientation of the gradient, the tilt angles require more complex analysis [AKB*08]. A Gaussian-weighted kernel is commonly applied to the gradient magnitudes in order to put less emphasis on the gradients that are further away from the detected feature location during the construction of the feature descriptor. Figure 3 illustrates the formulation of a 3D SIFT descriptor with its corresponding sub-volume.

During the construction of a feature descriptor, the maximum peak in the histogram of gradient orientations represents the dominant orientation of the neighborhood around the detected feature location. As a result, the gradient orientations in the neighborhood region are then rotated in relation to the identified dominant orientation to attain a rotation invariance descriptor. While the resulting descriptor is a unique representation of the detected feature, discarding the other gradient orientations of lower magnitude can have negative impact on the feature identification stage. For instance, by creating additional descriptors for smaller peaks, which are of 80% the maximum peak in the histogram of gradient orientations, the result in the feature identification can be improved [Low04].

Feature identification. Once the features descriptors have been constructed for two data sets to be compared, they can be used to identify matching features. Therefore, different techniques such as RANSAC [FB81], Best-Bin-First (BBF) [BL97] have been used. Since a descriptor is basically a multi-dimensional histogram built in a special way, the Euclidean distance between descriptors is usually used, as it is a good indicator for a high probability match:

$$d(p,q) = \sqrt{\sum_{i=1}^{N}(p_i - q_i)}. \tag{3}$$

Here, p and q are two descriptors, p_i and q_i are the i-th elements of these descriptors, and N is the size of the descriptors. To find the matching features in the two data sets, I_1 and I_2, the Euclidean distances from each descriptor in I_1 to all descriptors in I_2 are computed. The minimum distance value is an indicator of a high probability match.

4. GPU-based Implementation

In 2007 Heymann et al. have proposed a GPU-based implementation of SIFT supporting real-time feature detection and feature matching for 2D images [HMS*07]. However, its extension to handle time-varying 3D volumetric data poses some additional challenges. As a result, within this section, we discuss and present our GPU-based implementation to address the issue when applying the SIFT algorithm to time-varying volumetric data sets.

Feature detection. During the construction of the scale-space representation of the input, the convolution operator is the most computational demand factor. Fortunately, the parallelism nature of GPU allows us to overcome this problem. For instance, the performance of the Gaussian convolution operator can be dramatically improved by using separable kernels. Additionally, the gradient calculation as well as the hyperspherical representation calculation can also be parallelized through the GPU-based approach. This enables us to improve the performance on the feature detection stage by a factor of 10 to 20.

Feature descriptor construction. In 2D, the neighborhood of size 8×8 is commonly use. The experimental findings from [Low04] show that the best results were achieved with a 4×4 array of histograms with 8 orientation bins in each which capture 45 degrees orientation differences. Consequently, each descriptor contains 128 elements. This allows a straightforward implementation of histogram calculation on the GPU that exploit the performance of the global and local memory architecture of the hardware.

In 3D, the neighborhood of a larger size, 16×16×16, is commonly used. The neighborhood is divided into 64 subregions of 4×4×4 voxels. Thus, a 1D representation of the descriptor contains 4096 elements. The size of the histogram

makes it difficult to have a GPU-based implementation of the histogram calculation process that exploits the performance of the memory architecture on the GPU. Moreover, it is worth noting that the size of the neighborhood needs to be adapted to the input data and the application in mind. Therefore, standard optimized histogram calculation on the GPU can not be easily adapted as the size of the histogram is large than the size local memory on the GPU. To overcome the hardware limitation, a multi-pass approach for histogram construction is required.

To avoid disruptive changes in the histogram of gradient orientations, the neighborhood around a detected feature location is usually divided into sub-regions. The histograms of gradient orientations of these sub-regions are first calculated and then combined together to form the final feature descriptor [Low04]. This poses a challenge to standard histogram calculation techniques on the GPU. Therefore, in this work, we implemented a generic OpenCL kernel that supports descriptor construction of an arbitrary neighborhood size. Moreover, the algorithm automatically switches to the optimized implementation based on the size of the descriptor.

In the initial SIFT operator, a smoothing operator is applied to the constructed histograms to include interpolation effects. In our implementation, we avoid this smoothing, and instead rotate the neighborhood around the feature locations to the dominant orientation. This enables us to achieve a bi-linear interpolation by default through a GPU-based implementation. With these modifications, we were able to realize an interactive GPU implementation with OpenCL, which runs ten times faster then previous approaches (see Section 5). Thus, we can not only use the SIFT algorithm for matching the different time steps on a global scale, but also can use it for interactive tracking of points of interest.

Feature identification. In the matching process, the Euclidean distances between each descriptor in the first input image and all the descriptors in the second one are calculated. Then the minimum Euclidean distance is identified to determine the highest probability match. The finding of the minimum distance is a reduction problem, which does not exploit the power of the parallelism nature of the GPU. As a result, we implemented a hybrid approach to the problem of feature identification. For instance, the computation of the Euclidean distance between one descriptor in the first input image and all the descriptors in the second image are performed in parallel using the GPU. This result is then passed to the CPU implementation to identify the minimum distance that serves as an indicator of a high probability match.

5. Test Case

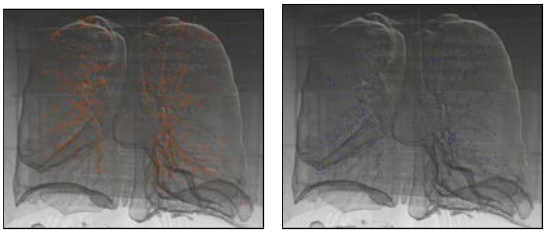
To show the impact of our introduced SIFT algorithm, we compare the results of our GPU-based implementation to the results achieved with the recent approach presented by Paganelli et al. [PPP*12]. We have tested our approach with the same data set, a 4D CT thorax scan[†] [CCG*09, CCZG09]. The data is given by ten equally sampled phases of the respiratory cycle in which the maximum exhale phase and the maximum inhale phase are denoted as $L0$ and $L5$ respectively. The reconstructed volumes have the dimensions of $512\times512\times128$ voxels of $0.97\times0.97\times2.5$ mm, while the reference landmarks in $L0$, and $L5$ were manually setup by an expert [CCZG09, CCG*09]. While the parameters used in our SIFT algorithm were set to match the ones used by Paganelli et al. [PPP*12], our approach allows more than one descriptors per detected feature location by considering orientations that are above 80% of the maximum peak in the histogram. In addition, due to the advantage of the GPU-based implementation, we do not apply smooth operator to the histogram to avoid disruptive changes of gradient orientations but instead rotate the neighborhood region to the dominant orientation, which implicitly takes advantage of the bi-linear interpolation on the GPU.

Feature location detection. We have evaluated the 4D CT SIFT matches, whereby we have computed the error as 3D residual distance between matching SIFT feature locations at the $L0$ and the $L5$ phase (SIFT $L0$–$L5$). Figure 2 illustrates the visualization of the inhale lung overlain with features of interest. While the detected features from the SIFT algorithm are colored as red spheres in Figure 3(a), the manually input reference landmarks from the data set are colored as blue spheres in Figure 3(b). The Mann-Whitney U test [MW47] was applied to this error distribution and the error distribution in the reference landmarks. Table 1 shows our results in comparison to the results reported in [PPP*12].

Besides the slightly higher number of matches between the maximum exhale and maximum inhale phase, the proposed approach has shown to improve the accuracy in the descriptor matching process, as we have achieved a lower median, 11.14 compared to 13.23, as well as a lower variability. In addition, the Mann-Whitney test confirms that the distributions of the residual distances in the reference landmarks and in the result of the enhanced SIFT operator are not significantly different (p-value = 0.736), which means that the proposed approach can be used to identify the feature locations.

Feature identification. To measure the impact on feature identification, we detect the feature locations in the time-series data and apply SIFT to consecutive volumes with two different variants. First, we always use the maximum exhale phase, $L0$, as a reference. Second, we move step-by-step along the breathing cycle such that the previous breathing phase is served as the reference for tracking the feature locations in the next breathing phase. For each approach, we computed the number of feature locations between all phases. Table 2 reports the number of detected feature locations which were preserved along the breathing cycle. While

[†] The data set is available at www.dir-lab.com.



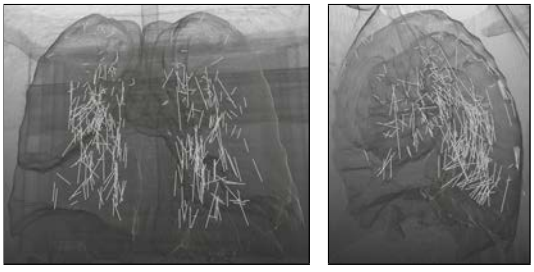
(a) (b)

Figure 2: *Visualization of the inhale lung. (*a*) is the visualization of the lung overlain with the detected features (in red) from the SIFT algorithm, (*b*) is the visualization of the inhale lung with the manually edited landmarks (in blue).*

	#Matches	Median (mm)	Variability (mm)
L0–L5	300	12.98	18.22
SIFT (L0–L5)	509	13.23	17.90
GPU-based SIFT (L0–L5)	525	11.14	12.78

Table 1: *Number of matches, median and variability of error distributions at maximum exhale, L0, and maximum inhale, L5, phases obtained by the proposed enhanced SIFT compared to the manual reference landmarks. Variability is the difference between the 25th and 75th percentiles.*

tracking along the breathing cycle allows us to achieve a higher number of preserved feature locations and excludes the trailing ones, tracking referred to a reference phase exclude the most stable feature locations over time. As seen in Table 2, the proposed approach provides more landmarks as feature locations, which makes it better suitable for motion estimation as well as visualization.



(a) (b)

Figure 3: *Visualization of the inhale lung overlain with the displacement vectors representing the transition of the detected features from L0 to L5. (*a*) is the visualization from the front, and (*b*) is the visualization from the side.*

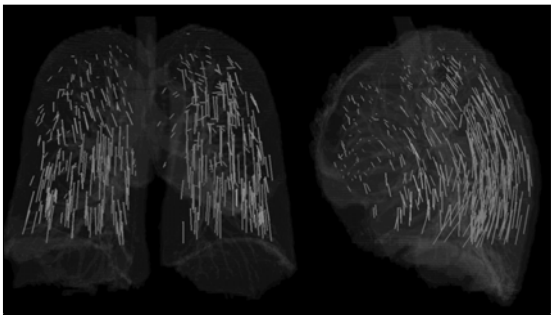


Figure 4: *Visualization of the inhale lung overlain with the displacement vectors representing the transition of the manually edited landmarks from L0 to L5 (courtesy of Castillo et al [CCG*09, CCZG09]).*

	SIFT (L0–L5)	GPU-based SIFT (L0–L5)
Reference (Phase 0)	117	243
Along breathing cycle	9	264

Table 2: *Number of preserved feature locations along the breathing cycle from the maximum exhale, L0, to the maximum inhale, L5, phase.*

As reported in Table 1 and Table 2, the proposed enhanced descriptor construction helps to increase the uniqueness of the descriptors at feature locations. Thus, it helps to improve the accuracy of the feature matching process in time-series data. It is also worth noting that by exploiting the power of the GPU implementation, we also achieved a much better performance in time. For instance, the overall time required for descriptors generation for each volume and descriptors matching between volumes is approximately 5.5 minutes. This is almost ten times faster than the result in [PPP*12], which is 50 minutes. As for the interactive visual analysis, we can precompute the feature descriptors, we can perform this process interactively.

Figure 3 shows the result of the proposed SIFT to track the detected features over the time domain. The displacement vectors (in white) show the transition of the detected features from the maximum inhale and exhale phases. Although the visual result is not very close to the visualization using manually input reference landmarks in Figure 4, the Man-Whitney test shows that there is no significant difference in residual distance distribution between the two results. As a result, this shows that the proposed SIFT is applicable to the application of automatic landmarks identification and tracking in time-varying dataset. This allows the proposed SIFT to serve as a tool for initial landmarks identification in different deformable image registration algorithms. Moreover, it can also be used to evaluate the results of different deformable image registration techniques.

6. Conclusion

In this paper, we presented a GPU-based implementation of the SIFT algorithm. By exploiting the power of the GPU, we do not only achieve better performance but also better results in comparison to the previously published work. The performance improvement of the algorithm enables us to investigate the effect of different parameters settings, such as σ in the scale-space construction, the level of the scale-space pyramid, to the quality of the detected features as well as the quality of the feature identification process in the future work. Furthermore, we would like to apply the proposed technique to the analysis of different dynamic data sets.

References

[AKB*08] ALLAIRE S., KIM J. J., BREEN S. L., JAFFRAY D. A., PEKAR V.: Full orientation invariance and improved feature selectivity of 3D SIFT with application to medical image analysis. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on* (2008), IEEE, pp. 1–8. 2, 3

[BL97] BEIS J., LOWE D.: Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on* (1997), pp. 1000–1006. 3

[CCG*09] CASTILLO R., CASTILLO E., GUERRA R., JOHNSON V. E., MCPHAIL T., GARG A. K., GUERRERO T.: A framework for evaluation of deformable image registration spatial accuracy using large landmark point sets. *Physics in Medicine and Biology 54*, 7 (2009), 1849. 4, 5

[CCZG09] CASTILLO E., CASTILLO R., ZHANG Y., GUERRERO T.: Compressible image registration for thoracic computed tomography images. *Journal of Medical and Biological Engineering 29*, 5 (2009), 222–233. 4, 5

[CH07] CHEUNG W., HAMARNEH G.: n-SIFT: N-dimensional scale invariant feature transform for matching medical images. In *Biomedical Imaging: From Nano to Macro, 2007. ISBI 2007. 4th IEEE International Symposium on* (2007), IEEE, pp. 720–723. 2

[CH09] CHEUNG W., HAMARNEH G.: n-SIFT: n-dimensional scale invariant feature transform. *IEEE Transactions on Image Processing 18*, 9 (2009), 2012–2021. 2

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395. 3

[FBMB] FLITTON G., BRECKON T., MEGHERBI BOUALLAGU N.: Object Recognition using 3D SIFT in Complex CT Volumes. In *British Machine Vision Conference 2010*, British Machine Vision Association, pp. 11.1–11.12. 2

[HMS*07] HEYMANN S., MULLER K., SMOLIC A., FROHLICH B., WIEGAND T.: Sift implementation and optimization for general-purpose gpu. In *Proceedings of the international conference in Central Europe on computer graphics, visualization and computer vision* (2007), p. 144. 2, 3

[Lin94] LINDEBERG T.: Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics 21*, 1-2 (1994), 225–270. 2

[Low99] LOWE D. G.: Object recognition from local scale-invariant features. *Computer Vision, 1999, The Proceedings of the Seventh IEEE International Conference on 2* (1999), 1150–1157. 1

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110. 1, 2, 3, 4

[MTS*05] MIKOLAJCZYK K., TUYTELAARS T., SCHMID C., ZISSERMAN A., MATAS J., SCHAFFALITZKY F., KADIR T., GOOL L. V.: A comparison of affine region detectors. *International journal of computer vision 65*, 1 (2005), 43–72. 2

[MW47] MANN H. B., WHITNEY D. R.: On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics 18*, 1 (1947), 50–60. 4

[NQY*08] NI D., QU Y., YANG X., CHUI Y. P., WONG T.-T., HO S. S., HENG P. A.: Volumetric Ultrasound Panorama Based on 3D SIFT. In *MICCAI '08: Proceedings of the 11th International Conference on Medical Image Computing and Computer-Assisted Intervention, Part II* (Sept. 2008), Springer-Verlag. 2

[PPP*12] PAGANELLI C., PERONI M., PENNATI F., BARONI G., SUMMERS P., BELLOMI M., RIBOLDI M.: Scale invariant feature transform as feature tracking method in 4d imaging: A feasibility study. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE* (2012), pp. 6543–6546. 2, 4, 5

[SAS07] SCOVANNER P., ALI S., SHAH M.: A 3-dimensional SIFT descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia* (2007), ACM, pp. 357–360. 2

[TWICA10] TOEWS M., WELLS III W., COLLINS D. L., ARBEL T.: Feature-based morphometry: Discovering group-related anatomical patterns. *NeuroImage 49*, 3 (2010), 2318–2327. 2

[YWC12] YU T.-H., WOODFORD O. J., CIPOLLA R.: A Performance Evaluation of Volumetric 3D Interest Point Detectors. *International Journal of Computer Vision 102*, 1-3 (Sept. 2012), 180–197. 2

SIGRAD 2013, pp. 17–24
T. Ropinski and J. Unger (Editors)

3D Visualization of Pre-operative Planning for Orthopedic Surgery

A. Steen[1] and M. Widegren[1]

[1]Linköpings Universitet, Sweden

Abstract

This paper presents ideas on 3D visualization for pre-operation planning of orthopedic surgery. The focus is on visualizing clinically relevant data for planning a Total Hip Replacement (THR) and investigating how 3D visualization can help improve the planning of procedures. The result is an application that showcases the ideas and reached conclusions. The application visualizes the fit of implants by shading them depending on distance to bone while at the same time giving the user contextual information of the bone anatomy. It also offers ways of measuring and visualizing important distances in a THR by visualizing the end points and the distance of a measurement.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications—Medical Visualization

1. Introduction

Approximately one million THR are performed annually in the world [UH12]. It is a procedure commonly done because of osteoarthritis (degradation of cartilage) in the hip joint. When the surgery begins it is important that the chosen implant fits the patient well and does not lead to complications, like pain or unequal leg length. Pre-operative planning is an important step in predicting and preventing complications [S E98]. The conventional method for planning for a THR has been to use hard-copies of X-ray images and printed out templates for implants to find out which implant to use and how it should be inserted.

Today it is more common to use software for pre-operative planning that is connected to the hospital's Picture archiving and communication system (PACS). Studies have shown that using a digital planning tool is as accurate as conventional planning [Ber07, Yon09]. Most of the currently available planning tools are in 2D, using X-ray images and 2D templates of the implants.

This paper presents research done to investigate the usefulness of using 3D visualization in orthopedic planning and presents a developed prototype showcasing the results. The main contribution of this work is a way of visualizing the fit of a hip implant while still giving the user contextual information about the thigh bone's anatomy. This is done with a color coded distance visualization combined with a clipping plane, an opaque volume rendering of the bone and semi-transparent slices of the bone. With respect to previous wok, our implementation improves the visualization of the contextual information of the thigh bone giving the user a better view of the bone's internal structure.

2. Related work

The importance of planning before performing an operation is well-understood and documented [S E98]. Planning helps the surgeon anticipate the correct implant size and can also help to anticipate possible intra-operative difficulties (more than 80% of intra-operative difficulties were anticipated in the study performed in [S E98]). At the time of writing, pre-operative planning is primarily done in 2D using conventional X-ray films or digital X-ray images but there is research being done in using 3D in the planning process.

2.1. Related work in 2D

There have been studies that compare traditional X-ray films and digital X-ray images. In their study, The et al. even con-

cluded that digital plans slightly outperform analogue plans in their accuracy regarding the implant size [Ber07]. Today, there are a number of well-established companies that offer software that allows the surgeon to perform planning in 2D before going into the operation room.

2.2. Related work in 3D

The accuracy of using 3D templates in planning has been studied.The study by Sariali et al. compared the accuracy of analogue 2D and 3D planning in THR and found that when counting both the stem and the cup in a THR, the plans would predict the implant that ended up being used in 96% of the cases when using 3D and in 16% of the cases when using 2D [E. 12]. Another study also found 3D planning to be accurate and repeatable, especially among less experienced surgeons [VLA*03].

In their article, Dick et al. discuss two methods for visualizing the 3D planning [Chr11]. The article puts its main focus on the important distances present in operation planning. It presents two different approaches and investigates the advantages and disadvantages of the two. The article also discusses some of the inherent problems with 3D visualization such as visual cluttering and occlusion and presents ways of dealing with them. These approaches were helpful as starting points during the development of this paper.

2.3. Comparison Between 2D and 3D

An advantage of using 3D data sets of the body is that it contains more information. Instead of seeing the resulting attenuation along an X-ray you have values for each voxel. The sizes are more accurate and the structures are invariant to if the patient has rotated their limbs during the scanning process. There are however some drawbacks in using 3D instead of 2D. One issue is that it is generally harder to intuitively navigate in a 3D environment since a regular monitor shows a 2D representation of the world and the mouse only has two degrees of freedom. Another disadvantage is that it is harder to present a general overview. Also specific to 3D is that the learning curve can be steep for people used to working with X-ray images. This might be more of a problem for radiologists than orthopedists.

Viceconti et al. concluded in their study that their 3D orthopedic templating tool was comparable in usage to the conventional planning with radiographs [VLA*03], indicating that the learning curve might not be so bad. They also concluded that the sizes predicted by their tool were generally more accurate than the sizes predicted with conventional planning, especially for the acetabular cup and especially for less experienced surgeons.

Otomaru et al. [Oto08] presented a method for creating pre-operative plans for THR using Computed Tomography (CT). They concluded that the plans created by their automatic software were roughly equivalent to the plan created by an experienced surgeon using the conventional method of radiographs and templates. They argue that this method will not only save time for experienced surgeons but also let less experienced surgeons create plans as good as if they were made by experienced surgeons.

3. Medical background

3.1. Anatomy

This section will discuss the parts of the body that are within the scope of this paper. These are the thigh bone (femur) and the pelvis.
The top part of the femur together with part of the pelvis can be seen in figure 1. The names of the different parts that are important for pre-operative planning are also marked in the figure. The head of the femur (caput) is connected to the pelvis via a socket in the pelvis (acetabulum) and is held in place by tendons (not pictured in the figure). The surface of the caput is covered with a layer of cartilage that together with a lubricating fluid (called “synovial fluid”) allows the caput to rotate with low friction.

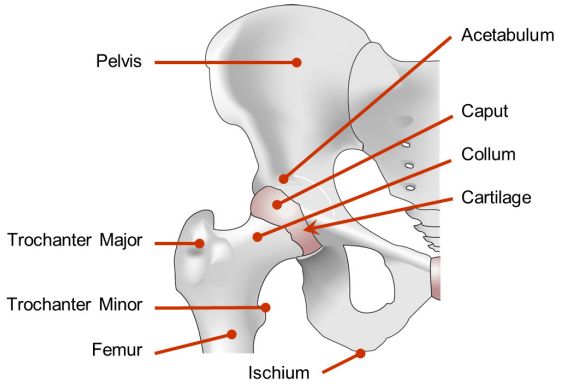


Figure 1: *Top part of the thigh bone (femur) and part of the pelvis*

3.2. Osteoarthritis

The hip joints are one of the areas in the body that are under practically constant stress daily. Because of this, most people run a risk of developing osteoarthritis with age. Osteoarthritis is the degeneration of cartilage in a joint. As the cartilage degrades, the body may try to compensate by producing more bone. Part of the smooth cartilage is then replaced by rough bone. This can eventually lead to bone rubbing against bone, causing friction and pain.

Osteoarthritis can be diagnosed using X-rays. Even though cartilage itself does not show on an X-ray due to

being too soft, the lack of cartilage can be identified using X-rays. There is a certain amount of space between caput and acetabulum and in the knee where the cartilage is and if this space (called "joint space") is smaller than it should be, that indicates a lack of cartilage. An example of this can be seen in figure 2. More information on osteoarthritis can be found in Ref [A.D].

