SIGRAD
2014

Proceedings of SIGRAD 2014

Visual Computing

June 12-13, 2014

Göteborg, Sweden

Edited by

Mohammad Obaid, Daniel Sjölie, Erik Sintorn and Morten Fjeld

Linköping Electronic Conference Proceedings No. 106
Linköping University Electronic Press
Linköping, Sweden, 2014
ISBN: 978-91-7519-212-3
ISSN: 1650-3686 (print)
ISSN: 1650-3740 (online)
http://www.ep.liu.se/ecp_home/index.en.aspx?issue=106

SIGRAD 2014

FOREWORD

SIGRAD

Welcome to SIGRAD 2014 –

the annual meeting of the Swedish Computer Graphics Association (SIGRAD) – taking place at the Chalmers University of Technology in Gothenburg.

The association's mission is to be a meeting point for researchers and industry professionals who are interested in computer graphics and adjacent areas, such as visualization and human-computer interaction (HCI). In recent years, the association's activities were focused on the organization of the annual conference, which attracts a growing number of participants. The rapid development of computer graphics technologies and the acknowledged need for visual computing solutions has led to a growing interest in graphical computing in both commercial and academic areas. Started in Sweden in 1976, SIGRAD has become an annual appointment for the Nordic community of graphics and visual computing experts with a broad range of backgrounds. Through more than three decades, SIGRAD has offered a forum to present and disseminate new technological results, new paradigms and new visions advancing the state-of-the-art of visual computing.

SIGRAD 2014 offers a strong scientific program that well represents current research and developments in the field in our part of the world. Leading researchers, from 6 different countries will present their recent work; there will be 14 papers. Peer reviewing was carried on by a highly qualified group of researchers of the Program Committee; we engaged 38 reviewers from 11 countries on 4 continents. Each paper was reviewed by at least three members of the Program Committee, one of them playing as Associate Chair. The various sessions explore research questions and solutions in several domains, and cover a wide combination of subjects.

We are especially honored to welcome our invited speakers: Albrecht Schmidt (professor of Human Computer Interaction at the University of Stuttgart, Germany) who gives an opening keynote called "The World around us is the User Interfaces - From Graphical User Interfaces to Interaction with the Ubiquitous Computing". Yossi Somer (CEO of Animaze Technology AS, Oslo) who gives an industrial keynote called "Revolutionizing a booming 3D industry: The global evolution". Niklas Elmqvist (professor of electrical and computer engineering and the director of the Pivot Laboratory at Purdue University, USA) who gives a vision keynote called "Visualization is Dead! (Long Live Visualization!)". Frédéric Vernier (professor of computer science, South-Paris University, France) who gives a closing keynote on "Information Visualization at Tabletops".

Besides the keynotes, we invited two more speakers: Ulf Assarsson (professor of Computer Graphics at Chalmers University of Technology, Sweden), who presents "Experiences from publishing research at Siggraph" and Karthik Ramani (professor of Mechanical Engineering and professor of Electrical and Computer Engineering, Purdue University, USA), who presents his work on "Using Hands: Sketching, Shaping and Building to Create".

We would also like to thank those who have contributed behind the scenes to the conference organization: the Visual Arena staff (Thomas Hanson, Mia Leterius, and Åsa Andblad), Pawel Wozniak (student volunteers organization), the economic officers of the Department of Applied IT at University of Gothenburg and Chalmers (Pia Ovik and Sanna Erling Glyssbo), our universities information officers (Hanna Nordqvist, Karin Lundberg, and Peter Larsson), and the controller of the IT Faculty (Margaretha Jansson). For their session chairing engagement during the conference, we would like to thank Alexandru Dancu, Daniel Sjölie, Erik Sintorn, Mohammad Obaid, and Pawel Wozniak. We also thank student volunteers Chen Chen, Dan Dolonius, Jesper Molin, Lars Lischke, Matz Johansson, and Petras Sukys.

Finally, we are highly thankful to our partners: Visual Arena Lindholmen offers us to use their well-equipped, staffed studio. The researchers of the t2i Interaction Laboratory at Chalmers provided significant support in realizing this event. The colleagues at the Division of Interaction Design enabled us to have institutional support along the way, which was also the case for the Department of Applied IT, both at University of Gothenburg and Chalmers. Finally, we have largely benefited from the research network within the Marie Curie Actions of FP7/2007-2013/, REA grant number 290227, DIVA.

Researchers from Chalmers, Sweden and NUS Singapore are partly supported by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT) through the Strategic Grant for Internationalization; grant number 2012-019. Finally, we have received generous support from the IT Faculty at University of Gothenburg, as well as from the Swedish Research Council (Vetenskapsrådet, VR), grant number 91294, and to some extent from EON Reality Inc.