Figure 2: *Two X-ray images, the left showing some space between caput and acetabulum in the top right part of the picture, indicating the presence of cartilage and the right showing less space indicating a lesser amount of cartilage.*

If the symptoms of osteoarthritis are not too severe, they can be treated with medication. For severe osteoarthritis in the hip, the solution is often a total hip replacement.

3.3. Total Hip Replacement

In a THR the caput is completely replaced by a metallic implant consisting of a stem and a head. While the outer shell of the femur (called cortical shell) is hard, the marrow canal inside is softer. To allow the implant stem to fit in the bone, parts of the marrow canal are removed. A so-called cup is inserted in the acetabulum which the head is then inserted into. The different parts of the implants come in many different sizes and models, a sample of the different parts can be seen in figure 3.

3.4. Pre-operative Planning

Pre-operative planning helps the surgeon prepare for the operation. It helps decide what tools will be necessary, how the procedure will be performed and also helps anticipate possible problems that may occur during the procedure.

One of the purposes of the pre-operative planning in a THR is to predict which implant model should be used as well as the sizes of the different parts of the implant. It is important that the implant fits well and therefore the surgical team will bring a number of implants of different sizes to be able to find a good match. A good prediction on the resulting implant will allow the surgical team to bring fewer implants into the operating room, lowering the number of tools that need to be sterilized. A good size estimate will also help decrease the time for the operation, thereby lowering the infection risk [GZP01, SMT*82, PR73].

Figure 3: *Photograph of a titanium stem, a ceramic head and a polyethylene acetabular cup.*

Two other important factors regarding the acetabular cup are how much bone will need to be removed to reach "fresh" bone and at which angle the cup is to be inserted. If the angle of the cup is off, the stem might become dislocated if the patient bends their leg too much.

Due to the removing of bone and insertion of an implant there is also a risk that, after the operation, the leg will not be the same length as before the operation. This is called unequal leg length, or "leg length discrepancy". This can be compensated for by choosing an appropriate implant.

Another important factor is the femoral offset, or the perpendicular distance from the center of rotation of the head of femur to the long axis of femur. Increasing the femoral offset after a hip replacement has been shown to have positive effects on the range of abduction and abductor strength [MMC*95].

A good plan will help the surgeon to solve the discussed issues by choosing implants accordingly.

3.5. Medical Imaging

Medical imaging is the process of obtaining images of the human body for use in medicine, such as diagnosing diseases and examining anatomy.

3.5.1. Projectional Radiography

In projectional radiography, images are obtained by having electromagnetic radiation such as X-rays hit a film after passing through an object (e.g. a human body). The varying densities of the object will absorb different amounts of radiation, resulting in an image where dense parts of the objects appear opaque and less dense parts appear transparent. This is the traditional method commonly referred to simply as "X-rays".

3.5.2. X-ray Computed Tomography

X-ray CT is an imaging technique where X-rays are used to create tomographic images, or slices, of specific parts of the human body. These slices can be combined into a 3D volume of the scanned body part.

4. Result

4.1. Prototype

This section will describe the resulting prototype that was developed. The prototype was developed in collaboration with Sectra Medical Systems AB in Linköping, Sweden. The prototype was implemented as a module in Sectra's already existing framework for 3D visualization.

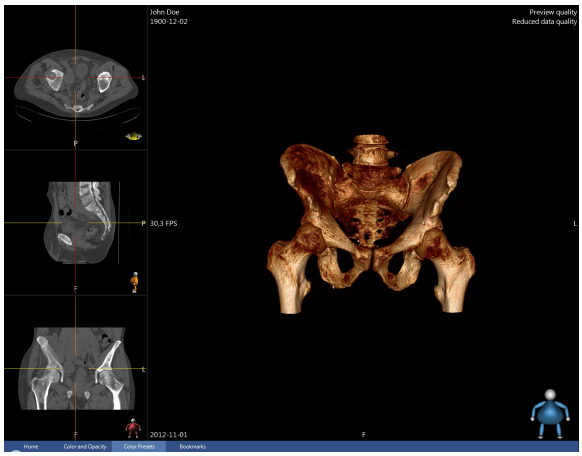


Figure 4: *Screenshot of Sectra's visualization software showing an frontal view of a pelvis*

Figure 4 shows a screenshot of Sectra's visualization software without any of our implementations. The three windows to the left are Multi-Planar Reformatting (MPR) views that show cross-sections of the volume. The volume is rendered in 3D in the large window to the right.

4.1.1. Basic Visualization and Control

The different implant parts are rendered as shaded polygon models. The models are semi-transparent to keep them from occluding bone. The distance shading of the implant, which is discussed later, can also be seen through the implant if it is semi-transparent. The user can switch to an implant with a different size by holding the mouse pointer over the implant and scrolling the mouse wheel. When an implant is selected, it becomes somewhat less transparent and a bounding-box is rendered around it to distinguish it from the other implant parts. Figure 5 shows a stem and an acetabular cup inserted in femur and acetabulum. The cup is selected as can be seen from the difference in transparency and the bounding box.

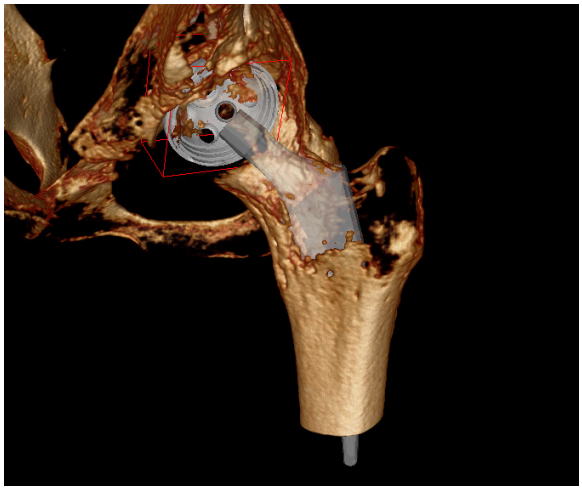


Figure 5: *Basic rendering of the implant parts with the acetabular cup selected*

4.1.2. Visualization of Relevant Data

One important aspect when deciding what implant to use is that it fits well. A good visualization of the fit is therefore essential. This section will describe the features that were implemented to help visualize the fit.

Bone Contours When positioning the implant model in the CT volume, it sometimes became difficult to see how close the implant was to the cortical shell without having to keep rotating the view. The implant was also often occluded by the bone. By allowing the contours of the bone to be visualized as well as the close proximity of the bone in the form of slices, the user can get a better understanding of how well the implant fits. The contours also help show the structure of the bone with less occlusion of the implant. The user can change the orientation and spacing as well as the number of slices. An implementation of the bone contour visualization can be seen in figure 6.

Implant Shading The shading of the implants was changed to visualize distance to the cortical shell of femur. An implant that is far away from any bone will be shaded using regular Phong shading but when being close to bone, it will interpolate between green and blue depending on the

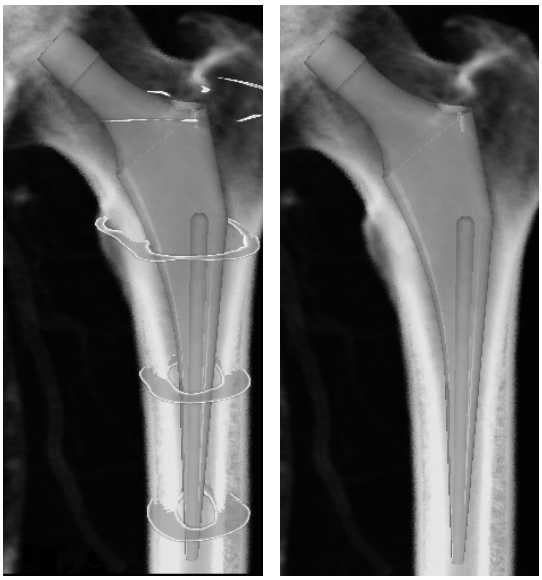


Figure 6: *Two close-up views of femur in an X-ray rendering of a CT volume. In the left image the hard outer shell is highlighted to show the contours of the bone, in the right image it is not.*

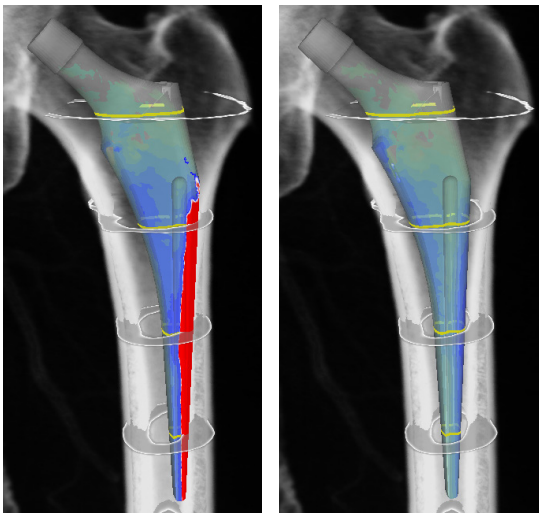


Figure 7: *Two renderings showing how the shading indicates when the implant intersects bone. In the left image the implant intersects the hard shell of femur, and in the right image it does not.*

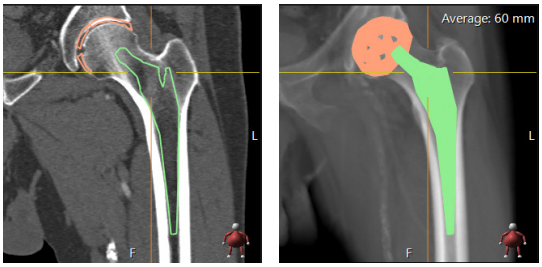


Figure 8: *The rendering of the implants in an MPR view for a thin slice (left) and a thick slice (right).*

distance to bone. Parts of the implant that are close to bone will be blue. The implant should be close to but not intersect with bone and therefore if some part of the implant intersects with bone, that part will be rendered bright red as a warning. Yellow lines that are aligned with the bone contour highlights are also drawn on the implant to make it easier to see how the outlines relate to the implant.

Implant Contour in MPR View Contours of the implants are visualized in the MPR views. For each triangle of the mesh, each edge is checked for an intersection with the plane that the MPR view defines. This gives a good outline of the mesh for a thin slice. For thicker slices, we also need to take into account edges that are inside the slice. So in addition to checking intersections with the edges, we also check if each edge of the triangle is inside the slice. If they are then we fill that triangle. So if the slice contains the whole model, a filled projection of it is rendered. An example of this can be seen in figure 8.

4.1.3. Acetabular Cup

To be able to visualize the cup's position relative to the volume, the cup had to be clipped by the clipping plane as having the whole cup visible made it occlude the volume. The parts of the cup that are close to the clipping plane are rendered yellow without any diffuse or other lighting. This is done to highlight the edges more clearly. The cup is not shaded differently with regards to intersections with bone, it only takes into consideration the distance from the clipping plane. The resulting visualization can be seen in figure 9.

4.1.4. Improved Control

Planning Guide A simple planning guide was implemented to help the user find implant parts with a good fit and put these in approximately the right place. When the guide is started, the user is asked to click on a number of landmarks in the MPR views. After the landmarks' positions are collected the program goes through a list of different implants to find the implant parts with the best fit and puts these into place. The camera is then moved to put focus on femur, a clipping plane is activated to give a cross-section of the bone and a number of bone contour slices are rendered. These operations make it easier to see how well the implant fits. Figure 10 shows the result of using the guide, with an implant automatically inserted into femur.

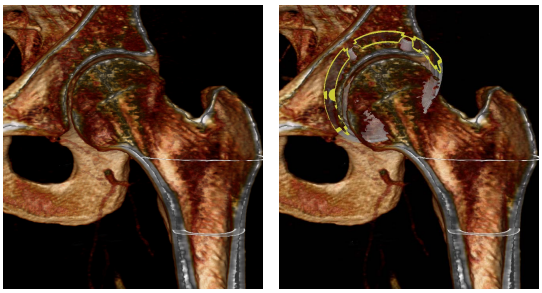


Figure 9: *Cross section of acetabulum before (left) and after positioning the acetabular cup (right).*

The goal with the guide is that with good accuracy when selecting the landmarks, the guide will create an operation plan that will give the user a good starting point and will only need some tweaking instead of having to manually move the implant parts into place and going through all possible implant sizes to find one with a good fit.

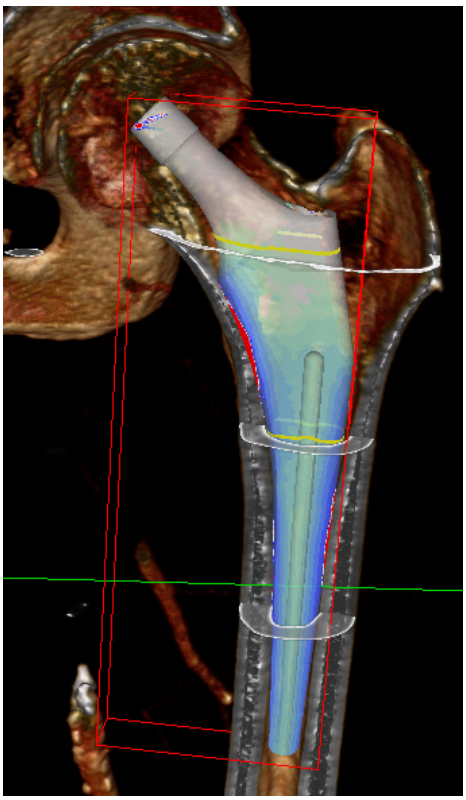


Figure 10: *Standard frontal view for visualizing how the stem fits in femur. Part of the clipping plane representation can be seen as a green out-line*

Widgets Widgets for translation and rotation in the 3D view were implemented. The coordinate system for the rotation and translation is defined by the femur coordinate system (which is set up with the planning guide), and the widgets translate and rotate in the coordinate system of the currently selected view.

4.1.5. Measurements

A tool for performing different measurements was implemented so that it could be used both manually by the user and automatically by the software itself. A simple measurement of length is visualized as a line between the two end-points and small shaded sphere at each end-point. The measured length is also displayed on the screen. The planning guide makes use of this measurement tool by providing measurements of the femoral offset by using the points provided by the user when using the guide. The distances between the points are calculated and visualized when the user completes the planning guide. The result can be seen in figure 11.

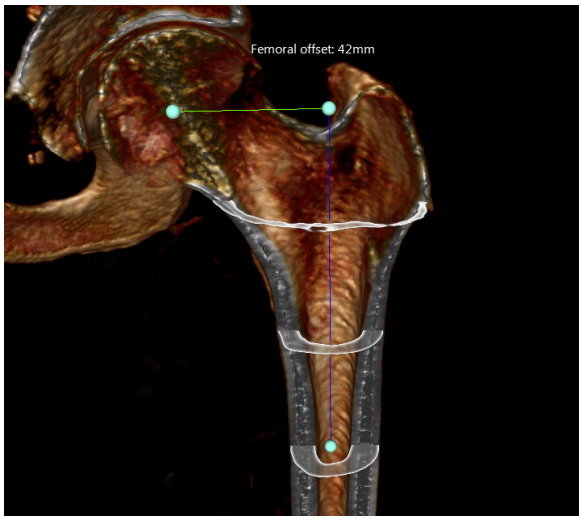


Figure 11: *Cross-section of femur with the visualization of the femoral offset*

5. Dicussion

We feel that the developed prototype helps give a good visualization of the plan by adding the 3D elements while still keeping the familiarity of 2D. Comparing our visualization methods to the ones presented in the related work we feel that our methods also improve the visualization of the contextual information of the thigh bone by rendering the bone opaque and utilizing a clipping plane to help fix the occlusion problem. The linear color map also gives a better understanding of the distance compared to a rainbow color map.

5.1. Current limitations

Some limitations existed in the 3D engine we used to implement our prototype. The volume and the polygon models are rendered separately, which lead to transparency not always working as intended. In the case of visualizing the stem in the femur, in the rendering pipeline the implant is always completely behind the volume or completely in front of the volume. This led to the implementation of an opaque bone rendering with clip planes. After some discussion with a medical expert at Sectra we feel that this is a good solution, since it is easy to see contextual information about the anatomy with this method.

Repositioning of different parts of the volume is not possible in the current engine. This means you currently can not visualize how the patient would look after the surgery is complete.

5.2. Radiation dose

The radiation dose from a CT scan increases the risk of cancer by approximately 1 in 2000. This is a noticeable increase from the 1 in 16000 increase by a pelvic X-ray. However the numbers are small compared to the 2 in 5 average risk we all have of developing cancer [NAM*12]. The additional risk of cancer introduced by a CT scan would also have to be weighted against the improved results obtained by using 3D planning reported from the discussed studies and the potential benefits from improving the 3D tools further. As concluded in the studies, a good plan will reduce the risk for intraoperative complications [S E98] as well as the risk for the need for revisional surgery (which in turn would require additional X-rays to be performed). Lowering these risks may be enough to warrant the higher dose of radiation obtained from a CT scan.

5.3. Acetabular Cup

Unlike when positioning the stem in the femur, it became hard to define a "good" or "bad" position for the acetabular cup. The hard cortical shell in femur should not be removed but when inserting the cup it is necessary to remove some hard bone from the pelvis in order to insert the cup. Different orthopedic surgeons also prefer to insert the cup at different depths. These factors led to the visualization of the cup to focus on position and visibility instead of trying to visualize the fit.

5.4. Bone cement

The current implementation assumes that the implant chosen is uncemented. A cemented implant needs a bit more space from the cortical shell for the cement coating to fit. Since most hip replacements in Sweden use cemented implants it is important that a production implementation of the application includes support for cemented implants.

5.5. Future work

This section will present some features and possible next steps for improvement that would help the prototype move closer towards a commercial product.

5.5.1. Usability testing

Usability testing would be a good method to assess how well the current interaction methods work in practice. The testing should ideally be performed with orthopedic surgeons familiar with using Sectra's current planning tools so that the feel and work flow of the program is consistent with Sectra's 2D planning tools.

5.5.2. Image analysis

Image analysis should be used together with the planning guide to calculate which implant fits best and how it should be placed. It could also assess the current fit, as well as find anatomical landmarks to calculate leg length discrepancy.

5.5.3. Repositioning

The ability to be able to visualize in 3D how the leg will look after the surgery would be a useful aid. It could help to visualize leg length discrepancy and femoral offset and show how different implants would affect these issues.

6. Conclusion

3D visualization and planning seems to have potential for improving the performance of pre-operative planning, as indicated by both our implemented prototype and the presented studies. The prototype was able to show relevant data and also visualize elements that are not as easily visualized in 2D. However, one of the biggest challenges for future work would be to make the interactions as intuitive as they are in 2D, as people working in the medical field have a long history of working in 2D and may not be as familiar with 3D. Furthermore, if the potential improvement of using 3D is large enough it might outweigh the extra risks from CT scans.

References

[A.D] A.D.A.M MEDICAL ENCYCLOPEDIA: Osteoarthritis. http://www.ncbi.nlm.nih.gov/pubmedhealth/PMH0001460/, Accessed January 23rd 2013. 3

[Ber07] BERTRAM THE ET AL.: Digital Versus Analogue Preoperative Planning of Total Hip Arthroplasties. *The Journal of Arthroplasty 22*, 6 (2007). 1, 2

[Chr11] CHRISTIAN DICK, RAINER BURGKART, RÃIJDIGER WESTERMANN: Distance Visualization for Interactive 3D Implant Planning. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011). 2

[E. 12] E. SARIALI ET AL.: Accuracy of the preoperative planning for cementless total hip arthroplasty. a randomised comparison between three -dimensional computerised planning and conventional templating. *Orthopaedics Traumatology: Surgery Research 98* (2012), 151–158. http://www.ncbi.nlm.nih.gov/pubmed/22381566. 2

[GZP01] G. ZANETTI R. G., PLATT R.: Cefoxitin versus erythromycin, neomycin, and cefazolin in colorectal operations. Importance of the duration of the surgical procedure. *Emerging Infectious Disease Journal 7*, 5 (2001), 828–831. 3

[MMC*95] MCGRORY B. J., MORREY B. F., CAHALAN T. D., AN K.-N., CABANELA M. E.: Effect of femoral offset on range of motion and abductor muscle strength after total hip arthroplasy. *British Editorial Society of Bone and Joint Surgery 77* (1995). 3

[NAM*12] N H., AM N., M K., N N., R A., W W., SF A., CL K., J R., Z T., H C., A M., MP E., DR L., HS C., EJ F., (EDS) C. K.: SEER Cancer Statistics Review, 1975-2009 (Vintage 2009 Populations). http://seer.cancer.gov/csr/1975_2009_pops09/, Accessed May 2nd 2013. 7

[Oto08] OTOMARU I.: Automated Preoperative Planning of Femoral Component for Total Hip Arthoplasty (THA) from 3D CT Images. *Journal of Biomechanical Science and Engineering 3*, 4 (2008), 478–489. 2

[PR73] PE C., R F.: A five-year prospective study of 23,649 surgical wounds. *Archives of Surgery 107*, 2 (1973), 206–210. 3

[S E98] S EGGLI, M PISAN, M.E. MÃIJLLER: The value of preoperative planning for total hip arthroplasty. *The Journal of Bone Joint Surgery (British Volume) 80*, 3 (1998), 382–390. 1, 7

[SMT*82] SHAPIRO M., MUÃŚOZ A., TAGER I. B., SCHOENBAUM S. C., POLK B. F.: Risk factors for infection at the operative site after abdominal or vaginal hysterectomy. *New England Journal of Medicine 307*, 27 (1982), 1661–1666. PMID: 6755254. 3

[UH12] UWE HOLZWARTH G. C.: Total Hip Arthroplasty. *Institute for Health and Consumer Protection, European Union Joint Research Center* (2012). 1

[VLA*03] VICECONTI M., LATTANZI R., ANTONIETTI B., PADERNI S., OLMI R., SUDANESE A., TONI A.: Ct-based surgical planning software improves the accuracy of total hip replacement preoperative planning. *Medical Engineering Physics 25*, 5 (2003), 371 – 377. 2

[Yon09] YONA KOSASHVILI ET AL.: Digital versus conventional templating techniques in preoperative planning for total hip arthroplasty. *Canadian Journal of Surgery 52*, 1 (2009). 1

Chading and Illumination

SIGRAD 2013, pp. 27–33
T. Ropinski and J. Unger (Editors)

Octree Light Propagation Volumes

John David Olovsson[1] and Michael Doggett[2]

[1]EA DICE, Sweden
[2]Lund University, Sweden

Abstract

This paper presents a new method for representing Light Propagation Volumes using an octree data structure, and for allowing light from regular point light sources to be injected into them. The resulting technique uses full octrees with the help of a separate data structure for representing the octree structure. The octree structure enables light propagation to be performed at a lower resolution than the base resolution, resulting in fast propagation and overall faster rendering times. The implementation of the technique is capable of rendering the Sponza scene at 9 *frames per second, which is 8% faster than the original LPV technique.*

1. Introduction

Realistic real-time illumination has been an important field of research in computer graphics and games for a long time. One large subset of real-time rendering techniques are dedicated to providing realistic local illumination. There are also techniques that try to model various aspects of global illumination, for example *diffuse indirect lighting*.

Unfortunately the nature of global illumination makes it computationally difficult to achieve in real-time. As such it is common for techniques targeted at real-time applications to use rather crude approximations or to rely on preprocessing. One problem with techniques relying on preprocessing is that they are usually static. The introduction of dynamic changes of the environment would require the preprocessing to be redone in order to retain correct illumination.

Light Propagation Volumes [Kap09] (LPV) is a technique which approximates global illumination, specifically diffuse indirect lighting, without any preprocessing stage, in real-time. While LPV runs in real-time, the uniform grid representation does limit the scalability. In order to achieve a more detailed and accurate result, the grid resolution would have to be increased. Unfortunately this also means that it would require more iterations to propagate the light throughout the scene. Such a high resolution representation would also be wasteful in any part of the scene which has no or relatively uniform geometry.

This paper proposes to use a full octree, where all nodes are allocated, instead of a uniform grid, to create a more efficient and scalable technique. We further investigate the possibility of using multiple fully dynamic point light sources instead of a single directional light source as used in the original LPV technique. Our approach incorporates omnidirectional point lights as the primary light sources and a dense octree implementation as the data structure. We present results that compare performance to an LPV implementation with a uniform grid and omnidirectional point lights.

2. Previous work

The LPV process is performed in four distinct steps. The scene is first rendered from each light source into *reflective shadow maps* [DS05]. The texels in these reflective shadow maps are then used to represent virtual point lights. After the reflective shadow maps have been created the contributions of the virtual point lights are injected into a uniform three dimensional grid. The grid encodes light color, intensity and direction as two-band spherical harmonics. After the injection stage these initial values in the grid are then propagated throughout the rest of the grid using an iterative process. In each iteration each grid element receives a contribution from each of the grid elements immediately surrounding it. The accumulated result of all the propagation iterations is then sampled during rendering to illuminate the final image.

LPV was extended with Cascaded LPV [KD10], which replaces the single uniform grid with multiple such grids with different density, positioned relative to the camera. These grids all have the same resolution, but because of the different densities they have different extents in the scene. A

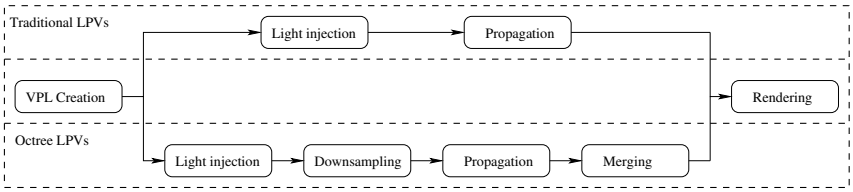


Figure 1: *An overview of the steps included in the octree based and the traditional light propagation volumes technique. The steps are divided into those that are the same for both techniques and those that need to be implemented specifically for one particular technique.*

high density grid is used close to the camera providing high detail close to the viewer while gradually lower density grids are used further away from the camera.

The data structure used in CLPV is in some ways very similar to that of the octree light propagation volumes which is presented in section 3. Both use multiple overlapping grids. The difference is that CLPV uses a small preset of grids which all have the same resolution. Octree LPV (OLPV) uses more volumes but with different resolutions. In CLPV the different grids also cover a differently sized area in world space, while OLPV grids all span the same volume, no matter their resolution. Also, CLPV does not need any separate grid to maintain the structure of the different grids. CLPV aims to prioritize spatial proximity to the viewer when it comes to the quality and accuracy of the technique. In this paper we focus instead on proximity to scene geometry and reflected indirect light instead.

Subsurface LPV [rea11] instead approximates the effects of subsurface light scattering by usilizing a structure identical to LPV. It also introduces a new method for injecting point lights into an LPV which is adapted by OLPV.

Sloan et al. [SKS02] introduced the concept of storing pre-computed indirect illumination using spherical harmonics. LPV uses this concept and in our implementation we use the first two bands of spherical harmonics to store light flux.

Crassin et al. [ea11] present a technique for real-time indirect illumination that uses octrees. Their technique uses a more complex sparse pointer based octree, resulting in more complex memory access patterns than our dense octree method.

3. Algorithm

There are essentially two important parts of the octree data structure. First and foremost are the octree levels themselves. These store the actual data, but have no information about the current structure of the octree. Then there is the index volume. It complements the octree levels by keeping track of how to index the octree levels and what structure the octree currently has. To create the initial light contribution, omni-directional light sources are rendered into six RSMs. These RSMs are converted to Virtual Point Lights which have their light propagated through the octree. The steps involved in this technique compared to the original technique are illustrated in Figure 1.

3.1. Octree Representation

We use a full octree which allocates memory for all nodes that could possibly exist down to a pre-defined level. By ensuring that this memory is laid out linearly, it is possible to compute the index or address for any node in the tree. Such a representation is highly inefficient with regard to memory usage if the represented data is sparse. It is however suitable in cases where the larger nodes represent a more coarse representation of the data.

Because of the ease of use and the very straightforward representation on the GPU, a full octree representation has been chosen to replace the light propagation volumes from the original technique. This imposes an additional cost in GPU memory but has the benefit that each individual level of the octree is just a uniform grid which is relatively easy to work with. There will essentially be several overlapping light propagation volumes of different resolutions covering the same scene.

In order to simplify the description of these octrees a basic notation is proposed. An octree is assigned a size, or resolution n, which should be a power of two $n = 2^1, 2^2, \ldots$. Given an octree that corresponds to a $n \times n \times n$ light propagation volume, the highest resolution level of that octree also has the size $n \times n \times n$. This level is the *first* level, or level $i = 0$. There are a total of $l = \log_2 n + 1$ levels in such an octree, $i = 0, \ldots, l-1$. Since every level of the octree is a three dimensional grid it can be indexed using three indices, $\hat{x}$, $\hat{y}$ and $\hat{z}$. Along with a level index i in which these indices are used they uniquely denote a single element in the octree. The indices are defined within the range $\hat{x}, \hat{y}, \hat{z} \in [0, \ldots, \frac{n}{2^i}]$. The notation $O_i(\hat{x}, \hat{y}, \hat{z})$ denotes the grid element on octree level i at $(\hat{x}, \hat{y}, \hat{z})$.

As a complement to the full octree representation a separate *index volume* can be introduced. The index volume would store integer indices instead of spherical harmonics.

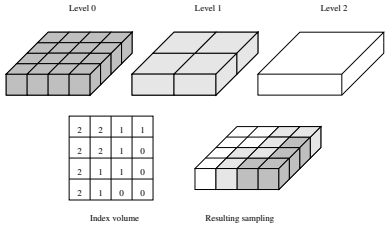


Figure 2: *A two dimensional slice of a flattened index volume, three levels of a full octree and the resulting sampling.*

This volume is denoted I. It could be represented as just another three dimensional structure of the same size n as O_0. Unfortunately, such a representation requires some quite inefficient operations to be added to the propagation step. In order to avoid this, a hierarchy of such structures is used instead. These are laid out in the same way as the data storage structures themselves and can be denoted I_i where $i = 0,\ldots,l-1$, just as for the data itself. It is also indexed using the same coordinates (x,y,z) as the data. Each element in the index volume is an integer index $I_i(x,y,z) = 0,\ldots,l-1$, originally initialized to $l-1$. This index is the number of the octree level to sample for a particular volume in world space. It will be populated with values during several of the steps performed throughout the technique. Details of this will follow later. During the sampling, the world position is then mapped to an element in the *index* volume. The value in the index volume is then used to do a second mapping into the correct octree level. The complication with this representation is that there are l index volume elements that could be picked for each world space position, one from each level i. To solve that, the element containing the minimum value is used.

An index volume I, with the levels I_i, $i = 0,\ldots,l-1$ where I_0 has size $n \times n \times n$, is considered to have size n. It can be indexed using the coordinates (x,y,z) where $x,y,z \in [0,\ldots,n-1]$ according to:

$$I(x,y,z) = \min_i I_i\left(\frac{x}{2^i}, \frac{y}{2^i}, \frac{z}{2^i}\right)$$

The resulting sampling in the octree corresponds to $O_{I(x,y,z)}(\frac{x}{2^{I(x,y,z)}}, \frac{y}{2^{I(x,y,z)}}, \frac{z}{2^{I(x,y,z)}})$. This is illustrated, using a flat visualization of the index volume, in Figure 2. Such a sampling is additionally given the simplified notation:

$$S(x,y,z) = O_{I(x,y,z)}(\frac{x}{2^{I(x,y,z)}}, \frac{y}{2^{I(x,y,z)}}, \frac{z}{2^{I(x,y,z)}}) \qquad (1)$$

There are several possible ways to use the proposed octree structure to replace an LPV. In this project it was chosen that the propagation of the injected light should be performed individually on each level of the octree. This allows the LPV propagation scheme to be used almost without any modifications in the octree representation. On first sight this may seem wasteful compared to just propagating on the highest resolution level, as in the original technique. However, the elements of the other levels in the octree cover a larger volume in world space. As a result the light on those levels will propagate further during each iteration. This allows the more detailed levels of the octree to be used to light the parts of the scene close to where the light was first injected. This will often be the areas where that light still has high enough intensity to be clearly visible. On the other hand, parts of the scene further away from this reflected light are unlikely to be hit by much indirect lighting. Because of that, those parts can be safely lit using a more coarse approximation of the indirect lighting.

3.2. Light Injection and Downsampling

Parts of the LPV technique were implemented using CUDA. CUDA was chosen to perform the light injection and light propagation in the place of shaders. Among other things this requires a different approach to light injection then the point based rendering used in the original papers [Kap09, KD10]. It also relies upon efficient sharing of memory between the GPGPU processing and the shaders.

Light is injected into the octree level O_0 in the exact same way as in the original technique. Each element $O_i(x,y,z)$, $i = 1,\ldots,l-1$ will cover exactly the same volume in world space as a specific set of eight elements from O_{i-1}. These elements are $O_{i\ \ 1}([2x,2x+1],[2y,2y+1],[2z,2z+1])$. In order to maintain a uniform range of light intensities in all octree levels, each element in O_i is populated by simply averaging the corresponding eight elements from O_{i-1}:

$$O_i(x,y,z) = \frac{1}{8}\sum_{\hat{x}=2x}^{2x+1}\sum_{\hat{y}=2y}^{2y+1}\sum_{\hat{z}=2z}^{2z+1} O_{i-1}(\hat{x},\hat{y},\hat{z}) \qquad (2)$$

The process of populating the layers $i \neq 0$ using the above averaging formula is called the *downsampling* step. It is considered as a separate step, since it is done after, and completely without any interaction with the injection step.

3.3. Propagation

After the injection and downsampling steps the octree contains an initial distribution of the diffuse reflected light in the scene. The index volume generally gives lower values close to geometry and higher values in empty areas. The next task is to allow this initial light distribution to propagate throughout the scene.

The propagation is done individually on each level of the octree. Each level is propagated in the same way as the single level was propagated in the original technique. But because of the hierarchy of the octree, light on lower levels of the octree is propagated further each iteration and can thus achieve coverage of the entire scene using much fewer iterations.

We use the light propagation scheme described by Kaplanyan and Dachsbacher [KD10], because it has better the-

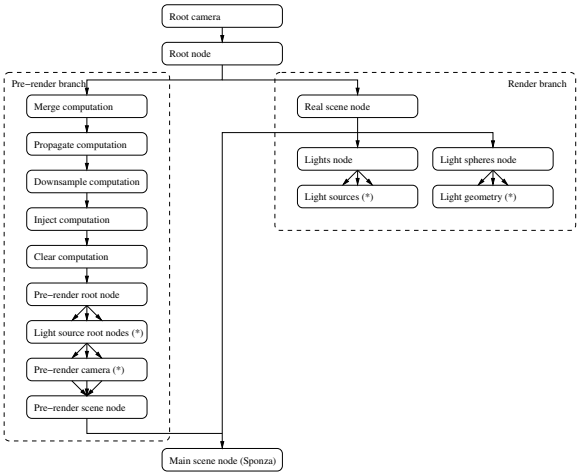


Figure 3: *Diagram showing the structure of the scene graph used in the real-time implementation. Non-essential branches have been omitted. Nodes marked with a star (⋆) and diverging incoming arrows, may exist in multiple copies for each parent node.*

oretical coverage. This includes maintaining a separate grid that contains the geometry of the scene.

3.4. Merging and Rendering

After the propagation, the scene can be rendered as in the original technique, with the exception that the sampling scheme from Equation 1 must be used. This is however not practical, since it would require the rendering pipeline to have access to, and be able to read both all the levels of the octree O_i, and all the levels of the index volume I_i. To avoid this an additional merge step is introduced. It stores all the sampled values using $S(x,y,z)$ into a single traditional $n \times n \times n$ LPV M where:

$$M(x,y,z) = S(x,y,z)$$

Once the merge step has been completed the result is just a single LPV that works in the exact same way as in the traditional technique. This also makes it possible to use the exact same rendering step as in the traditional technique.

3.5. Real-Time Implementation

A scene graph is used to put together the rendering passes and the setup is shown in Figure 3. The pre-render branch is responsible for performing all the steps of the technique from the creation of the RSMs to the merging step. Note that it does not perform the actual rendering. That is done in a separate branch, the rendering branch. Both these branches eventually end up in the main scene node which contains the loaded Sponza model.

Figure 4: *Samples of rendered images using the real-time implementation. The leftmost image shows only indirect illumination without any textures. The center image shows indirect illumination but with textures. The rightmost image shows the final rendering with both direct light, indirect light and textures. Note that the effect of indirect lighting has been slightly exaggerated in these images.*

4. Results

We have implemented our algorithm using OpenSceneGraph with osgCompute and CUDA. The final version of the real-time implementation has been manually tuned both based on visuals, performance and simulated results. This version uses a total of $k = 4$ iterations, since further iterations no longer contributes visually to the resulting intensity levels. The octree has a size of 32^3 and the RSMs have resolution 1024×1024, while the effective resolution with regard to the LPV technique is 256×256. The volume is sampled only once in each 4×4 region. This allows for high quality shadows while maintaining reasonable performance during light injection. In our implementation the parameters for propagation are based on those used in NVIDIA's implementation of CLPV from [Cor0]. The propagation factor is set to $p = 3$. A rendering using our technique of the Sponza dataset is shown in Figure 4. Figures 5, 6 and 7 show the full version, and other examples.

Figure 5: *The sponza scene rendered using only indirect illumination and without any diffuse texturing. Note that the indirect lighting has been slightly exaggerated in this image.*

There are small intensity variations between levels of the octree, but these have little visual impact on the final image. Also, light can propagate further in the lower levels of the

Figure 6: *The sponza scene rendered using only indirect illumination but with diffuse texturing. Note that the indirect lighting has been slightly exaggerated in this image.*

Figure 7: *The final rendering of the sponza scene with both direct and indirect illumination and diffuse texturing. Note that the indirect lighting has been slightly exaggerated in this image.*

octree leading to a slightly higher overall ambient lighting of the scene. Again this has minimal impact on the final image.

4.1. Error Measurement

There are a variety of ways to measure errors of the type of data contained in an LPV. Here we measure the average pairwise error between two spherical harmonics vectors a_e and b_e. The infinity norm $||v||_\infty = \max_i v_i$, or max norm, of the vectors is used to calculate the errors. Both the absolute error $||a_e - b_e||_\infty$ and the relative error $\frac{||a_e - b_e||_\infty}{||a_e||_\infty}$ is considered.

Each element in the octree can be denoted by four indices, the octree level i and the coordinates within that level (x,y,z). For each such element $i \neq 0$, it is possible to calculate an averaged value from the corresponding elements from level $i-1$ according to the downsampling in Equation 2. The downsampled value can be denoted $\hat{O}_i(x,y,z)$. There are then two error metrics, the average absolute error e_{abs}

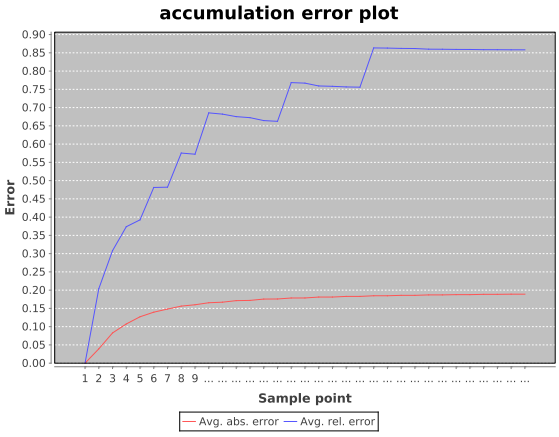


Figure 8: *The errors measured after each iteration during a simulation. Both the average absolute errors and the average relative errors are presented.*

and the average relative error e_{rel} of all elements in the octree $i \neq 0$. They are defined according to:

$$e_{\text{abs}} = \frac{7}{8^l - 8} \cdot \sum_{i=1}^{l-1} \sum_{p \in O_i} ||O_i(p) - \hat{O}_i(p)||_\infty$$

$$e_{\text{rel}} = \frac{7}{8^l - 8} \cdot \sum_{i=1}^{l-1} \sum_{p \in O_i} \frac{||O_i(p) - \hat{O}_i(p)||_\infty}{||O_i(p)||_\infty}$$

In the above expressions, $\frac{7}{8^l-8}$ is the inverse of the combined number of elements in $O_1, \ldots, O_{l-1}$, derived from the geometric sum, and $p = (x,y,z)$.

We have created a simulator that models just the light propagation. The two metrics e_{abs} and e_{rel} are the primary error measurements calculated by the simulator. It calculates the values for both of them after each iteration during the propagation. Figure 8 shows the errors for a simulation closely corresponding to the situation in the real-time implementation. This shows that the average relative error generally increases steadily up until approximately 20 iterations. After that it is essentially constant. This is simply because there is no noticeable light intensity left in the buffers at that point. This causes the accumulation buffer to remain the same after each iteration. The average absolute error exhibits a less erratic behaviour. It also stops increasing much sooner then the average relative error.

It makes sense for the errors to be increasing gradually each iteration. After all, during the downsampling, the values of O_i are chosen to be exactly that of $\hat{O}_i$. Each iteration then introduces a small error compared to what a newly downsampled value would give. This makes O diverge from $\hat{O}$ more and more until they stop changing. Using fewer iterations will result in less coverage by high resolution levels of the octree. At the same time it will also result in less diver-

gence from the ideal light distribution. Using many iterations can however void the problem with the higher divergence to some extent since the final solution would instead contain more of the higher resolution levels of the octree, and specifically O_0. Along with the related performance considerations it is still a trade off between speed, accuracy and visual quality.

4.2. Performance

The original LPV technique theoretically needs $k = 32$ iterations in order to propagate light throughout the entire volume, even though $k = 16$ iterations is often more then enough in practice. Using the OLPV technique, this number was lowered to $k = 4$. The intention is that this decrease should be able to more than make up for the added time to manage the more complex data structure.

All of the performance data in this section is generated on a computer with an NVIDIA GeForce GTX 480 graphics card. The focus has been on the newly added steps that are closely tied to the octree representation. This also includes modifications to previously existing steps from the original technique which were required for compatibility with the new octree structure. This includes the downsampling, propagation and merging steps, but generally not the creation of RSMs, light injection or the final rendering.

To ensure that timings were accurate, the CUDA kernels were run with `CUDA_LAUNCH_BLOCKING=1` to disable asynchronous kernel launches. This only ensures that CUDA kernels are not asynchronous, so other CUDA operations and OpenGL operations could run synchronously. We rendered 500 frames and timing results are shown in Table 1.

Table 1 shows that the OLPV technique outperforms traditional LPV by 8%. The propagation is the stage of the technique which experiences the highest increase in speed, almost 60%. This follows from the decrease in iterations from $k = 16$, when using the uniform implementation, to $k = 4$, when using the octree implementation. This speed increase is partially countered by the addition of the downsampling step, the merge step and managing the index volume.

The NVIDIA implementation of CLPV [Cor0] reaches a frame rate of between 60 and 80 frames per second with similar settings, but with a single directed light source. This equals a single RSM, rather than the 18 that are used in the implementation in this project.

5. Future Work

There are several possible improvements to the OLPV technique. Currently there is only one propagation factor for the entire octree. Analytically computing separate propagation factors for each level of the octree would give a more realistic propagation in the lower levels. Also propagating light on the lowest level of the octree instead on each level could improve performance and enable investigation of more sparse octree representations.

The biggest performance gain could be achieved by using a more efficient implementation of the light injection, such as point based light injection. Several optimiziations of the CUDA implementation are possible. Including making better use of local, shared and texture memory instead of global memory, ensuring memory access is optimized for cache access, and optimizing branching within warps to ensure threads are less divergent.

6. Conclusion

The paper presented a new technique for Light Propagation Volumes using octrees. It uses a representation based on full octrees and adds a so called index volume which keeps track of which level of the octree to use for each part of the scene. On top of the original technique and the new octree representation, the technique also adds new steps of downsampling the injected light and merging the octree into a traditional uniform LPV. Also light is injected from omnidirectional point light sources rather than just directional light sources.

With more restricted usage of RSMs and careful artist control of the scene, we believe that this technique could be used for real-time global illumination effects in games.

References

[Cor0] CORPORATION N.: Nvidia direct3d 11 sdk - diffuse global illumination demo. http://developer.nvidia.com/nvidia-graphics-sdk-11-direct3d, 0. 4, 6

[DS05] DACHSBACHER C., STAMMINGER M.: Reflective shadow maps. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), I3D '05, ACM, pp. 203–231. 1

[ea11] ET AL C. C.: Interactive indirect illumination using voxel cone tracing. http://research.nvidia.com/sites/default/files/publications/GIVoxels-pg2011-authors.pdf, 2011. 2

[Kap09] KAPLANYAN A.: Light propagation volumes in cryengine 3. http://www6.incrysis.com/Light_Propagation_Volumes.pdf, 2009. 1, 3

[KD10] KAPLANYAN A., DACHSBACHER C.: Cascaded light propagation volumes for real-time indirect illumination. http://dx.doi.org/10.1145/1730804.1730821, 2010. 1, 3

[rea11] RLUM ET AL J. B.: Sslpv subsurface light propagation volumes. http://cg.alexandra.dk/wp-content/uploads/2011/06/SSLPV_preprint.pdf, 2011. 2

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 527–536. 2

		Octree	(ms)		**Uniform**	(ms)
Part	**Count**	**Average**	**Total**	**Diff**	**Average**	**Total**
Frame	500	106	53193	-4801	115	57994
RSMCreate	9000	0	3053	316	0	2737
GVClear	500	1	671	-50	1	721
GVInject	1500	10	15593	-110	10	15703
GVDownsample	500	0	55	55	-	-
IXClear	500	5	2923	2923	-	-
LPVClear	500	8	4461	270	8	4191
LPVInject	1500	13	19929	-2884	15	22813
LPVDownsample	500	0	142	142	-	-
LPVPropagate	500	7	3788	-5615	18	9403
LPVMerge	500	0	39	39	-	-
Total				-4914 8.28%		

Table 1: *Data specifying the amount of times each part of the technique was executed and how long these executions took, both in total and on average. The data is presented both for the octree based implementation and the implementation using uniform light propagation volumes.*

SIGRAD 2013, pp. 35-40
T. Ropinski and J. Unger (Editors)

A Comparison of Volumetric Illumination Methods by Considering their Underlying Mathematical Models

Neda Rostamzadeh, Daniel Jönsson, and Timo Ropinski

Scientific Visualization Group, Linköping University, Sweden

Abstract

In this paper, we study and analyze seven state-of-the-art volumetric illumination methods, in order to determine their differences with respect to the underlying theoretical mathematical models and numerical problems potentially arising during implementation. The chosen models are half angle slicing, directional occlusion shading, multidirectional occlusion shading, shadow volume propagation, spherical harmonic lighting, dynamic ambient occlusion and progressive photon mapping. We put these models into a unified mathematical framework, which allows them to be compared among each other as well as to the global volume rendering equation. We will discuss the mathematical differences of the compared models and describe the numerical implications of the simplifications made by each method.