Morten Fjeld, Chalmers University of Technology, Sweden, General Chair
Daniel Sjölie, University of Gothenburg, Sweden, Co-Chair
Mohammad Obaid, Chalmers University of Technology, Sweden, Co-Chair
Erik Sintorn, Chalmers University of Technology, Sweden, Co-Chair

PROGRAM
OVERVIEW

SIGRAD

JUNE 12TH

8:30	Registration
9:15	Welcome
9:30	**Opening keynote** Albrecht Schmidt
10:30	Announcements
10:35	Coffee break
11:00	Paper session 1
12:00	Lunch
13:20	**Industrial keynote** Joseph (Yossi) Somer
14:20	Paper session 2
15:00	Announcements
15:05	Coffee break
15:30	Paper session 3
18:30	Dinner

JUNE 13TH

9:00	**Keynote** Niklas Elmqvist
10:00	Paper session 4
10:40	Announcements
10:45	Coffee
11:00	Paper session 5
12:00	Lunch
13:20	**Invited speaker** Ulf Assarsson
13:40	**Invited speaker** Karthik Ramani
14:00	**Keynote** Frédéric Vernier
15:00	Closing remarks

ORGANIZATION

OVERVIEW

SIGRAD

Morten Fjeld, General Chair (Chalmers University of Technology, Sweden)
Daniel Sjölie, Co-Chair (University of Gothenburg, Sweden)
Mohammad Obaid, Co-Chair (Chalmers University of Technology, Sweden)
Erik Sintorn, Co-Chair (Chalmers University of Technology, Sweden)

PROGRAM COMMITTEE

Alex Olwal (Google, USA)
Alireza Entezari (University of Florida)
Andreas Kunz (ETH Zurich, Switzerland)
Andreas Kerren (Linnaeus University, Sweden)
Bernhard Preim (University of Magdeburg, Germany)
Burkhard Wuensche (University of Auckland, New Zealand)
Csébfalvi Balázs (Budapest University of Technology and Economics)
Francesco Banterle (ISTI-CNR, Italy)
Gerd Bruder (University of Würzburg, Germany)
Heidrun Schumann (University of Rostock, Germany)
Jonas Unger (Linköping University, Sweden)
Jose Diaz (CRS4, Italy)
KangKang Yin (NUS Singapore)
Kresimir Matkovic (VRVis Forschungs GmbH)
Manuela Waldner (TU Vienna, Austria)
Marco Fratarcangeli (Sapienza University of Rome, Italy)
Matt Cooper (Linköping University, Sweden)
Michael Doggett (Lund University, Sweden)
Nils Andersson (EON Reality, Sweden)
Oliver Mattausch (University of Zurich, Switzerland)
Przemyslaw Musialski (TU Vienna, Austria)
Rafal Mantiuk (Bangor University, UK)
Ruediger Westermann (TU Munich, Germany)
Stefan Bruckner (TU Vienna, Austria)
Stefan Seipel (University Gävle, Sweden)
Tania Pouli (Technicolor, France)
Thomas Ertl (University of Stuttgart, Germany)
Thomas Wischgoll (Wright State University, USA)
Timo Ropinski (Linköping University, Sweden)
Tomas Akenine Möller (Lund University, Sweden)
Tomas Larsson (Mälardalen University, Sweden)
Ulf Assarsson (Chalmers University of Technology, Sweden)
Veronika Solteszova (University of Bergen, Norway)
Veronica Sundstedt (Blekinge Institute of Technology, Sweden)

In cooperation with:

Visual Arena Lindholmen
t2i Interaction Laboratory, Chalmers
Division of Interaction Design, University of Gothenburg and Chalmers
Department of Applied IT, University of Gothenburg and Chalmers
Marie Curie Actions of FP7/2007-2013/, REA grant n°290227, DIVA

With the support of:

IT Faculty at University of Gothenburg
Swedish Research Council (Vetenskapsrådet, VR): grant number 91294
EON Reality Inc.
STINT, the Swedish Foundation for International Cooperation in Research and Higher Education: grant number 2012-019