Categories and Subject Descriptors:I.3.3 [Computer Graphics]: Picture/Image Generation - Display Algorithms; I.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism

.

1. Introduction

Volumetric data sets are acquired in many scientific disciplines, and direct volume rendering (DVR) is frequently used to transform them into images. In recent years, several interactive volumetric illumination methods have been proposed in order to improve the perceptual qualities of DVR images. All of these methods are based on an optical model which maps data values in a volumetric data set to optical properties, such as color and opacity. Thus, these models describe how light particles inside the volume are generated, reflected or scattered, and simulation of multiple scattering, volumetric shadows and color bleeding are some of the effects that can be created by these advanced illumination methods.

In this paper, we analyze the underlying mathematical models of seven state-of-the-art volumetric illumination methods. We put all these models into a unified mathematical framework and compare them with respect to their mathematical simplifications, which are usually made in order to decrease rendering times. This enables us to compare the features of volumetric illumination methods by comparing their mathematical components to a reference model that includes all illumination features.

The illumination models that have been selected to be analyzed in this paper are half angle slicing [KPHE02], directional occlusion shading [SHB*08], multidirectional occlusion shading [SPBV10],shadow volume propagation [RDRS10], spherical harmonic lighting [KJL*12], dynamic ambient occlusion [RMSD*08] and progressive photon mapping [HOJ08]. These methods are selected based on their novelty, the number of citations as well as actual spread in real-world applications [LR11].

The paper is structured as follows. In the next section we will review work related to our approach. In Section 3 we will go through the volume rendering integral and introduce the unified mathematical framework that will be used to describe the mathematical models of the selected DVR techniques. In Section 4, we discuss the details on how to describe the selected illumination models within the unified mathematical framework. Section 5 contains a discussion of the different equations (models), the difference between them and what impact the mathematical simplifications have on the possible illumination effects. Finally, Section 6 concludes by summarizing the results.

2. Related Work

In recent years several interactive volumetric illumination models have been proposed [JSYR13]. In this work we

have chosen to study seven models that are representatives from each group of direct volume rendering techniques. We refer to [JSYR13, KFC*10, GJJD09] for a comprehensive overview of the methods and here mainly focus on related work in other areas.

A perceptual comparison of the images generated by many of the methods used in this work has been analyzed by Lindemann et al. [LR11]. Other perceptual studies include the work of Wanger et al. [WFG92], which investigated spatial relations in images and concluded that, for instance, shadows and perspective were important cues. Hubona et al. [HWSB99] studied the perception of relative position and size of objects in 3D space and found that, for instance, the use of shadows increase the accuracy of object positioning, but not the speed.

Some of the work not included in this comparison are for instance Vicinity Shading proposed by Stewart et.al. [Ste03], which simulates illumination of isosurfaces by taking into account neighboring voxels. Also in the area of computing occlusion are Penner and Mitchell [PM08] and Hernel et al. [HLY07]. The formed proposed a technique to compute the visibility around a voxel based on histograms and the latter presented a technique that computes local visibility of each voxel by integrating the opacity of each voxel in its surrounding sphere. None of the techniques presented here utilizes splatting, which for instance Zhang and Crawfis [ZC02] exploit to create shadows in volume rendering.

3. Volume Rendering Equation

To be able to describe each of the analyzed model within a unified mathematical framework, we first need to clarify the notation for the volume rendering equation considering global illumination. The following mathematical derivation of the volume rendering equation builds upon the work of Jensen and Jarosz et al. [JC98, Jar08], as it is a widely accepted notation for describing optical models. Light that travels through a participating medium is affected by emission, in-scattering, absorption and out-scattering. By taking these four terms into account, the change in radiance in direction $\vec{\omega}_o$ at the point x can be described by:

$$(\vec{\omega}_o.\bigtriangledown)L(x,\vec{\omega}_o) = -\sigma_a(x)L(x,\vec{\omega}_o) - \sigma_s(x)L(x,\vec{\omega}_o) + \sigma_a(x)L_e(x,\vec{\omega}_o) + \sigma_s(x)L_i(x,\vec{\omega}_o). \quad (1)$$

Here $\vec{\omega}_o$ is the viewing direction, $\sigma_a(x)$ and $\sigma_s(x)$ are the absorption and scattering coefficients, $L_e(x,\vec{\omega}_o)$ is the emitted radiance and $L_i(x,\vec{\omega}_o)$ is the in-scattered radiance. Integrating both sides of the Equation 1 along a straight path gives us the following integral:

$$\begin{aligned} L(x,\vec{\omega}_o) &= L(x_0,\vec{\omega}_o)\tau(x_0,x) \\ &+ \int_{x_0}^{x} \tau(x',x)\sigma_a(x')L_e(x',\vec{\omega}_o)\,\mathrm{d}x' \\ &+ \int_{x_0}^{x} \tau(x',x)\sigma_s(x')L_i(x',\vec{\omega}_o)\mathrm{d}x'. \end{aligned} \quad (2)$$

$$L_i(x',\vec{\omega}_o) = \int_{\Omega} f(x',\vec{\omega}_i,\vec{\omega}_o)L(x',\vec{\omega}_i)\mathrm{d}\omega_i,$$

where $L(x,\vec{\omega}_o)$ is the radiance scattered from point x inside the volume in direction $\vec{\omega}_o$ (the viewing direction), x_0 is the point behind the medium (the end point of the viewing direction), and $\vec{\omega}_i$ is the incoming direction (towards the point). $\tau(x,x')$ is the transmittance, which gives us the fraction of incident light that can travel unobstructed between two points x and x' in the medium along the straight line, and can be computed by the following equation:

$$\tau(x,x') = e^{-\int_{x'}^{x}\kappa(t)\mathrm{d}t}, \quad (3)$$

where $\kappa(x) = \sigma_a(x) + \sigma_s(x)$ is the extinction coefficient. The term $\int_{x'}^{x}\kappa(t)\mathrm{d}t$ is called optical thickness or optical depth.

Equation 2, which is the integral form of the radiative transfer equation, is called the volume rendering equation. The first term of the equation is the reduced surface radiance, the second term is the accumulated emitted radiance and the last term represents the accumulated in-scattered radiance inside the medium. We use the notation derived in this section to integrate the selected volumetric illumination models into a unified mathematical framework.

4. Unified Mathematical Framework

In this section the derivation of mathematical models for each of the seven chosen direct volume rendering techniques will be discussed in detail.

4.1. Half Angle Slicing

Half angle slicing is a slice-based approach introduced by Kniss et al. [KPHE02]. This model captures the appearance of translucency by approximating multiple scattering. Scattering effects can be fully captured using Equation 2 when taking the incoming light from all directions into account. However, this is generally expensive and the Half Angle Slicing method therefore makes an approximation by assuming that the light only scatters in forward direction. More specifically, the light is assumed to propagate within a cone in the direction of the light source only, instead of considering the incoming light from all directions.

The model can be expressed using the following equation:

$$\begin{aligned} L(x,\vec{\omega}_o) &= L(x_0,\vec{\omega}_o)\tau(x_0,x) \\ &+ \int_{x_0}^{x} \tau(x',x)\sigma_s(x')L_i(x',\vec{\omega}_o)\mathrm{d}x'. \\ L_i(x',\vec{\omega}_o) &= \int_{\Omega} f(x',\vec{\omega}_i,\vec{\omega}_o)L(x',\vec{\omega}_i)\mathrm{d}\vec{\omega}_i. \end{aligned} \tag{4}$$

We can imediately see that the emission term is removed when comparing it to Equation 2. The change that enforces forward scattering is less obvious and comes from limiting the evaluation of $L(x',\vec{\omega}_i)$ to only consider incoming light directions within a cone with an apex angle θ in the direction of the light source, which in turn is enabled by assuming a cone phase function:

$$f(x',\vec{\omega}_i,\vec{\omega}_o) = \begin{cases} \frac{1}{2\pi\cdot(1-\cos(\theta))} & \text{if } \vec{\omega}_i\cdot\vec{\omega}_l < cos(\theta) \\ 0 & \text{otherwise.} \end{cases}, \tag{5}$$

where w_l is the light direction.

4.2. Directional Occlusion Shading

Directional occlusion shading is a method that is able to create soft shadow effects of ambient occlusion. Similar to Half Angle Slicing, a specialized phase function is used to derive an occlusion factor. This phase function is known as a backward-peaked cone phase function of user specified aperture angle.

This model can be described by the following equation:

$$\begin{aligned} L(x,\vec{\omega}_o) &= L(x_0,\vec{\omega}_o)\tau(x,x_0) \\ &+ \int_{x_0}^{x} \tau(x,x')\sigma_s(x')L_i(x',\vec{\omega}_o)\mathrm{d}x'. \\ L_i(x',\vec{\omega}_o) &= L_b\int_{\Omega} \tau(x',x_0')f(x',\vec{\omega}_i,\vec{\omega}_o)\mathrm{d}\vec{\omega}_i. \end{aligned} \tag{6}$$

When comparing the above equation to Equation 2 we can see that the emission term is dropped and a term $L_b = 1$ is introduced, which represents the constant background intensity. Ω is also limited to the angles covered by the backward-peaked cone phase function, expressed using the following equation:

$$f(x',\vec{\omega}_i,\vec{\omega}_o) = \begin{cases} \frac{1}{2\pi\cdot(1-\cos(\theta))} & \text{if } \vec{\omega}_i\cdot\vec{\omega}_o < cos(\theta) \\ 0 & \text{otherwise.} \end{cases}, \tag{7}$$

4.3. Multidirectional Occlusion Shading

This method extends the directional occlusion shading technique allowing the light source to be placed anywhere within the hemisphere defined by the view vector. Even though it in practice and implementation wise is different from directional occlusion shading they are actually theoretically equivalent in our formulation. Thus, Equation 6 is also used to describe the light propagation together with the same cone shaped phase function in Equation 5.

4.4. Shadow Volume Propagation

Shadow volume propagation is an illumination model that supports scattering and shadowing effects. However, in order to have hard shadow borders that results in improvement of depth perception, the shadow computation is decoupled from scattering [RDRS10]. In addition, similar to Kniss et al. [KPHE02] the in scattered radiance is calculated by blurring the incoming light within a given cone centered about the incoming light direction instead of considering the scattering of light coming from all directions on the unit sphere.

This model can be specified using the following equation:

$$\begin{aligned} L(x,\vec{\omega}_o) &= L(x_0,\vec{\omega}_o)\tau(x,x_0) \\ &+ \int_{x_0}^{x} \tau(x',x)L_e(x',\vec{\omega}_o)\mathrm{d}x' \\ &+ \int_{x_0}^{x} \tau(x',x)L_i(x',\vec{\omega}_o)\mathrm{d}x'. \\ L_i(x',\vec{\omega}_o) &= \lambda(L(x',\vec{\omega}_l))\int_{\Omega} f(x',\vec{\omega}_i,\vec{\omega}_o)\gamma(L(x',\vec{\omega}_i))\mathrm{d}\vec{\omega}_i, \end{aligned} \tag{8}$$

where $\gamma(L(x',\vec{\omega}_i))$ and $\lambda(L(x',\vec{\omega}_l))$ are the functions that return the chromaticity and luminance, respectively.

As mentioned earlier, in order to generate hard shadow borders, the blurring does not apply to luminance part of the incoming light direction. In order to blur chromaticity, they use a strongly forward-peaked phase function:

$$f(x',\vec{\omega}_i,\vec{\omega}_o) = \begin{cases} C.(\vec{\omega}_i.\vec{\omega}_o)^{\beta} & \text{if } \vec{\omega}_i.\vec{\omega}_o < \cos(\theta); \\ 0 & \text{otherwise.} \end{cases}$$

The phase function is a Phong lobe whose extent is controlled by the exponent β and restricted to the cone angle θ which is used to control the amount of scattering. The constant C is chosen with respect to β.

4.5. Spherical Harmonic Lighting

Using several light sources and moving them can be challenging with respect to performance in DVR. Spherical harmonic lighting techniques allow different types of dynamic light sources to be used without increasing the computational burden much. Real time performance is achieved by decoupling visibility information and the light information.

This illumination model can be expressed using the following equation:

$$\begin{aligned} L(x,\vec{\omega}_o) &= L(x_0,\vec{\omega}_o)\tau(x_0,x) \\ &+ \int_{x_0}^{x} \tau(x',x)L_e(x',\vec{\omega}_o)\mathrm{d}x' \\ &+ \int_{x_0}^{x} \tau(x',x)L_i(x',\vec{\omega}_o)\mathrm{d}x'. \\ L_i(x',\vec{\omega}_o) &= \int_{\Omega} f(x',\vec{\omega}_i,\vec{\omega}_o)L_a(x_0,\vec{\omega}_i)V_R(x',-\vec{\omega}_i)\mathrm{d}\vec{\omega}_i. \\ V_R(x',\vec{\omega}_i) &= \tau(x',x'+R\vec{\omega}_i). \end{aligned} \tag{9}$$

Here, R is the distance from x' in direction of $\vec{\omega}_i$, and a spherical harmonic basis function is used to store visibility information, $V_R(x',\vec{\omega}_i)$, and the radiance distribution $L_a(x_0,\vec{\omega}_i)$. $V_R(x',\vec{\omega}_i)$ estimates the local visibility and shows how much of the incoming light reaches the point x' in direction $\vec{\omega}_i$. Efficient rotation in the spherical harmonic basis is enabled by assuming that an isotropic phase function:

$$f(x',\vec{\omega}_i,\vec{\omega}_o) = \frac{1}{4\pi}.$$

Spherical harmonics (SH) are orthonormal basis defined over the unit sphere S^2, where the 2D domain can be seen as the set of all possible directions. They allow both the visibility and lighting to be projected into spherical harmonic basis functions. In this case the original functions can be approximated as:

$$\begin{aligned} L_a(x_0,\vec{\omega}_i) &= \sum_{l=0}^{n}\sum_{m=-l}^{l} L_{l,m}Y_l^m(\vec{\omega}_i) \\ V_R(x',\vec{\omega}_i) &= \sum_{l=0}^{n}\sum_{m=-l}^{l} V_{l,m}Y_l^m(\vec{\omega}_i), \end{aligned} \tag{10}$$

where Y_l^m are the spherical harmonics basis functions by the degree l and order m with $l \geq 0$ and $-l \leq m \leq -l$. The coefficients are calculated using the following equation:

$$\begin{aligned} L_{l,m} &= \int_S LY_l^m(\vec{\omega})\mathrm{d}\vec{\omega} \\ V_{l,m} &= \int_S VY_l^m(\vec{\omega})\mathrm{d}\vec{\omega}. \end{aligned} \tag{11}$$

In order to find a function approximation of a finite number of coefficients the outer sum in 10 is truncated. For interactive purposes no more than $n_l \leq 3$ is being used, resulting in 16 coefficients. Orthonormality of the SH basis function provides us with efficient integral evaluation of in-scattered radiance (L_i) in equation 9. We can compute the integral using truncated SH expansions $L_a(x_0,\vec{\omega}_i) \approx L(x_0,\vec{\omega}_i)$ and $V_R(x_0,\vec{\omega}_i) \approx V(x_0,\vec{\omega}_i)$ as:

$$L_i(x',\vec{\omega}_o) = \frac{1}{4\pi}\sum_{l=0}^{n}\sum_{m=-l}^{l} L_{l,m}V_{l,m} \tag{12}$$

The implementation of the method uses a pre-processing stage, where local visibility is projected into the spherical harmonics space and then global visibility spherical harmonics expansions are computed by integrating the stored local visibility. The lighting environment is projected into spherical harmonic space as well. Equation 12 is evaluated at each step along the ray during ray casting in order to compute the radiance reaching the eye.

4.6. Dynamic Ambient Occlusion

Although the naming of this technique implies support for ambient occlusion only, it also provides an approximation of color bleeding.

The mathematical model can be described as follows:

$$\begin{aligned} L(x,\vec{\omega}_o) &= L(x_0,\vec{\omega}_o)\tau(x_0,x) \\ &+ \int_{x_0}^{x} \tau(x',x)L_i(x',\vec{\omega}_o)\mathrm{d}x' \\ L_i(x',\vec{\omega}_o) &= \int_{\Omega}\int_{x'}^{(x'+R\vec{\omega}_i)} |x'-x''|\sigma_a(x'')\mathrm{d}x''\mathrm{d}\vec{\omega}_i. \end{aligned} \tag{13}$$

In order to find the in-scattered radiance L_i for a the position x', the scattering of light coming from all directions on the sphere with radius R is considered. L_i is approximated numerically by integrating the occlusion over all the voxels lying in a distance R from x', weighted based on their distance and their absorption coefficient.

4.7. Progressive Photon Mapping

Progressive photon mapping (PPM) is a volume rendering technique proposed by Hachisuka et al. [HOJ08] and later reformulated by Knaus et al. [KZ11]. PPM naturally inherits the properties of standard photon mapping by Jensen [Jen96] and is thus capable of rendering complex illumination in participating media and in practice can be used to create unbiased renderings. Thus, Equation 2 can be solved without any simplifications, although the emission term is usually not solved using photon mapping.

The progressive photon mapping formulation by Knaus et al. [KZ11] is based on a probabilistic analysis of the error stemming from using a kernel radius to represent the photon extent. The variance (noise) and expected value (bias) of this error are studied. The main idea of PPM is to reduce both

variance and expected error continuously by averaging the results generated using independent photon maps.

The average error of the photon map radiance estimate over N samples can be expressed as:

$$\bar{\varepsilon}_N = \frac{1}{N}\sum_{i=1}^{N}\varepsilon_i, \tag{14}$$

where ε_i represents the error of radiance estimation in the pass i.

Convergence can be achieved by making sure that both the noise and bias goes to zero simultaneously as the number of passes goes to infinity ($N \to \infty$):

$$\begin{aligned} Var[\bar{\varepsilon}_N] &\to 0 \\ E[\bar{\varepsilon}_N] &\to 0, \end{aligned} \tag{15}$$

where

$$\begin{aligned} Var[\bar{\varepsilon}_N] &= \frac{1}{N^2}\sum_{i=1}^{N} Var[\varepsilon_i] \\ E[\bar{\varepsilon}_N] &= \frac{1}{N}\sum_{i=1}^{N} E[\varepsilon_i]. \end{aligned} \tag{16}$$

The main idea of PPM is to increase the variance in each step such that in average it still goes to zero. This will decrease the radius used for radiance estimates, which leads to a reduction of the expected error. Thus, there will be a trade-off between noise and bias in the radiance estimation.

5. Discussion

Comparing the half angle slicing mathematical model to Equation 2, the volume rendering integral, it is clear that the half angle slicing model assumes that particles inside the volume do not emit light. In addition, it only considers the incoming light within a cone in the direction of the light source when computating the in-scattered radiance. The assumption of forward scattering and the use of a specialized cone shaped phase function allow it to be implemented using a sweeping pass in the light direction.

Our derivation of the directional occlusion shading mathematical model shows that it is a specialization of the half angle slicing method and therefore inherits the same properties, i.e. the medium does not emit light and scatters light only in directions within a cone. The main difference is that the light direction must be aligned and point in the opposite view direction. Also, it only takes into account first order scattering and the in-scattered radiance is the attenuated constant background radiance scaled by the cone-shaped phase function. This model is capable of producing soft shadow effects of ambient occlusion.

As mentioned earlier, multidirectional occlusion shading is an extension of the directional occlusion shading model but in this model the light can be placed anywhere within the hemisphere defined by the view vector. A different numerical evaluation of the cone-shaped phase function make this possible, we refer to [SPBV10] for details.

The shadow volume propagation model is also similar to the mathematical model of half angle slicing. Both models solve the in-scattered radiance numerically by blurring the incoming light within a cone centered on the incoming light direction instead of considering the scattering of light coming from all directions on the unit sphere. However, shadow volume propagation decouples the chromaticity and luminance when computing the in-scattered radiance. Blurring applied to the chromaticity part, while the luminance is used directly without blurring. This produces hard shadow borders, which is shown to improve the depth perception [RDRS10].

The mathematical model of the spherical harmonic lighting [KJL*12] differs in that it assumes an isotropic phase function and assumes single scattering. The isotropic phase function allows the discretization of the mathematical model to use rotation properties of the spherical harmonic basis function in an efficient manner. Furthermore, by using the spherical harmonic basis functions to represent visibility and radiance separately, the light sources can be changed withouth recomputing the visibility.

Dynamic ambient occlusion approach is restricted to capture light interactions between adjacent voxels only since, in order to compute in-scattered radiance at a given voxel x, it only considers the voxels that lying in a certain distance from x. This model is the only model among other studied advanced illumination techniqus in this paper that the whole dataset does not affect the lighting in it.

6. Conclusion

We chose seven direct volume rendering approaches to study in this paper from several advanced illumination models that have been proposed recently. Three of them are slice-based techniques and the other ones can be combined with ray-casting based methods. The selected models are half angle slicing, directional occlusion shading, multidirectional occlusion shading, shadow volume propagation, spherical harmonic lighting, dynamic ambient occlusion and progressive photon mapping.

The mathematical model for each of the chosen illumination models has been derived. One can easily notice the differences of these models as they are written using a consistent mathematical formulation. Also, we can figure out the conditions and assumptions of each of these models by comparing their mathematical models with the most general volume rendering integral.

References

[GJJD09] GUTIERREZ D., JENSEN H. W., JAROSZ W., DONNER C.: Scattering. In *ACM SIGGRAPH Courses Program* (2009). 2

[HLY07] HERNELL F., LJUNG P., YNNERMAN A.: Efficient Ambient and Emissive Tissue Illumination using Local Occlusion in Multiresolution Volume Rendering. In *IEEE/EG Volume Graphics* (2007). 2

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 130:1–130:8. 1, 4

[HWSB99] HUBONA G. S., WHEELER P. N., SHIRAH G. W., BRANDT M.: The relative contributions of stereo, lighting, and background scenes in promoting 3d depth visualization. *ACM Trans. Comput.-Hum. Interact. 6*, 3 (1999), 214–242. 2

[Jar08] JAROSZ W.: *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, sep 2008. 2

[JC98] JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scences with participating media using photon maps. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 311–320. 2

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Rendering Techniques* (1996), Springer-Verlag, pp. 21–30. 4

[JSYR13] JÖNSSON D., SUNDÉN E., YNNERMAN A., ROPINSKI T.: A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering. *Computer Graphics Forum (conditionally accepted)* (2013). 1, 2

[KFC*10] KŘIVÁNEK J., FAJARDO M., CHRISTENSEN P. H., TABELLION E., BUNNELL M., LARSSON D., KAPLANYAN A.: Global Illumination Across Industries. In *ACM SIGGRAPH Courses Program* (2010). 2

[KJL*12] KRONANDER J., JÖNSSON D., LÖW J., LJUNG P., YNNERMAN A., UNGER J.: Efficient visibility encoding for dynamic illumination in direct volume rendering. *Visualization and Computer Graphics, IEEE Transactions on 18*, 3 (2012), 447–462. 1, 5

[KPHE02] KNISS J., PREMOZE S., HANSEN C., EBERT D.: Interactive translucent volume rendering and procedural modeling. In *In Proceedings of IEEE Visualization 2002* (2002), pp. 109–116. 1, 2, 3

[KZ11] KNAUS C., ZWICKER M.: Progressive photon mapping: A probabilistic approach. *ACM Trans. Graph. 30*, 3 (May 2011), 25:1–25:13. 4

[LR11] LINDEMANN F., ROPINSKI T.: About the influence of illumination models on image comprehension in direct volume rendering. *IEEE TVCG(Vis Proceedings) 17*, 12 (2011), 1922–1931. 1, 2

[PM08] PENNER E., MITCHELL R.: Isosurface ambient occlusion and soft shadows with filterable occlusion maps. In *Proceedings of the Fifth Eurographics / IEEE VGTC conference on Point-Based Graphics* (2008), SPBG'08, pp. 57–64. 2

[RDRS10] ROPINSKI T., DORING C., REZK-SALAMA C.: Interactive volumetric lighting simulating scattering and shadowing. In *Pacific Visualization Symposium (PacificVis), 2010 IEEE* (2010), pp. 169–176. 1, 3, 5

[RMSD*08] ROPINSKI T., MEYER-SPRADOW J., DIEPENBROCK S., MENSMANN J., HINRICHS K. H.: Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum (Eurographics 2008) 27*, 2 (2008), 567–576. 1

[SHB*08] SCHOTT M., HANSEN C., BOULANGER K., STRATTON J., BOUATOUCH K., SCHOTT M., PEGORARO V., HANSEN C., BOULANGER K., STRATTON J., BOUATOUCH K.: A directional occlusion shading model for interactive direct volume rendering, 2008. 1

[SPBV10] SOLTÉSZOVÁ V., PATEL D., BRUCKNER S., VIOLA I.: A multidirectional occlusion shading model for direct volume rendering. *Comput. Graph. Forum 29*, 3 (2010), 883–891. 1, 5

[Ste03] STEWART A. J.: Vicinity shading for enhanced perception of volumetric data. In *IEEE Visualization* (October 2003). 2

[WFG92] WANGER L., FERWERDA J., GREENBERG D.: Perceiving spatial relationships in computer-generated images. *Computer Graphics and Applications, IEEE 12*, 3 (1992), 44–58. 2

[ZC02] ZHANG C., CRAWFIS R.: Volumetric shadows using splatting. In *Visualization, 2002. VIS 2002. IEEE* (2002), pp. 85–92. 2

SIGRAD 2013, pp. 41-46
T. Ropinski and J. Unger (Editors)

No more texels, no more facets: Emerging trends in GPU procedural shading

Stefan Gustavson, Linköping University, ITN
stefan.gustavson@liu.se

Abstract

Procedural textures have long been a staple of off-line rendering, and impressive creative results have been accomplished by using procedural methods to their advantage. As GPU speeds and computational capabilities continue to increase, procedural texturing will likely become a useful tool also for real time rendering. In fact, it is already possible to generate procedural patterns of considerable complexity at real time frame rates on a modern GPU. Even on current (2013) low-end and mobile GPUs, the programmable shading system offers considerable processing power that often remains largely unused. Such untapped resources could be used for procedural patterns and procedural geometry computed entirely on the GPU.

This article presents the state of the art in the field. Most of this is yet to be implemented in commercial projects like games, VR and visualization applications. Code for the shader examples in this article is available under permissive open source licenses or as public domain software and is collected on the address:
`http://www.itn.liu.se/~stegu76/sigrad13`

1 History of procedural shading

The concept of a *surface shader*, a small program to compute the color of a surface point using the view direction, surface normal, material properties and information on the scene illumination, has been around for a large portion of the history of computer graphics. The concept was formalized in the commercial product Photorealistic RenderMan from Pixar, first released to the public in 1989 [11, 8]. While the RenderMan Interface was made famous by its software implementations, the original intent was to make a hardware renderer, and a prototype device named RM-1 was manufactured and sold by Pixar for a few years in the 1980's. The Academy award winning Pixar short film "Luxo Jr." was rendered on that custom hardware.

After Pixar abandoned their hardware renderer project, hardware graphics rendering took off in a different direction with Silicon Graphics' IrisGL and OpenGL, where real time performance took priority over both image quality and flexibility. For a long time there was a harsh disconnect in terms of image quality between real time graphics, where a frame needs to be rendered in fractions of a second, and cinematic pre-rendered graphics, where rendering times are allowed to extend to several hours per frame.

2 GPU procedural shading

Graphics hardware has now come to a point where the image quality can match off-line rendering from about a decade ago. Around 2004, programmable shading was introduced into the hardware rendering pipeline of mainstream GPUs, which added considerable flexibility. The line between real time graphics and off-line rendered graphics is blurring, and GPU rendering is now successfully used not only for real time rendering, but also for acceleration of off-line rendering. In a way, we have now come full circle from Pixar's RM-1 hardware in the 1980's through a long tradition of software rendering, and back to a focus on hardware.

Instead of traditional fixed-function lights and materials, with a predetermined, rigid computational structure and a small set of parameters to control both the surface appearance and the illumination, programmable shading has finally set real time graphics free of the simplistic illumination and surface reflection models from the 1980's, and made it possible to explore more modern concepts for real time lighting, shading and texturing.

To date, this new-found freedom has been used mainly to explore better illumination and reflection models, which was indeed an area in great need of im-

provement. However, another fascinating area remains largely unexplored: procedural patterns. A good review and a long standing reference work in the field is [1]. This article is an attempt to point to the many fascinating possibilities in that area for real time applications.

3 Procedural patterns

A procedural pattern is a function over spatial coordinates that for each point (u, v) on a surface describes its color, transparency, normal direction or other surface property as $f(u, v)$. This function is defined over a continuous domain (u, v), not a discrete set of sample points, and contrary to a texture image it is computed directly for each surface point for each frame, rather than sampled and stored for repeated use. While it might seem wasteful and contrary to common sense in computing to throw away previous hard work and compute each point anew for each surface point, it offers many clear advantages:

- No memory is required, and no memory accesses are made. This saves bandwidth on the often congested memory buses of today's massively parallel GPU architectures.
- Patterns can be computed at arbitrary resolution.
- Patterns can be changed by simple parameters.
- Animated patterns come at no extra cost.
- 3D and 4D functions can be used for texturing, removing the need for 2D texture coordinates.
- The shader program can perform its own analytic anti-aliasing.

Of course, there are also disadvantages:

- Not all patterns can be described by simple functions.
- Creating a good procedural pattern is hard work.
- The pattern must be programmed, which takes a very different skill than painting it or editing a digital photo. Such skill is hard to find.
- Production pipelines for real time content are firmly set in their ways of using texture images as assets.
- Current GPU hardware is designed to provide huge bandwidth for texture images, making it somewhat counter-productive not to use them.

Procedural patterns are not a universal tool, and they are not going to replace sampled texture images altogether. However, many animation and special effects studios have seen an advantage in using procedural patterns for many tasks, not least because of the ability to easily tweak and change the appearance of any surface in the scene late in the production process.

A modern GPU has a massively parallel architecture with lots of distributed computing power, but with thousands of computing units sharing a common memory of limited bandwidth. To counter the imbalance, local memory is used to implement cache strategies, but fact remains that memory-less procedural texturing is becoming ever more attractive for real time use, at least as a complement to traditional texturing. In a texture intensive shader, the execution speed is often limited by memory access bandwidth, and the arithmetic capabilites of the GPU processing units are often not fully utilized. Sometimes the computing power to generate a procedural pattern is available, but remains unused. Moreover, computations in local registers may be more power efficient than accesses to global memory.

4 Simple patterns

Some patterns are very suitable for description as functions. Stripes, spots, tiles and other periodic functions can be described by a combination of modulo functions and thresholding operations on the surface parameters (u, v) or directly on the object coordinates (x, y, z). Some examples with GLSL code are shown in Figure 1.

5 Noise

The world around us is not clean and perfect. The natural world is largely a result of stochastic processes, the same sort of random variation is seen in manufactured objects due to process variations, dirt and wear. Recognizing this, Ken Perlin created *Perlin noise* in the 1980's [10] as a tool for adding complexity and variation to surfaces and models. Variations on his function have seen heavy use ever since in most off-line rendering. Perlin noise has several nice and useful properties:

- It is fairly easy to compute.
- It is band limited and reasonably isotropic, which makes it suitable for spectral synthesis of more complex patterns.
- It has a well defined derivative everywhere.

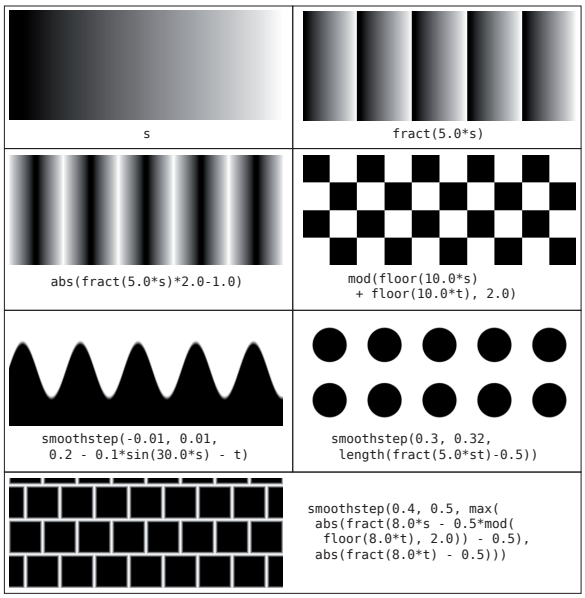


Figure 1: Some simple procedural patterns in GLSL.

Noise by itself does not make a very interesting pattern, although it can be used to generate waves and bumps. More interesting patterns are generated by performing spectral synthesis (combining noise of several spatial frequencies), and by using color tables or thresholding the result. Noise really shines when it is used to add variation to other, more regular patterns. Some patterns based on Perlin noise are shown in Figure 2.

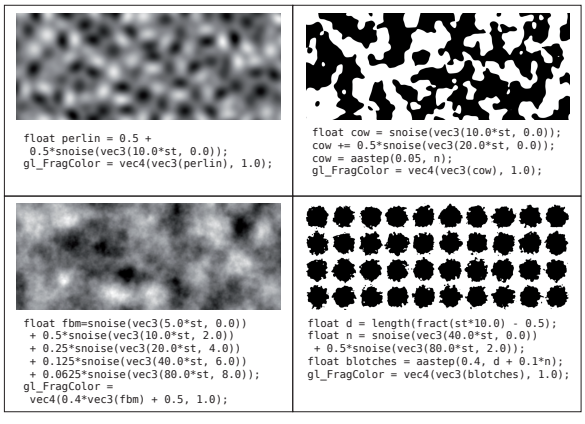


Figure 2: Some Perlin noise patterns in GLSL.

Another useful and popular pseudo-random pattern function is cellular noise, introduced by Stephen Worley [12]. It is a different kind of function than Perlin noise, and it generates a different class of patterns: tiles, cells, spots or other distinct features distributed across a surface with a seemingly random placement. Some patterns based on cellular noise are shown in Figure 3.

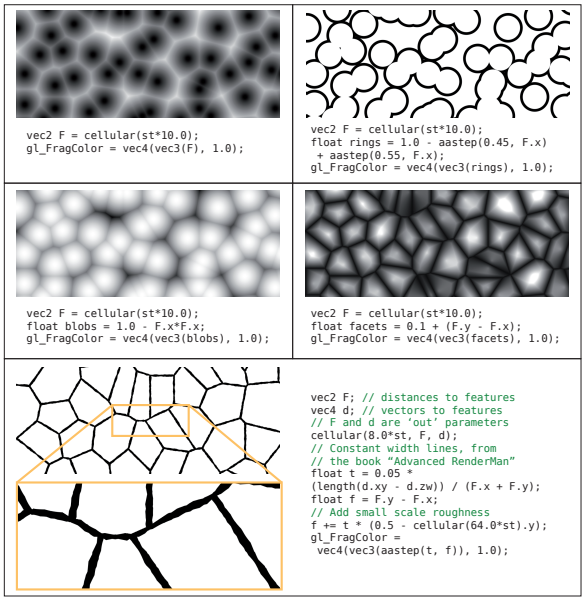


Figure 3: Some simple procedural patterns in GLSL.

Noise is ubiquitous in off-line rendering, in particular when rendering scenes from nature, but it has not been a good fit for real time rendering because of its relative complexity. Noise is not a very complicated function, but a typical noise-based texture needs several noise values per surface point, and each value takes more work to compute than a simple texture image lookup. When GLSL was designed, noise functions were included in the language, but to date (2013) they remain unimplemented. The desire to implement hardware support for Perlin noise has to compete for silicon area with other design goals of a modern GPU. Recent research has provided some very hardware friendly noise functions, and it might be time to reconsider. However, what GPU manufacturers choose to implement is largely beyond our control. In the meantime, we can get by with shader-implemented software versions of noise. Recent advances in GPU capabilites in combination with new variations on the kind of noise functions introduced by Perlin and Worley has provided hardware-friendly noise functions implemented in GLSL [9, 6]. The functions are licensed with a permissive MIT li-

cense, they are very easy to use and show good real time performance on modern GPUs. A visual example of the kind of patterns that can be generated with these functions, at fully interactive frame rates even on a mid-range GPU, is in Figure 5.

6 General contours

Another problem with traditional image-based texturing is crisp edges. Formally, an edge is not a feature that is suitable for sampling, because it is not band limited. Other representations have been proposed over the years to render edges and contours on 2D surfaces, but none have prevailed. Recently, NVIDIA presented an extension to OpenGL named `NV_path_rendering`, which is their take on rendering 2D graphics directly from contour descriptions of the kind that are used in PostScript, PDF, SVG and other modern standards for 2D object graphics. While successful, it taxes even a top performing GPU considerably to render contours in this manner, and it is not useful on lower end hardware or hardware from other manufacturers.

Another method, providing a significant new take on existing ideas from 3D shape rendering [3], was proposed in 2007 by Chris Green of Valve Software [4]: a distance transform representation of the contour is computed and stored as a texture, and a shader is used to generate the crisp edge. This allows a fully general, anisotropic analytic anti-aliasing of the edge, and the shader can be kept simple and fast. Combined with recent development in distance transforms [7], this method is very suitable for certain important kinds of surface textures, like alpha-masked outlines, text, decals and printed patterns. Until now, real time graphics has had an annoying tendency of making such patterns blurry or pixelated in close-up views, but contour rendering based on a distance transform can eliminate that and make edges crisp and clear in a very wide range of scales from minification to extreme magnifications. Moreover, the edges can be anti-aliased in an anisotropic, analytic manner. An example of this is presented in [5]. A pattern processed and rendered in that manner is shown in Figure 4.

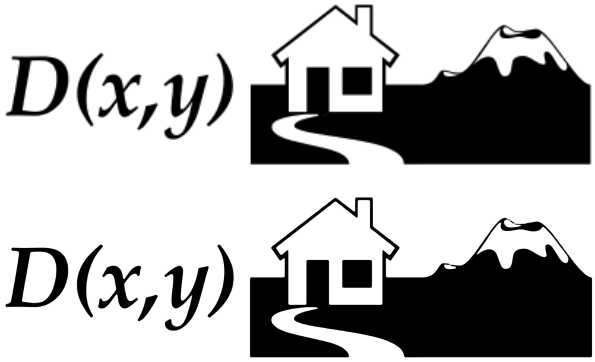


Figure 4: Shapes rendered by a distance transform. *Top:* bitmap texture used as input to the distance transform. *Bottom:* crisp, high resolution shapes rendered by the distance transform and a shader.

7 Anti-aliasing

The texture subsystem of a GPU has built-in mechanisms to handle anti-aliasing. Mipmapping has been part of OpenGL since its early days, and anisotropic mipmap filtering has been common for years. However, mipmapping only handles minification, and there is a limit to how much anisotropy it can handle. Procedural patterns can also fill in detail when the pattern is magnified. Edges remain crisp, and there is no pixelization. Finally, because the anti-aliasing is analytic in nature, a procedural surface pattern can be properly anti-aliased even in very oblique views with strong anisotropy in an arbitrary orientation. Anti-aliasing of procedural patterns is a problem that requires extra care in designing the shader, but it's not magic. Producers of off-line rendered graphics have a long tradition of anti-aliasing in shader programming, and the methods and concepts are directly applicable for real time use as well.

8 Displacement shaders

Procedural shaders in off-line production can not only set the properties of a surface point. Another important class of shaders is *displacement shaders*, which set the position of the surface point in relation to some original surface. Displacement mapping is the tangible concept that is simulated by bump mapping, normal mapping or parallax mapping. By performing a true displacement of the surface, a displacement shader can act as a modeling tool and add true geometric detail to a surface. Because the displacement shader is aware of the viewpoint and the view direction, it can adapt the level of detail to what is really needed, thus striking an important balance between performance and quality.

A powerful tool in off-line production is the combination of a displacement shader and a matching surface shader. By passing information about the position and orientation of the displaced surface from the displacement shader to the surface shader, very detailed and realistic surfaces can be created. Displacement shaders have been an important part of the toolbox for off-line production for a long time.

When GPU shading was introduced, vertex shaders did not allow for displacement of arbitrary surface points, only vertices. Thereby an important part of what displacement shaders can do was lost. To simulate a point by point displacement, dynamic tessellation to an appropriate resolution had to be performed on the CPU. While this has been done for things like real time terrain rendering, it is a process that is taxing for the CPU, and the tessellation needs to stop before the triangles become too small and too many for the geometry transfer from the CPU to the GPU to handle them at real time frame rates.

The recent introduction of hardware tessellation shaders changed that. Now, the GPU can execute a shader that dices the geometry into very small pieces in a view dependent manner, without even bothering the CPU. The GPU has a high internal bandwidth and can do this for many triangles in parallel. We have seen this being put to good use for adaptive tessellation of curved surfaces, finally making it possible to use bicubic patches, NURBS and subdivision surfaces as the native model description for real time content. This is a welcome thing by itself, as real time graphics has been somewhat hampered by the fixation on polygon modelling, while content for off-line rendering has been free to also use more flexible and adaptively tessellated curved surface primitives [2].

However, tessellation shaders can be used for more. They can bridge the gap between the comparably crude vertex shaders for hardware rendering and the much more versatile displacement shaders used in software rendering. GPUs are not quite yet at the point where they can easily dice all geometry in the scene to triangles the size of a single pixel, but we are rapidly getting there, and then the vertex shader stage will have the same capabilities as a traditional displacement shader for off-line rendering. The REYES rendering algorithm, which is the traditional scanline rendering method for Pixar's RenderMan, has a very hardware friendly structure because it originated as an algorithm for Pixar's hardware renderer RM-1. Simply put, it works by dicing all geometry to micropolygons that are about the projected size of a pixel, and executing shaders for each micropolygon. The time is now right to start doing similar things in GPU rendering.

9 Conclusion

We are used to dealing with real time graphics and off-line graphics in different ways, using different methods and different tricks to render the images we want. We have also been forced to accept a low image quality from real time graphics, incapable of matching even previous generations of off-line renderings. That gap is now closing. While some methods from software rendering, like the more advanced methods for global illumination, will probably remain out of reach for real time execution for quite some time longer, there are definitely possibilities to mimic in a real time setting what software renderers like RenderMan did ten years ago.

Current research and development in GPU rendering seems to be focusing on adding even more sophisticated illumination and surface reflectance models to the hardware rendering pipeline. The two concepts emphasized in this article, procedural patterns and displacement shaders, do not seem to attract an interest nearly as strong. That is a shame, as there are many fun and useful things to be done in that area. Looking further into this could have a significant impact on the overall perceived realism in real time graphics, possibly more so than current incremental improvements in illumination and reflectance. Real time illumination and surface reflectance models are now roughly at the point where off-line scanline rendered graphics was ten years ago. Geometry modelling and surface texturing, however, are still twenty years behind or more. To put it bluntly, real time graphics is basically still pasting pixels onto polyhedra like we did in the early 1990's, only at a much greater speed.

Imagine if in real time graphics applications we could eliminate pixelated textures in close-up views, add arbitrary surface variation programmatically to any texture, model geometry features down to a microscopic scale using displacement shaders, and stop worrying about polygon edges showing.

No more texels, no more facets!

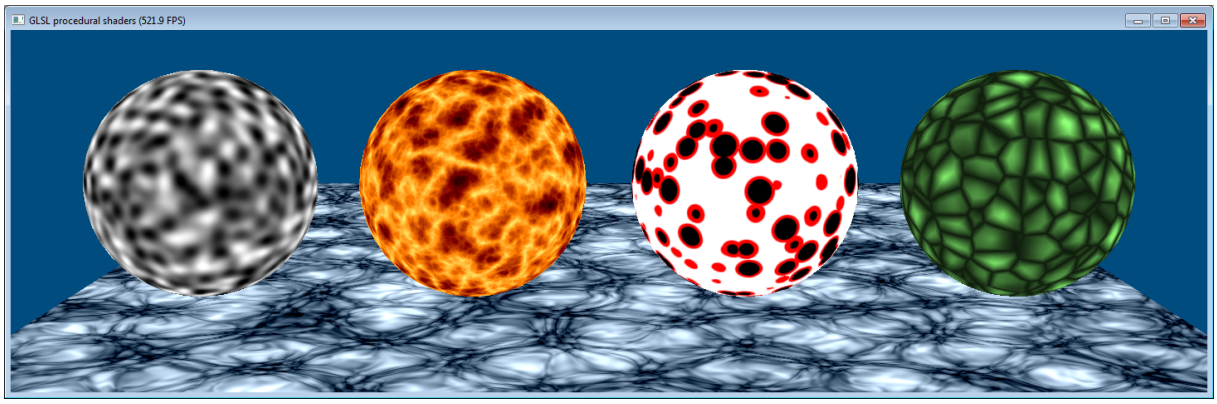


Figure 5: Some more complex noise-based procedural patterns in GLSL

References

[1] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2002.

[2] A. R. Fernandes and B. Oliveira. Gpu tessellation: We still have a LOD of terrain to cover. In P. Cozzi and C. Riccio, editors, *OpenGL Insights*, pages 145–162. CRC Press, 2012.

[3] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 249–254, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[4] C. Green. Improved alpha-tested magnification for vector textures and special effects. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 9–18, New York, NY, USA, 2007. ACM.

[5] S. Gustavson. 2D shape rendering by distance fields. In P. Cozzi and C. Riccio, editors, *OpenGL Insights*, pages 173–181. CRC Press, 2012.

[6] S. Gustavson. Procedural textures in GLSL. In P. Cozzi and C. Riccio, editors, *OpenGL Insights*, pages 105–120. CRC Press, 2012.

[7] S. Gustavson and R. Strand. Anti-aliased euclidean distance transform. *Pattern Recogn. Lett.*, 32(2):252–257, Jan. 2011.

[8] P. Hanrahan and J. Lawson. A language for shading and lighting calculations. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '90, pages 289–298, New York, NY, USA, 1990. ACM.

[9] I. McEwan, D. Sheets, M. Richardson, and S. Gustavson. Efficient computational noise in GLSL. *Journal of Graphics Tools*, 16(2):85–94, 2012.

[10] K. Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, July 1985.

[11] S. Upstill. *RenderMan Companion: A Programmer's Guide to Realistic Computer Graphics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[12] S. Worley. A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 291–294, New York, NY, USA, 1996. ACM.

Volume Sampling and Segmentation

SIGRAD 2013, pp. 49-54
T. Ropinski and J. Unger (Editors)

Poor Man's Rendering Of Segmented Data

Stefan Lindholm[1] and Alexander Bock[1]

[1]Scientific Visualization Group, Linköping University, Sweden

Abstract

In this paper we present a set of techniques for fast and efficient rendering of segmented data. Our approach utilizes the expected difference between two co-located texture lookups of a label volume, taken with different interpolation filters, as a feature boundary indicator. This allows us to achieve smooth class boundaries without needing to explicitly sample all eight neighbors in the label volume as is the case with previous methods. We also present a data encoding scheme that greatly simplifies transfer function construction.

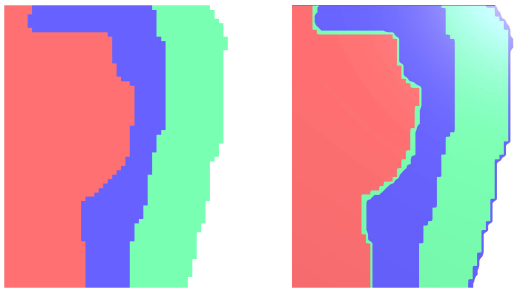
Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Picture/Image Generation—Antialiasing I.4.6 [Computer Graphics]: Segmentation—Pixel classification

1. Introduction

Rendering of segmented data is a core topic in the field of volume rendering. It is characterized in that it utilized external sources for the classification of grid points, rather than relying on user driven classification through, for example, a transfer function. By spending more resources on the classification as a pre-process step, much more reliable feature boundaries can be identified than what is possible through classification schemes applied at rendering time. Naturally, the topic of rendering segmented data is closely related to that of the actual segmentation. See [KDC*00, HJ04, UH00] for comprehensive overviews of the available literature. In this paper, we assume that a full segmentation has been performed such that the *source data* is complemented with a *label volume*, i.e. an integer volume of the same dimensionality of the source data with a single per-voxel label denoting the class membership (material) of the voxel.

The basic idea behind rendering segmented data is identical to standard volume rendering: to select visual parameters based on the class membership of each sample. This is greatly simplified since the required class membership can be accessed directly though the label volume. Therefore, the visual properties are often defined on a per-class basis, rather than in a single global definition for the entire data. The complexity of the per-class visual properties vary depending on what the situation requires and can span from a unique color to a fully defined class-specific transfer function. In this paper we use a single transfer function per class but apply the same shading scheme to all classes, however, it should be noted that this is a stylistic choice rather than a requirement of the approach.

A problem that arises when rendering segmented data is that the basic approach to 'just sample the label volume' to extract class memberships is not as straight forward as one might think. The simplest solution is, of course, to apply nearest neighbor sampling, which is guaranteed to return a valid class membership. Unfortunately, this leads to boundaries with a very blocky appearance (see Figure 1(a)). The membership operation is said to have *voxel-resolution*. The most intuitive way to achieve a higher, *pixel-resolution*, membership operation is naturally to allow for interpolation when sampling the label volume. This is, however, not guaranteed to return a valid class label unless there is no more than two classes in the entire volume. For example, an interpolation in a boundary area between class 3 and class 5 would return a class label of 4 even if this was not existent at that particular location in the data (see Figure 1(b)). An approach to achieve a pixel-resolution membership operation for segmented data, called two-level volume rendering, was presented in [HBH03]. In this approach, all eight neighboring grid points in the label volume are sampled using nearest neighbor interpolation in order to acquire the information necessary to remap the 3–5 operation to a 0–1 range and thereby avoid illegal interpolations. The result is much smoother boundary representations that are more visually pleasing. Unfortunately, the method requires an additional *seven* texture lookups per step along the ray which is many times unfeasible.



(a) Nearest neighbor (b) Linear

Figure 1: Using the two native interpolation kernels alone will result in undesired results. Nearest neighbor interpolation (a) will create a blocky result, visually unpleasing. Linear interpolation (b) will produce misclassifications between labels (see I and II) as the labels are integer values, and naïve linear interpolation will create a linear transition between the source values.

In this paper we present a novel method that delivers results comparable to pixel-resolution schemes while only requiring a *single* extra texture lookup. The core of our approach is to sample the level volume twice at the same location, with and without interpolation (i.e, with nearest neighbor vs. linear filtering). The difference between the two values is then used to smoothen the visual representation near class boundaries without creating incorrectly interpolated class values. We show that our method produces results visually comparable to methods that utilize full neighborhood sampling while requiring a factor of $7:1$ less additional texture lookups. Additionally, we also present a data compression approach that removes the need to have the label volume accessible during rendering.

2. Poor Man's Rendering With Label Volume

The implementation will be presented in three steps. First we provide a few brief details on how to sample a single texture with different interpolation filters in OpenGL. We then present how to compute the actual class membership of each sample as well as the attenuation parameter t that is used to smoothen the boundaries.

Nearest Neighbor and Linear Sampling

As previously noted, the implementation relies on sampling the same volume twice with different interpolation filters. For many architectures, however, a single texture can only be associated with a single sampling mode. One way to get around this restriction is to always force the label volume to be associated with linear interpolation. The nearest neighbor sample can then be accessed by adding an offset that forces the sample to be taken at the center of the nearest voxel. The nearest grid point can be accessed as follows

$$s_{\text{near}} = \text{floor}\big(s + vec3(0.5)\big) \tag{1}$$

where s is the original sample position along the ray and s_{lin} is the position of the nearest neighbor. For OpenGL/GLSL both samples can then be accessed using `texture3D`. Note, that OpenGL uses voxel centric values, as opposed to grid centric values, for `texture3D` and that the sample positions needs to be mapped accordingly. Alternatively. the nearest neighbor sampling can be achieved using `texelFetch` which should not require any additional mapping of the sampling point.

Once the linear and nearest sample points have been computed, they are used to extract the linear and nearest values from the label volume, l_{lin} and l_{near} respectively.

Class Membership and Class Attenuation

In our approach the class membership is directly decided by the nearest neighbor sample in the label volume, l_{lin}. This means that membership itself is defined at voxel-resolution. To achieve smoother class boundaries we introduce a *class attenuation parameter*, t.

The attenuation parameter is computed from the nearest and linear label values

$$t = \text{abs}(l_{\text{lin}} - l_{\text{near}}). \tag{2}$$

In Equation 2, we see that $t = 0$ as long as the interpolated value is identical to the nearest neighbor value. This will be the case as long as the local neighborhood around the sample point only contains a single class. We can also note that if the local neighborhood contains two or more classes, then $t > 0$. The class attenuation parameter thereby functions as a boundary indicator.

Interpreting the attenuation parameter as a boundary indicator, we use it to adjust the opacity of the current sample. Ideally, we want full opacity when the samples are fully inside a single class and zero opacity when samples are half way in between two (or more) classes. Unfortunately, this requires us to know all classes that affect the interpolated value, which in turn would require us to sample the entire neighborhood in the label volume (which we don't want to do). Instead, we always map the opacity to zero as t reaches 0.5

$$\alpha' = (1 - (2t)^2)\,\alpha. \tag{3}$$

The full process is described in Algorithm 1 and illustrated in Figure 2.

While Equation 3 largely achieves the desired effect, it is important to note that it introduces a few predictable irregularities. First, $t = 0.5$ only corresponds to the half way mark between two labels if their numerical labels are separated by a single unit (e.g. 1–2, 3–2). For all other cases, the opacity will reach zero before the half way mark. Second,

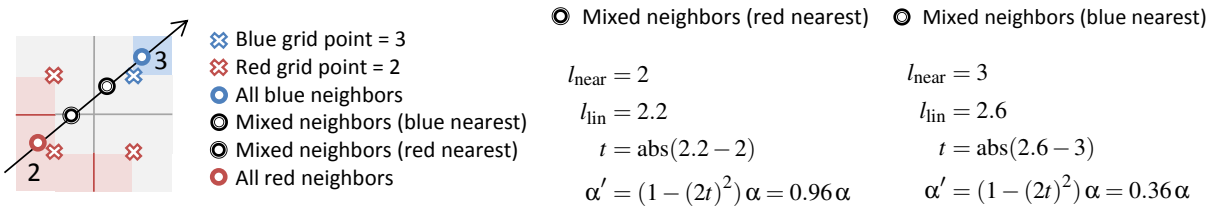


Figure 2: Example computations of ray contributions. The visible ray segment starts inside the red feature (class id 2), and ends inside the blue feature (class id 3). The class membership of the two samples taken in between the features needs to be decided. In our approach, their membership is fully determined by a nearest neighbor query in the label volume (here resulting in $l_{\text{near}} = 2$ and $l_{\text{near}} = 3$ respectively). However, in order to achieve smooth boundaries at pixel-resolution we employ an alpha modulation based on the difference between the the nearest query and another query using linear interpolation (here resulting in $l_{\text{near}} = 2.2$ and $l_{\text{near}} = 2.6$ respectively).

Algorithm 1 *Poor Man's Rendering* ***with*** *Label Volume*

Require:
A source volume V_{src} and a label volume V_L,
both present on the GPU
A set of labels $\{l_1, l_2, \cdots, l_n\}$
A transfer function, TF_i for each label l_i
Algorithm:
for all sample points along ray **do**
◇ Sample V_L using *nearest neighbor* interp. $\rightarrow l_{\text{near}}$
◇ Sample V_L using *linear* interpolation $\rightarrow l_{\text{lin}}$
◇ Compute attenuation parameter t according to Eq. 2
◇ Compute alpha modulation according to Eq. 3
◇ Sample V_{src} using *linear* interpolation $\rightarrow v_{\text{lin}}$
◇ Select a TF based on l_{near}
◇ Evaluate $\text{TF}_{\text{near}}(v_{\text{lin}})$
◇ Apply alpha modulation to output
◇ Composite output to result
end for

in some rare corner cases, the nearest neighbor value may change before the opacity has reached zero, effectively creating a sharper than intended cutoff. In short, we have less fine-tuned control over the opacity behavior across boundary regions.

What makes Poor Man's Rendering such an attractive trade-off is that the visual impact of the expected irregularities is near insignificant and thus acceptable given the increase in performance compared to the full neighborhood analysis.

3. Poor Man's Rendering Without Label Volume

In this section we present a data encoding scheme that eliminates the need to upload the label volume to the GPU. The encoding relies on a set of linear mappings (one for each class) from the source volume to an encoded target volume. The key here is to ensure that the co-domain of the mappings each span a unique value range in the target volume.

For example, if we have three classes which voxels exhibit overlapping value ranges (0.3–0.5, 0.2–0.7 and 0.1–0.5). We then map the value ranges of these classes to three unique value ranges in the target volume

$$l_1 : [0.3, 0.5] \mapsto [0.1, 0.3]$$
$$l_2 : [0.2, 0.7] \mapsto [0.4, 0.6]$$
$$l_3 : [0.1, 0.5] \mapsto [0.7, 0.9].$$

This mapping makes it possible to determine the class of a sample solely based on which unique range it belongs to. Note that interpolation in this volume can still lead to invalid results. The data encoding is performed as a pre-process step and the encoded volume replaces the source volume on the GPU. Meta information from the mappings also needs to be uploaded in order to decode the volume during rendering. The decoding process is simply the inverse of the linear mapping for each class.

The computation of the attenuation parameter t without a label volume works as follows. Two samples are taken from the encoded volume, with nearest neighbor (e_{near}) and linear interpolation (e_{lin}) respectively. First, the nearest neighbor sample is used to identify the label and its valid value range (from the uploaded mapping information)

$$e_{\text{near}} \rightarrow l_{\text{near}},\ [e_{\text{min}}, e_{\text{max}}]. \tag{4}$$

Based on this the attenuation parameter t can be computed as

$$t = \text{abs}(v_{\text{lin}} - \frac{e_{\text{max}} + e_{\text{min}}}{2}) - \frac{e_{\text{max}} - e_{\text{min}}}{2} \tag{5}$$

clamped to the 0–1 range. Poor Man's Rendering can now be performed using Equations 5 and 3. The full process is described in Algorithm 2.

A positive side effect to the encoding is that the user can now specify a single global transfer function without labels while still maintaining separate visual properties for the different classes (since the value ranges are known to be non-overlapping). This is particularly beneficial for systems that are not previously set up to load and render segmented data

Algorithm 2 *Poor Man's Rendering* ***without*** *Label Volume*

Require:
A source volume V_{src} and a label volume V_L
A target volume for the encoding V_{enc}
A set of labels $\{l_1, l_2, \cdots, l_n\}$
A transfer function, TF_i for each label l_i

Pre-process:
for all labels l_i **do**
◇ Find the value range $[r_{\min}, r_{\max}]$ covered by by the voxels of class i in V_{src}
◇ Assign a unique range $[e_{\min}, e_{\max}]$ where $e_{\min} = 0.1 + 0.3i$ and $e_{\max} = 0.2 + 0.3i$ in V_{enc}
◇ Linearly map all voxels of class i from $V_{\text{src}} \mapsto V_{\text{enc}}$ as $[r_{\min}, r_{\max}] \mapsto [e_{\min}, e_{\max}]$
◇ Save mapping information as M_i
end for
◇ Upload V_t to GPU together with meta information of each class mapping

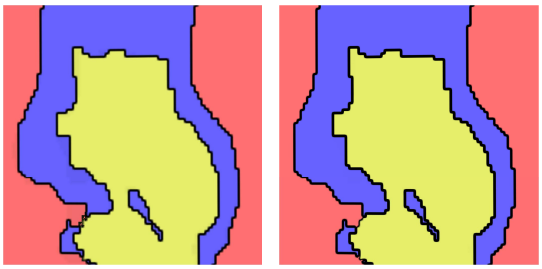
Algorithm:
for all sample points along ray **do**
◇ Sample V_{enc} using *nearest neighbor* interp. $\rightarrow e_{\text{near}}$
◇ Identify current label l_{near} and valid value range $[e_{\min}, e_{\max}]$ based on which unique range that contains e_{near}
◇ Sample V_{enc} using *linear* interpolation $\rightarrow e_{\text{lin}}$
◇ Compute attenuation parameter t according to Eq. 5
◇ Compute alpha modulation according to Eq. 3
◇ Compute the inverse mapping $e_{\text{lin}} \mapsto v_{\text{lin}}$ using M_j
◇ Evaluate $\text{TF}(v_{\text{lin}})$
◇ Apply alpha modulation to output
◇ Composite output to result
end for

or multiple volumes. A potential issue with this type of encoding is the loss of precision that follows from the remapping. Since this naturally becomes a situational trade-off a complete analysis is out of the scope of this paper. What can be said is that as the more narrow the value ranges are per class in the source volume, the less precision will be lost.

3.1. Results

In this section we present the results that we achieve using our two proposed techniques. The real world dataset we use is the Walnut with its accompanying segmentation. For the comparisons we have applied alpha attenuation in boundary regions also for two-level volume rendering although this was not a part of the original presentation.

Figure 5 shows a comparison between our implementation of the two-level volume rendering to our method both in the cases where a label volume is present and in the case of the encoded volume. The test images were created using a small stack of slices of the Walnut dataset with each label value assign to a unique, fully opaque color. It is clearly visible that both interpolation kernels produce a superior visual quality when compared with the nearest neighbor interpolation. However, despite minor (expected) visual artifacts (see lower left corner in 5(c)) our method performs reasonably well and achieves a comparable result to the method employing a full neighborhood search for each segment. Figure 3 shows the two-level volume rendering and the our method applied to a single slice of the volume. Note how the background color is visible between feature boundaries as the opacity of the samples is reduced by the attenuation factor t.



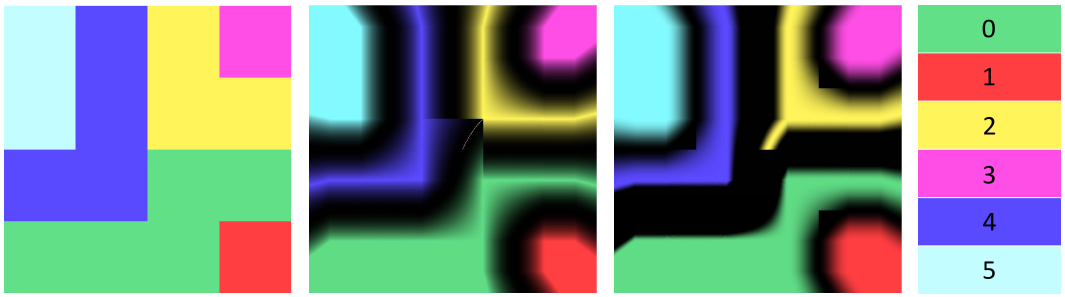
(a) Two-Level Volume Rendering (b) Poor Man's Rendering

Figure 3: Comparing a single slice off the Walnut dataset. In the areas between labeled regions, the opacity of the single sample is lowered due to the attenuation factor t and therefore the background color is visible. Note that the thickness of the boundary region in our method is depending on the label values, whereas it is normalized in the two-level volume rendering approach. This is visible in (b) as the width between the violet-red boundary is thinner than the violet-yellow boundary.

Figure 6 is a comparison of the walnut dataset showing the different structures within the walnut. The visual result for each segment is determined by its own, separate 1-dimensional transfer function. Figure 6(a) and 6(b) are rendered with the same rendering parameters and the differences between the two techniques are shown in Figures 6(c) and 6(d) with the difference magnified by a factor of 10. As expected, the bulk of difference is in the region around the center, where a segment with label value 4 is adjacent to the gray "air" surrounding it with a label value of 1. This means that the transition in our method is much sharper than in the reference image.

3.2. Discussion

In this paper we have presented an approach to achieve smooth boundary transitions when rendering segmented data while significantly reducing the number of necessary texture lookups compared to methods based on full neighborhood analysis. The approach drastically reduces the amount of samples necessary to achieve the boundary smoothness,



(a) Nearest Neighbor (b) Two-Level Volume Rendering (c) Poor Man's Rendering (d) Intensites

Figure 4: Rendering a synthetic dataset with the Two-Level Volume Rendering (b) and the Poor Man's Segmentation (c) compared to the nearest neighbor filtering (a). (d) shows the values and the ordering of the intensity values for this dataset. As expected, the Two-Level Volume Rendering produces uniform boundaries regardless value difference. In the Poor Man's rendering, the boundary regions are dependent on the intensity value difference of the features. Both methods show a visual artefact in the area where three features coincide and the linear interpolation of green and blue results in yellow.

but does so at the cost introducing minor irregularities. The artifacts introduced by Poor Mans Rendering has two main visual manifestations. The first is an apparent widening of the transitional region between two segments. This behavior is predictable and can be minimized by intelligent selection of segment labels. For example, if a volume contains three segments but two of the segments never overlap, then the numerical label values can be selected such that the difference between two neighboring values never exceeds one, in which case any widening will be prevented. The second visual artifact is a non-smooth transition in neighborhoods heavily dominated by a single segment, such as the bottom of a depression. In such cases, the attenuation parameter is not guaranteed to reach zero before the nearest neighbor switches, resulting in a less smooth transition. It is possible to lower the impact of this artifact is to apply a stronger mapping in Equation 3.

In terms of performance, the value of the presented methods depends on a set of circumstances, some more predictable than others. For example, an application that is heavily bottlenecked by texture lookups is far more likely to see significant speedups than an application that spends most of its time of computations. On the other hand, the impact of the introduced irregularities will vary between different lighting and transfer function combinations which makes the assessment on the speed-vs-performance trade-off very much case dependent.

We believe our approach fills the gap between nearest neighbor class assignment and methods that rely on full neighborhood analysis. It could potentially be useful in cases where acceptable framerates have a high priority, such as previewing large datasets, interaction and transfer function design. The option to encode the segment membership together with the source data should also simplify rendering of segmented volume data on systems that may only have a single global transfer function.

3.3. Acknowledgments

David Karlsson, technical director at Interactive Institute, for pointing out the gap in the literature that this method fills. The presented concepts have been realized using the Voreen framework (www.voreen.org).

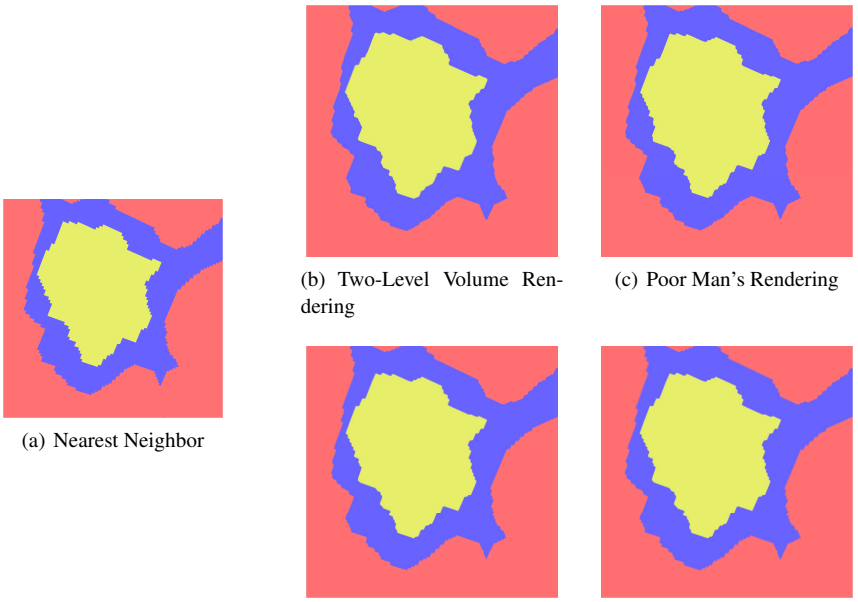
References

[HBH03] HADWIGER M., BERGER C., HAUSER H.: High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of the conference on Visualization 2003* (2003), pp. 301–308. 1

[HJ04] HANSEN C. D., JOHNSON C. R.: *Visualization Handbook*, 1 ed. Academic Press, 2004. 1

[KDC*00] KAUFMAN A., DACHILLE F., CHEN B., BITTER I., KREEGER K., ZHANG N., TANG Q., HUA H.: Real-time volume rendering. *International Journal of Imaging Systems and Technology 11*, 1 (2000), 44–52. 1

[UH00] UDUPA J., HERMAN G.: *Three D Imaging in Medicine*. CRC PressINC, 2000. 1



(a) Nearest Neighbor

(b) Two-Level Volume Rendering

(c) Poor Man's Rendering

(d) Two-Level Volume Rendering without Label Volume

(e) Poor Man's Rendering without Label Volume

Figure 5: Comparative results of the presented techniques. (b) and (d) show the Poor Man's Rendering with and without a label volume respectively, while (a) and (c) are rendered using the Two-Level Volume Rendering approach. Note that while the both the Poor Man's Rendering and the Two-Level approach produce much smoother and visually pleasing results than the nearest neighbor filtering in (e), the differences between them is minimal.

(a) Two-Level Volume Rendering

(b) Poor Man's Rendering

(c) Difference Image

(d) Difference Image (x10)

Figure 6: Rendering the segmented Walnut dataset with our proposed method. (a) shows the reference image, created using the two-level volume rendering, while (b) is rendered using Poor Man's Rendering. (c) shows the absolute pixel-wise difference images between the two results and in (d) this difference is enhanced by one order of magnitude. As expected, the techniques differ only in the areas, where features touch whose label values differ by more than unity.

SIGRAD 2013, pp. 55-60
T. Ropinski and J. Unger (Editors)

Towards Data Centric Sampling for Volume Rendering

Stefan Lindholm[1], Daniel Jönsson[1], Hans Knutsson[2] and Anders Ynnerman[1]

[1]Scientific Visualization Group, Linköping University, Sweden
[2]IMT, Linköping University, Sweden

Abstract

We present a new method for sampling the volume rendering integral in volume raycasting where samples are correlated based on transfer function content and data set values. This has two major advantages. First, visual artifacts stemming from structured noise, such as wood grain, can be reduced. Second, we will show that the volume data does no longer need to be available during the rendering phase; a surface representation is used instead, which opens up ample opportunities for rendering of large data. We will show that the proposed sampling method gives higher quality renderings with fewer samples when compared to regular sampling in the spatial domain.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Bitmap and framebuffer operations I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types

1. Introduction

Volume rendering is an essential and widespread technique used to explore volumetric data. Its use in areas such as medicine and engineering puts a demand on accurate visualizations while still allowing interactive exploration of the data. The majority of the techniques related to volume rendering use some form of emission/absorption model of light transport. The traditional computational approach involves a discrete Riemann sum, where the volume is sampled along viewing rays (emanating from the screen out through the scene), resulting in an iterative solution. It is of great importance that this model can be evaluated in an accurate and fast manner. An important aspect of the traditional approach is that the sample positions of neighboring rays are selected in the spatial domain with next to zero correlation in attribute space between neighboring rays. This paper investigates an alternate way to select samples to accurately and efficiently evaluate an emission/absorption model.

We propose a scheme that correlates samples in attribute space across all rays in the image. The effect is that each sample represents a given *change in function value* as opposed to a certain spatial distance. Our scheme creates correlated samples for all rays simultaneously (an operation of object space complexity) while still allowing the emission/absorption model to be evaluated independently for each ray (an operation of image space complexity). We show how the scheme is inherently data adaptive and how the content of the transfer function can be used to optimize the selection of the correlated samples. An illustration of the difference relative to the traditional approach is provided in Figure 1.

The paper is structured as follows. First we recap the popular optical model of volume rendering with an emphasis on the discretization that it typically leads to. We then outline some of the challenges, and their related work, that arise from such a discretization, both for the general case and within the context of visualization. Finally we present our approach.

2. Background And Related Work

The arguably most popular physical model of volume rendering is derived from optical models of light transport through participating mediums, as introduced by Max in [Max95]. The participating media is here modeled as "clouds" of small particles, where light that passes through such media extinguishes at a rate proportional to the number of particles per volume fraction (particle density). The continuous differential equation describing emission and absorption is the following

$$\frac{dI}{ds} = c(s)\tau(s) - \tau(s)I(s), \tag{1}$$

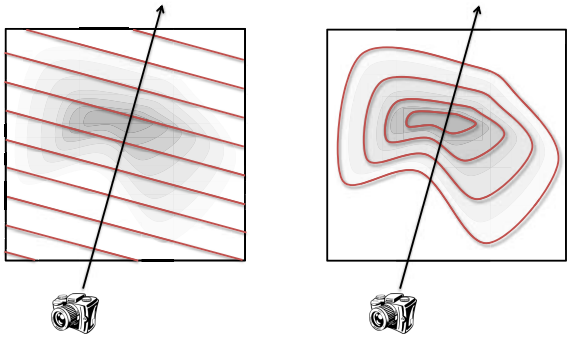


Figure 1: *Left:* Spatially correlated sampling used in standard volume rendering . *Right:* Radiometrically correlated sampling used in our ISO based volume rendering.

where s is a length parameter along a ray, $c(s)$ is the amount of light emitted, $\tau(s)$ is the amount of light absorbed and $I(s)$ is the light intensity. The solution to this differential equation is the well known *volume rendering integral* (VRI)

$$I(D) = I_0 e^{-\int_0^D \tau(t)\,\mathrm{d}t} + \int_0^D c(s)\tau(s)e^{-\int_s^D \tau(t)\,\mathrm{d}t}\,\mathrm{d}s. \quad (2)$$

The first term describes the amount of light coming from the end of the ray attenuated by the absorption in the volume. The second term describes light that is emitted and absorbed along the ray within the volume. The past twenty years have seen a large amount of work dedicated to the integral in Equation 2. A practical analytical closed-form solution has, unfortunately, proven to be elusive as such solutions tend to rely on the *erf* function, which is rather costly to evaluate. As a result, most of the literature concerns discrete numerical solutions, most notably Riemann sums of piecewise constant segments.

If the color and opacity each are assumed to be constant for each segment i, $c(s) \approx c_i$ and $\tau(s) \approx \tau_i$, then the continuous expression in Equation 2 can be approximated as

$$I(D) \approx \sum_{i=0}^{n} c_i \tau_i ds_i \prod_{j=i+1}^{n} 1 - \tau_j ds_j. \quad (3)$$

where each segment corresponds to a sample along the viewing ray. The question is how to place the samples in such a way that the VRI can be accurately and efficiently evaluated.

We will now briefly discuss the general problem of discretizing and solving Equation 2 in relation to related work on two aspects, numerical and perceptual. Many of these works address two main questions:

Where to sample? It is desirable to have the highest possible quality given as few samples as possible without overhead.

How many samples? It is desirable to determine the lowest number of samples needed to achieve an error that is not visible to the user.

The general problem: Approximation errors

Numerical approximations of an integral most often introduce errors. One way of minimizing such an error is to apply importance based sampling. Much work has been done to come up with good error metrics and even more work has been done to minimize the errors. Notable sub fields on this topic include Monte Carlo based schemes [Sha03], adaptive sampling techniques [Lev90], level of detail rendering [WWH*00] and predictive schemes [NH93].

The visualization problem: Structured noise

Differences in the approximation errors across pixels leads to noise in the rendered image. Unfortunately this noise is structured and often results in noticeable artifacts. One of the most prominent in the field of volume rendering is the so called wood grain effect. Several techniques have tackled this problem. The most notable being jittering [EHK*06], which randomly moves the start point of the ray to alleviate the visual appearance by making noise unstructured. Another partial solution is pre-integrated volume rendering, which in many cases can remove the artifacts [KE04] by implicitly super-sampling the ray. Similarly, additional samples can be enforced to correlate with sharp features in the transfer function [KHW*09] at the cost of solving a third degree polynomial between samples. Another approach to reduce visual artifacts is to include perceptual based sampling in the image plane using iterative methods and models of the human visual system [BM98].

3. A Coherent Sampling Method

Our new sampling method utilizes two important aspects of the VRI and human visual system, that a user specifies absorption through a transfer function and that structured noise is visually disturbing. We use these two realizations when answering the two questions in the previous section, where to sample and how many samples to take. We previously discussed that errors across pixels leads to visually disturbing artifacts. This is especially noticeable at sharp boundary transitions, where two consecutive samples along a ray have low and high opacity. This produces a high numerical error that becomes visually apparent because neighboring rays are likely to also have a high numerical error.

We reduce the structured noise by allowing a maximum value change between each pair of samples. This naturally leads to more samples in transitional regions and fewer samples in homogeneous regions. The key here is that the values at which samples are enforced to are global. As such, each layer of samples corresponds to an iso-surface. The surfaces are based on an importance sampling of the transfer function content as illustrated in Figure 2. Surfaces are extracted from the non-zero opacity regions in the transfer function and we are thus retrieving visually important samples that have a high impact on the result of the VRI. Furthermore,

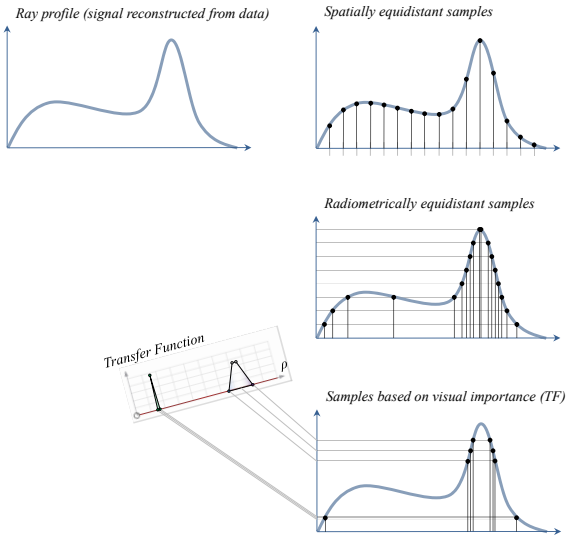


Figure 2: *Top Left:* Ray profile. *Top Right:* Samples uniformly distributed in the spatial domain (horizontal axis). *Middle Right:* Samples uniformly distributed in the attribute space (vertical axis). *Bottom Right:* Samples based on visual importance.

by extracting the surfaces at globally selected iso-values, we guarantee that neighboring rays in image space will sample the VRI in a coherent manner, and therefore also remove the wood grain artifact. Another benefit is that no samples will be placed in regions with zero opacity resulting in an efficient empty space skipping.

The second question is how to know how many samples need to be taken in order to ensure an artifact free image. Here we can utilize the knowledge that we can bound the error since the distances in both the spatial and data domain are known between each sample pair. That is, since we have generated surfaces based on values in the data domain, we know that the data variation between two surfaces are bounded by the values used to generate the surfaces. This information can be combined with the distance between the two surfaces in order to derive a maximum approximation error and thus continue to sub sample along the ray until going below the desired error. This particular aspect of the method is, however, not included in the scope of this paper, which focuses more on the former question.

We further introduce a novel data representation that allow the VRI to be evaluated without access to the original volume data. This is based on forming an assumption on how the data varies between each pair of sample points, i.e. between two surfaces. We use the assumption that the data, and thereby the transfer function response, varies linearly.

3.1. Implementation

Algorithm 1 *Coherent Sampling.*

Require:
A set of iso-values formed from the transfer function content.
Algorithm:
for all iso-values **do**
Generate proxy geometry (marching cubes)
end for
Setup per pixel linked list
for all Proxy geometries **do**
Render proxy geometry
Store entry, exit point and iso-value per pixel
end for
Sort list of per pixel fragments based on depth
Evaluate VRI using sorted fragment list as sample sequence

The overall structure of our method is described in Algorithm 1 and Figure 3. The implementation can be divided into two main parts:

A Extraction of proxy geometry, for each iso-surface, in the form of an closed manifold triangle mesh. This step is performed in object space, potentially as a pre-process.
B Management of fragments from rasterized iso-surfaces, in the form of a sorted linked list intersection points for each pixel. This step is performed in image space during rendering.

For the first part, which takes place *before* rasterization, we use a hardware implementation of the *marching cubes* algorithm, performed as a three step process:

A.1 Use *instanced rendering* to trigger a single vertex per voxel in the volume
A.2 Use a *geometry shader* to create the maching cubes triangles
A.3 Use *transform feedback* to capture the output of the geometry shader into a buffer on the GPU

For the third part, which takes place *after* rasterization, we use a GPU variant of linked lists based on A-buffers

B.1 Render the mesh representations for all iso-surfaces (to be rasterized)
B.2 Store all fragments on a per-pixel basis as linked lists in an A-buffer
B.3 Sort each linked list based on fragment depth

After the execution of B, each list of fragments can be interpreted as an ordered list of iso-surface intersections. For volume rendering, this means that the volume rendering integral can be evaluated along each ray' by looping over the list of fragments, treating each fragment as a sample. In this case, sample positions are given by the direction of the ray and the depth of each fragment, while the corresponding sample values are implicitly given by the particular iso-surface associated with each fragment.

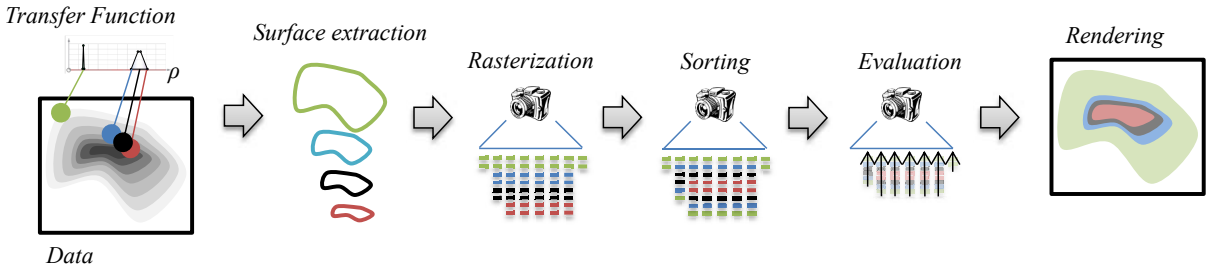


Figure 3: Our proposed solution which relies on extraction, rasterization and sorting of proxy geometry to generate samples for the evaluation of the volume rendering integral.

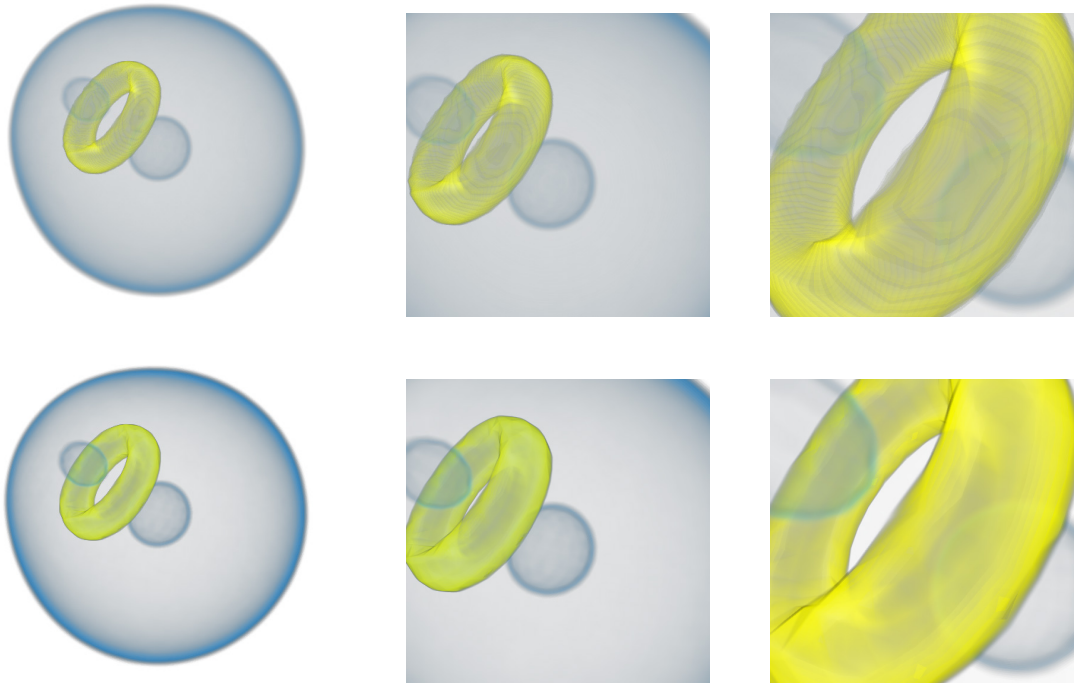


Figure 4: *Top row:* DVR of the Nucleon data set at an average of 4 samples per voxel. *Bottom row:* ISO at 26 surfaces for a total of 132k triangles. Using orthographic projection and looking from one side uniform sampling requires 164 samples per ray while our method only requires on average 30 samples per ray and still result in superior image quality.

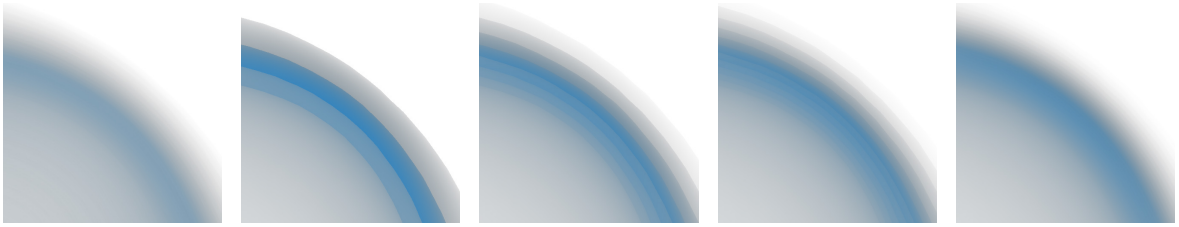


Figure 5: *Left-to-right:* DVR reference rendering of the Nucleon data set followed by ours with 5, 9, 13 and 31 surfaces (46k, 82k, 119k, 282k triangles respectively). The ISO rendering converges towards the reference with increasing number of surfaces. Convergence rate is directly dependent on transfer function width, the wider the transfer function, the more surfaces are needed for convergence.

4. Results

To evaluate the performance of the approach, we have used the a data set of a nucleon at a resolution of $41 \times 41 \times 41$ voxels. Figure 4 shows a comparison between our approach (*bottom row*) and standard volume rendering with samples distributed uniformly in the spatial domain (*top row*). The three images per row shows increasing magnification from left to right. The transfer function used consists of two separate, triangle shaped primitives (thus a total of four linear segments, see Figure 6 (top)). The image rendered with our approach utilizes one surface per segment end point in the transfer function (6 in total) with an additional 5 surfaces per linear slope (20 in total). The following table is an estimation of the number of samples used per ray (note that this number varies across the image)

White area No intersections and thus zero samples.
Blue area Intersections with 2×13 surfaces (in and out of the blue primitive).
Yellow area Intersections with $(2+2) \times 13$ surfaces (in and out of both the blue and the yellow primitives).
Extreme areas At a few locations rays pass in and out twice for each of the yellow and blue areas for a total of 8×13 surface intersections.

Note that for the majority of the rays, the average number of intersections, and thus samples that need to be evaluated and fragments that need to be sorted, is less than 30. Even the worst case scenario for this particular data and transfer function results in just above 100 samples per ray. The standard rendering was performed using an (above) average four samples per voxel for a total cost of 164 samples per ray, and still contains clearly visible wood grain artifacts.

We have also evaluated the amount of iso-surfaces needed to approximate a single linear segment of the transfer function. Figure 5 shows the same nucleon data set rendered with a single triangle primitive (two linear segments, see Figure 6 (bottom)). As the primitive is much wider, covering nearly half the data range of the volume, the spatial distance between the surfaces will be larger and therefore more surfaces are required to minimize the error than in the previous example. The most prominent visual artifact effect occurs at the edges of structures where rays that either intersect or miss a surface get different evaluations, effectively creating a visual step. More surfaces naturally makes the steps smaller such that they eventually disappear. The leftmost image was rendered with standard volume rendering at a very high sampling rate while the remaining images were rendered with the proposed method using increasing number of surfaces per transfer function segment. Note that the effects are magnified by the zoomed in camera setting and that less surfaces are necessary while viewing the entirety of the object.

Finally, derivative expressions can be included in the generation of the proxy geometry and made available during rendering. This enables surface based gradients to be used instead of volumetrically based gradients.

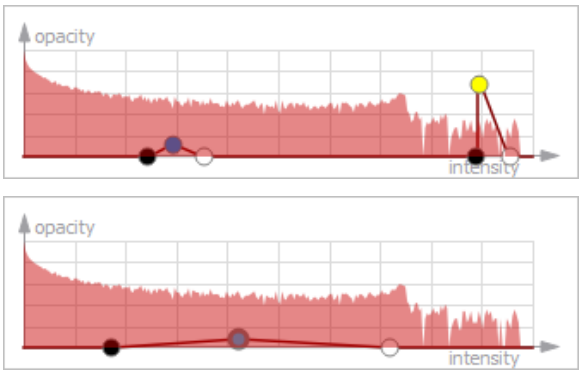


Figure 6: Transfer functions for Figure 4 *(top)* and Figure 5 *(bottom)* respectively.

5. Discussion and Future Work

This novel sampling approach and data representation opens up several opportunities. One implication of using the approach described in this paper is that the source data is not required to reside on the GPU during rendering. Since all "samples" are effectively acquired during the extraction of the iso-surfaces, no additional sampling needs to be performed during rendering. This naturally has both advantages:

+ Source data is not needed during rendering (zero texture fetches).
+ Very efficient data compression for visually sparse data (i.e, that the amount of visible voxels is small relative the size of the data).
+ Transfer functions no longer need to be evaluated during rendering as they will always be "sampled" at the value of the iso-surfaces.
+ The method is limited by the number of necessary samples, not directly by the number of voxels of the source volume.
+ Samples are coherent across pixels which effectively prevents the wood grain effect.

and disadvantages:

− Changing the transfer function means re-creating or re-uploading proxy geometry.
− Neighboring pixels can have vastly different evaluations due to tangent problem.
− Proxy meshes can be large due to granularity.
− Proxy meshes can be large due to an arbitrary number of shells for each mesh.
− Proxy meshes are inefficiently stored due to implicit data structure (could be mitigated by additional processing in the CPU).

Overall, the method has high potential due to the fact that its complexity seems to scale with information content rather than volume size. The fact that it contains a built in mech-

anism for importance driven rendering without much overhead in terms of specific management of LODs is also a great bonus. On the other hand, the method is somewhat sensitive to mechanism that increase the number of triangles that need to be generated. As such, it will probably be less suited for data that contain high amounts of noise, data that is uniform in nature, or where the visible data covers most of the volume. There is also great potential with respect to transfer functions as the transfer function will always, and only, be "sampled" at the values of the iso-surfaces. In fact, during rendering, the transfer function can be reduced to an array of pre-sampled values that is accessed without interpolation.

It is hard to make generalizations for the performance of the method as the computational complexity is highly dependent on the particular data and use case. Our empirical experience so far is that the rendering speed is up to par with other implementations for data that is sufficiently sparse, and then often with better visualization quality. In the future we will make a more thorough analysis of the individual parts of the method, including their limits and their sweetspots. We will also continue to work on the problem of potentially large meshes and will also investigate the possibilities to compute bounded errors during rendering.

References

[BM98] BOLIN M. R., MEYER G. W.: A perceptually based adaptive sampling algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 299–309. 2

[EHK*06] ENGEL K., HADWIGER M., KNISS J., REZK-SALAMA C., WEISKOPF D.: *Real Time Volume Graphics*. Ak Peters Series. A K Peters, Limited, 2006. 2

[KE04] KRAUS M., ERTL T.: Pre-integrated volume rendering. *The Visualization Handbook 2* (2004), 3. 2

[KHW*09] KNOLL A., HIJAZI Y., WESTERTEIGER R., SCHOTT M., HANSEN C., HAGEN H.: Volume ray casting with peak finding and differential sampling. *Visualization and Computer Graphics, IEEE Transactions on 15*, 6 (2009), 1571–1578. 2

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

[Max95] MAX N.: Optical models for direct volume rendering. *Visualization and Computer Graphics, IEEE Transactions on 1*, 2 (jun 1995), 99 –108. 1

[NH93] NING P., HESSELINK L.: Fast volume rendering of compressed data. In *Visualization, 1993. Visualization'93, Proceedings., IEEE Conference on* (1993), IEEE, pp. 11–18. 2

[Sha03] SHAPIRO A.: Monte carlo sampling methods. *Handbooks in operations research and management science 10* (2003), 353–426. 2

[WWH*00] WEILER M., WESTERMANN R., HANSEN C., ZIMMERMANN K., ERTL T.: Level-of-detail volume rendering via 3d textures. In *Proceedings of the 2000 IEEE symposium on Volume visualization* (2000), ACM, pp. 7–13. 2

SIGRAD 2013, pp. 61-65
T. Ropinski and J. Unger (Editors)

Visual Planning and Verification of Deep Brain Stimulation Interventions

E. Abdou

Gent University - iMinds - Multimedia Lab, Ledeberg-Ghent, Belgium

Abstract

Deep Brain Stimulation is an alternative way for treating some motion disorders such as Parkinson's disease and essential tremor. In order to stimulate some brain centers during this intervention, high frequency electric fields are generated close by them. This involves permanently implanting a number of electrodes inside the brain. The final position of the electrodes is specified by the neurologist with the aid of fused data from CT and MR scans. In order to improve the therapeutic benefits of this treatment, the generated electric field must be studied. I developed a visualization and image analysis framework for visualize and insert the electrodes inside the brain. A mesh generator for the brain was added to the framework. The result model can be used by a PDE solver for interpreting the electric field distribution.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications—
I.4.6 [Image Processing and Computer Vision]: Segmentation—Edge and feature detection

1. Introduction

Deep Brain Stimulation (DBS) has resulted in a renaissance as an alternative way for treatment of Parkinson's Disease (PD) and essential tremor (ET). The ultimate goal of the Deep Stimulation research is to clearly demonstrate the stimulation process, understanding the reasons beyond several types of motion disorders and optimize the surgery. Eventually, this may lead to safer and faster surgery or even non-invasive electrical stimulation for the motion disorders. Benabid in [BA94] demonstrated the efficacy of SubThalamic Nucleus (STN) stimulation in parkinsonian patients treatment and it is safer than stimulation of Ventralis InterMedius. After one year of following up for twenty patients, Benabid showed that 60% improved in the Unified Parkinson Disease Rating Scale (UPDS) compared to stimulation of Ventralis InterMedius. In order to stimulate some regions in the brain, Deep Brain Stimulation techniques, employ the use of a high frequency electric field in the brain. The electric field is generated through chronic implanted electrodes close to the motion centers of the brain. The final position of the electrodes are depending on neurological and psychiatric disorder. Two important factors are affecting the success of the DBS treatment; the target localization and the stimulation parameters. The SubThalmic Nucleus stimulation is considered the most promising brain center treatment for most of the PD motion disorders. The STN is located in the midbrain and has an small almond shape. The position of STN is clearly identified in T2-weighted MRI and can not be easily identified in T1-weighted MRI. Nevertheless, T2-weighted MRI suffers from geometric in-homogeneity, that makes the localization of the STN in the image inaccurate. The relation between the STN position and the anterior and posterior commissure of the third ventricle in T1-weighted MRI can be found in [PG08].

Within this paper, I describe a multi-view visualization and image processing system, that can be used by the neurologist and physicist to design the leads and insert leads in the brain. The fused volumes help the physicist to visualize the final placement of the leads inside the brain and design a model to be used by a mesh generator. I describe in the following sections the image processing, the placement of electrodes, the data visualization and mesh generation for the designed case.

2. Related Work

In this section I review some related work used during designing the proposed system. A lot of work was done to plan the trajectory of the electrodes inside brain, multi-volume visualization and brain segmentation.

Gemmar et al, proposed a semi-automatic procedure to localize the STN and trajectory planning in the T1-weighted MRI [PG08]. The T1-weighted MRI was smoothed using non-linear anisotropic filtering kernel to preserve the edges. He used a region growing algorithm to segment the third ventricle. AC-PC locations are identified on midsagittal plane calculated from the segmented ventricle. A cost function evaluates the trajectory path from each entry point on the brain surface to the STN location. The best group of trajectories will be elected to be used by the surgeon. A framework introduced by Henri et al for brain segmentation using 3D mathematical morphology [HM], was used in this work.

The electrical field distribution in the region of contact with the electrodes are evaluated by the use of Finite Element Methods. In [AM08] and [FA11], the importance of the stimulation parameters and new target investigation were discussed. A mapping algorithm from the MRI to the 3D mesh were used to measure the electrical conductivity of the tissue in contact of the electrodes. A 3D modeling of the electrodes were used with the electrical conductivity properties, to visualize the electrical distribution around the targets. Delaunay tetrahedralization is a common way to produce 3D meshes from a cloud of points. A Refinement Delaunay tetrahedralization algorithms modify the meshes generated from Delaunay tetrahedralization to fit some criteria such as; mesh size and the mesh angle [Law77]. In [DBB09b] [DBB09a] labeled segmented data was used to produce 3D meshes. The mesh engine in their work is used in this paper to produce a mesh for the brain and the labeled electrodes.

3. System Overview

The system proposed here, was designed to support the neurologist and physicist to design a case to be used for further studies of the electric field distribution around the localized target. In order to use the large amount of data produced from different modalities, image processing, segmentation and volume fusion are applied to give an insight of the data with less mental effort. The MR and CT images are used before and after the DBS intervention to examine the anatomical structure of the brain. The DBS procedure starts by the planning procedure, followed by surgical procedure and ending up with postoperative follow up [SH10]. During the preoperative procedure the head is scanned by CT and MR scanners. The *Stereotactic frame* are used during MR and CT scan to fix the position of the patient's head [GP01] and to guarantee accuracy and facilitate surgical planning. The trajectory plan of the leads and target localization, is main output from this phase. It requires a very accurate computing tool. The presented system was designed to use the trajectory plan data to place the leads inside the fused volumes from the MRI and CT. The stereotactic's landmarks appears in both scans serves as basis for rigid images registration and coordinate systems [DT08].

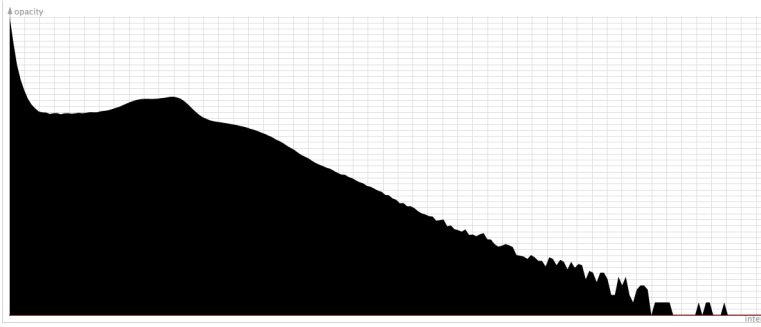


Figure 1: T1 MRI histogram

3.1. Brain Segmentation

The visualization process of the CT and the MRI includes fusing the two volumes in one model. MRI is supreme in soft tissue imaging. The aim of the analysis process in this section, is to segment out the brain from the skull and the skin in T1-weighted MRI. The resulted brain volume is visualized with the skull from the CT and is used for the mesh generation as I describe later in this paper.

I used a 3D mathematical morphology method to segment the brain from the skull in the MRI [HM]. This method uses prior anatomical structure information to be converted to morphological information. The brain can be defined as an elliptical shape that is surrounded by a dark circle (skull) and thin bright circle (skin). The developed algorithm can be divided in two stages. The first stage converts the gray valued volume to a binary volume by thresholding the images. The second stage applies, the 3D morphology operations and image filling to generate a brain mask. The distance function used for the erosion is a 3D version of chessboard.

A typical T1 weighted MRI histogram is shown in figure 1.

The first step is to threshold the image to eliminate as much as possible from the brain surroundings. The threshold values are crucial for the following steps to work perfectly. I proposed a histogram processing technique to generate the threshold values. The histogram of the MRI was first calculated and then smoothed by a Gaussian filter of standard deviation equal to 7 for removing local variations and window size of 5. The local minimums were computed. The lower minimum within every 10 pixels were picked. The previous process generates the lower and the higher threshold values. The lower value of threshold removes the CSF and the skull. The higher threshold value discards higher density materials (eyes, vessels and fat). The output from this step is a binary image representing the brain and the skin.

The resulted binary image is eroded by a cube of $3 \times 3 \times 3$ voxels, which is 2mm in every axial slice, to remove the narrow connections between the brain and the skin. A 2D fill-

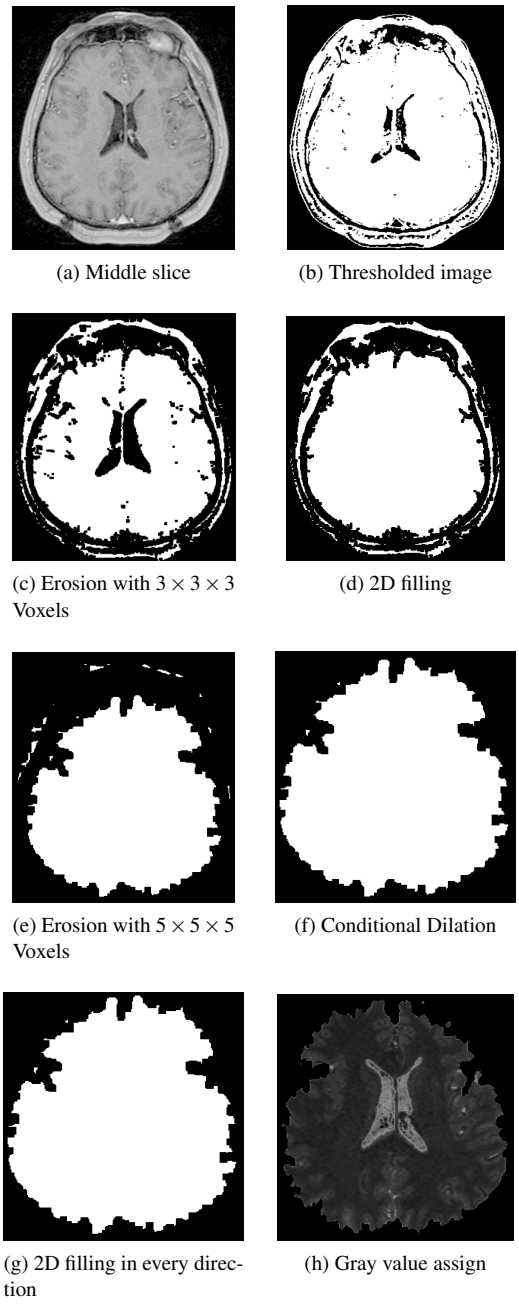
ing algorithm is applied after this to every slice in the image to fill up the holes in the connected components. In order to suppress the big connections between the brain, the eyes and the ears, the resulted volume from the previous step is eroded by a cube of (5mm) $5 \times 5 \times 5$. The biggest connected component from this step is considered to be the brain. Due to the successive erosions in the previous steps, some of the parts of the brain are eliminated. By applying conditional dilation on the result from the first step and by using the marker from the previous step, the brain is reconstructed. Finally, a 2D filling algorithm is applied to every slice. The result is a brain mask that is replaced by gray value from the original volume. The result of every step is shown in the figure 2.

The resulted volume has a 90 degrees corners and straight edges, that is resulted from the erosion and dilation of a cube shape structured element. Nevertheless, the produced edges are not affecting the accuracy of the calculation of the electric field. According to my knowledge, until now electric field calculation in the brain is using Dirichlet boundary condition and the edges of the brain are far away from the leads [ea]. The produced volume corners will be further smoothed during the mesh generation making the electric field calculation using Neumann boundary condition even possible. The insertion of the leads are not depending on the produced edges, because the insertion was planned by another sophisticated tools.

3.2. Multi-Volume Visualization and Lead Insertion

Medical image modalities blossomed during the 20th century, Computed Tomography and Magnetic Resonance Imaging marked a major steps in operation planning. The data acquired from CT and MRI are fused to reveal the interesting parts during insertion of the leads in the brain. During insertion of the leads and the electric field studying, MR data is the most interesting data. The volumes are rendered on the screen to discard the external shell of the brain from the CT volumes around the leads as shown in figure 3. The CT volume and MR volume are assumed to be registered. For volume rendering, I used the ray casting technique presented in [KW03]. I modified the previous work by applying conditional ray casting. The CT voxels are discarded when the MR voxels are not zero. I employed this to prevent the blending of gray value of the CT voxels when MR voxels are presented. The geometric information of both volumes are used by ray casting algorithm for early termination [Sch05]. I added to the previous rendered volumes, an axial and a sagittal view for the MRI and the inserted leads. The leads outside the slice position are clipped out.

The planning of the lead's and electrode positions usually is done by some sophisticated tools. The geometry of the leads are used to place the leads inside the image space. Vertex buffers are created for the surface representation of the leads. The leads surface patches are divided into three patches that represents the top of the lead (half sphere), the



(a) Middle slice (b) Thresholded image

(c) Erosion with $3 \times 3 \times 3$ Voxels (d) 2D filling

(e) Erosion with $5 \times 5 \times 5$ Voxels (f) Conditional Dilation

(g) 2D filling in every direction (h) Gray value assign

Figure 2: Brain segmentation steps

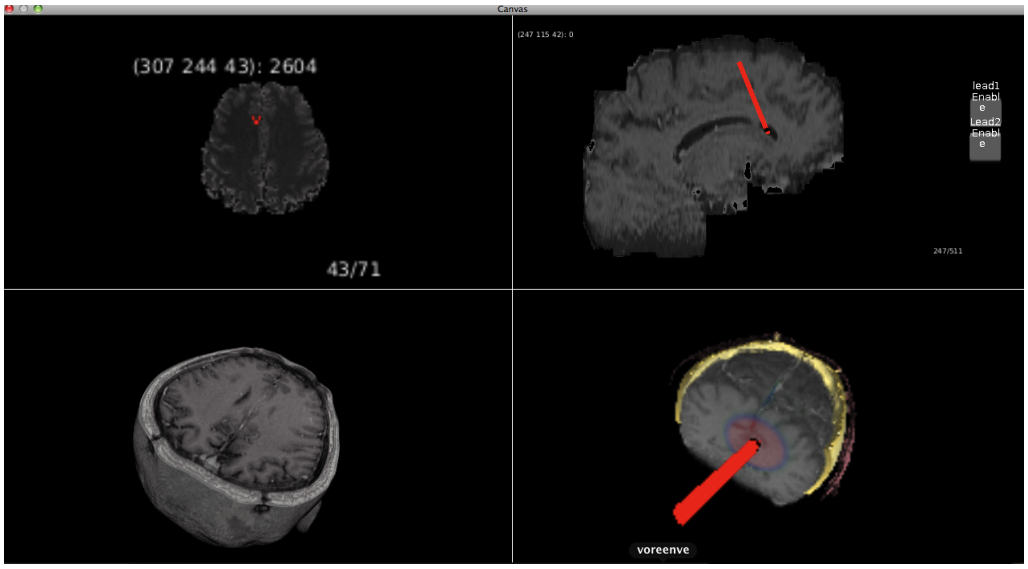


Figure 3: The top left view is axial view. The top right view is sagittal view. The bottom left view is MRI. The bottom right is reconstructed volume with a lead

electrodes (cylinders) and the body of the leads (cylinders). Color attributes are used to visually distinguish between the lead and the electrodes. The leads are rendered in a different path and the depth information is used to blend the colors in the final scene from the volume and the geometry. In this work, I used the lead specification from the Medtronic model 3387 manual [MTS].

3.3. Brain Meshing

The meshing processing is the process of converting a 3D domain of points into a set of tetrahedra. The most popular algorithms of mesh generation can be divided into three classes; Delaunay triangulation, advancing front methods and grid based methods [Owe98]. The studying of mesh generation algorithms is out of scope of this work. Nevertheless, the step of generation of the mesh is required to validate the work presented here. I used the three dimensional constrained Delaunay refinement algorithm described in [She98]. Mesh generation algorithms based on Delaunay refinement starts by Delaunay triangulation followed insertion of vertices in the mesh until certain mesh criteria specified meet. In this work I specified the facet size to be 0.1, the facet angle to be 30 and the ratio between the tetrahedrons circumradius and the length of the shortest edge to be 2 as mesh refinement criteria. Shewchuk [She98] called the input for the algorithm a *piecewise linear complex* (PLC). PLC defines the a set of vertices, segments (which are a contiguous edges presented in the final mesh) and facets which will be triangulated in the final mesh. *CGAL* [cga] provided an 3D mesh Delaunay mesh generator based on the previous described work.

In order to identify the facets for the mesh engine, the voxels of zero value are not considered for the triangulation. The voxels occupied by the leads were set to different values. The leads can be modeled by a quadric equations of cylinders and sphere to represent the lead ad lead's tip. I used these quadric equations that represents the leads, to specify the voxels that are inside the leads. I also used the binary mask representing the segmented brain, that was generated from the segmentation algorithm discussed earlier, to avoid treating the CSF inside the ventricles as holes. The facets and segments of an isosurface representing the input domain are used with the labeled voxels as inputs for the mesh generation engine. The resulting mesh of the segmented brain with one lead inserted is shown in figure 4.

4. Conclusion and Future Work

In this paper I presented a framework for multi-volume visualization and image processing, that can be used for visualizing the lead insertion in the brain. The inserted electrodes inside the brain are meshed to be used for further studies of the electric field. The segmentation process of the brain gives better interactivity on the result of the multi-volume visualization by controlling the transfer functions of every volume separately and removes the complication of ray cast algorithms to separate between bony structures and soft tissues. The main of idea of integrating the segmentation algorithm in this framework is because of its importance in the meshing step. The electric characteristic, that physicist wants to study exists in the brain soft tissues that is provided by MRI.

In the future, I would like to evaluate the presented system by a physicist or neurologist. I also hope to build and

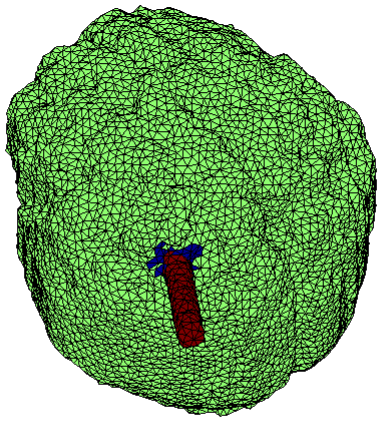


Figure 4: Mesh generation of the brain and one lead

enhance the image analysis system to fully segment out the brain organs. This will make the visualization task for the brain organs clearer and more accurate for the electric field calculation. Also I hope to use DTI for adding organ's conductivity values to generated mesh. I would like to include visualization techniques for the electric field distribution and to find an intuitive way for the adding electric field values in the MRI of the brain.

ACKNOWLEDGEMENTS

The work presented in this paper was conducted during my master thesis of computational science at Uppsala University. I would like to thank the Center of image analysis at Uppsala university, Sweden for their help and comments. The presented work is realized using Voreen framework (www.voreen.org).

References

[AM08] ASTROM M ZRINZO LU T. S. T. E. H. M. W. K.: Method for patient-specific finite element modeling and simulation of deep brain stimulation. *Med Biol Eng Comput* (2008). 2

[BA94] BENABID AL POLLAK P G. C. E. A.: Acute and long-term effects of subthalamic nucleus stimulation in parkinsonâĂŹs disease. *StereotacticFunction Neurosurgery* (1994). 1

[cga] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org. 4

[DBB09a] DOBRINA BOLTCHEVA M. Y., BOISSONNAT J.-D.: Feature preserving delaunay mesh generation from 3d multi- material images. *Computer Graphics Forum* (2009). 2

[DBB09b] DOBRINA BOLTCHEVA M. Y., BOISSONNAT J.-D.: Mesh generation from 3d multi-material images. *Medical Image Computing and Computer-Assisted Intervention* (2009), 283–290. 2

[DT08] DANIEL TARSY JERROLD L. VITEK P. A. S. M. S. O.: *Deep Brain Stimulation in Neurological and Psychiatric Disorders*. Humana Press, 2008. 2

[ea] ET AL C. C. M.: Electric field and simulating influence generated by deep brain stimulation of subthalamic nucleus. 3

[FA11] FYTAGORIDIS A ÃĚSTRÃŰM M W. K. B. P.: Stimulation induced side effects in the posterior subthalamic area: distribution, characteristics and visualization. *Clinical Neurology and Neurosurgery* (2011). 2

[GP01] GILDENBERG P L SPIEGEL W.: The early years. *Stereotact Funct Neurosurg 77* (2001), 11–16. 2

[HM] HENRI MAITRE THIERRY GERAUD I. B.: Structures from mr images using morphological approaches. *Medical Image Analysis, Oxford University Press.* 2

[KW03] KRUGER J., WESTERMANN R.: Acceleration techniques for gpu-based volume rendering. In *Visualization, 2003. VIS 2003. IEEE* (2003), pp. 287–292. 3

[Law77] LAWSON C. L.: Software for c1 surface interpolation. *Mathematical Software 3* (1977), 161–194. 2

[MTS] : *DBS electrode specification model 3387.* 4

[Owe98] OWEN S. J.: A survey of unstructured mesh generation technology. In *INTERNATIONAL MESHING ROUNDTABLE* (1998), pp. 239–267. 4

[PG08] P. GEMMAR O. GRONZ K. F. P. M. F. H. C. D.: Automated target location and trajectory selection for stereotactic planning in deep brain stimulation. *Biosignal 2008* (2008). 1, 2

[Sch05] SCHARSACH H.: Advanced gpu raycasting. In *In Proceedings of CESCG 2005* (2005), pp. 69–76. 3

[SH10] SIMONE HEMM K. W.: Stereotactic implantation of deep brain stimulation electrodes: a review of technical systems, methods and emerging tools. *Medical and Biological Engineering and computing 48* (2010), 611–624. 2

[She98] SHEWCHUK J. R.: Tetrahedral mesh generation by delaunay refinement. In *Proc. 14th Annu. ACM Sympos. Comput. Geom* (1998), pp. 86–95. 4