VISUAL ARENA
LINDHOLMEN

UNIVERSITY OF
GOTHENBURG

CHALMERS

Table of Contents

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

IPFViewer: Incremental, approximate analysis of steel samples

M. Thurau[1] & C. Buck[1] & W. Luther[1]

[1]University of Duisburg-Essen, Germany

Abstract

This paper describes the design of a visual analysis tool for our Inclusion Processing Framework, called IPFViewer. The tool has been designed in cooperation with a large German steel production facility in order to acquire knowledge from data collected about nonmetallic inclusions and other defects in steel samples. We have highlighted parts of the framework in previous publications in interdisciplinary journals. Here, we describe our contribution in the area of grouped or clustered data items. These data groups are visualized by techniques known from uncertainty visualization to make visible fluctuations and corresponding variations in steel samples. However, our results are also transferable to other ensemble data. To find an optimal way to design algorithms and visualization methods to process the huge data set, we discuss the project-specific requirements regarding memory usage, execution behavior and precision. By utilizing approximate, incremental analysis techniques when needed, the responsiveness of the application is ensured while high precision is guaranteed for queries with fast response times. The design allows workers at the steel production facility to analyze correlations and trends in billions of data rows very quickly and to detect outliers in routine quality control.

Categories and Subject Descriptors (according to ACM CCS): Computing Methodologies [I.3.6]: Computer Graphics—Methodology and Techniques

1. Introduction

In [HBT*12] we presented a new approach to measuring nonmetallic inclusions and other defects in steel samples at a real-world steel production facility. These samples are continuously sliced by a milling machine to accumulate volumetric data about any defects they might contain through repeated image capturing and processing. For each defect various attributes, including volumetric statistics, such as volume and sphericity, are calculated and stored. In [BBPL14] we continued our work by classifying the defects with the help of machine learning algorithms. As a result, three types of defects can be identified: pores, such as those containing trapped gases; solid inclusions containing non-metallic elements, like aluminum oxide; or longish cracks. In [TBL14] we reported on some features and the visual capabilities of the visual analysis tool IPFViewer, which was built explicitly for visually analyzing the huge data set. We also responded to objectives and tasks defined by the steel experts in greater detail and presented applicable visualizations and interaction techniques. In this paper, we report on ongoing research in the area of data groups — in particular, groups of samples. Samples using the same production parameters are of special interest. Through natural fluctuations during the steel-making process, each sample is different. These differences lead to uncertainties in all kinds of measurements, including the amount, size and position of defects in the sample. Uncertainties are visualized with the help of uncertainty visualization techniques to allow the analysis of variances. These large data groups, which contain billions of data items, yield performance problems that we have not previously discussed in detail.

Visual analytics of the data set provides highly valuable information. Steel experts want to know what influence various process parameters have on the outcome of the steel product (sensitivity analysis). Their interest lies in a better understanding of the steel making process, which has been a significant focus of research for over 80 years. Defects have a major impact on the mechanical properties of steel and therefore on its steel. The goal is to improve the quality of steel while, at the same time, lowering the costs of producing

it. Additionally, quality control in daily usage is of interest. By comparing the measured production outcomes with reference data that show the intended outcomes, manufacturers can monitor production and detect outliers.

Our scientific contribution is a design study that solves a specific real-world problem in the domain of steel production. The targeted audience is steel experts who want to analyze their steel quality. First, we define a new task which was not discussed in our previous papers: the grouping of multiple objects, mostly the grouping of samples of similar production parameters. After specifying the design requirements from a visual analytics perspective, we justify our general design choices based on related work in the literature. The design itself is then presented along with a use case. These are analyzed in order to validate the design choices and point out possible improvements. In short, the design choices are

- uncertainty visualization to show fluctuations.
- data management based on expected query cost.
- incremental, approximate analysis for large queries.

2. Task Analysis and Design Requirements

We begin by defining the tasks necessary for an analysis of milled steel samples based on data collected by a large German steel facility. Each sample is an ensemble member in an ensemble data set [WP09]. There are multiple steel samples containing multiple defects. Therefore, the data set is a tree of ensemble members, or hierarchical ensemble data. The data is also large, multidimensional and multimodal. In [TBL14], we presented a new way of dealing with that data, which came down to three tasks:

- **Task #1: Single node analysis**
  Analyzing a single node (a subtree) in the data tree (e.g., a sample and its defects) with the help of a multiple view system to generate hypotheses and to find relationships and anomalies.
- **Task #2: Repeatability analysis**
  Knowledge from task #1 is reexamined in neighboring objects by utilizing *small multiples of multiple views*.
- **Task #3: Trend analysis**
  How does a correlation between two or more attributes vary with respect to other attributes? For example, one hypothesis is that, the larger a defect, the more spherical it will be. Such correlations may only be true for some of the smoother steels. This is shown by sorting the small multiples.

Next, we define a new task, which is the aggregation or grouping of multiple samples and defects:

- **Task #4: Aggregation of objects**
  Repeat task #1 to #3 with user-defined groups of probations or defects.

The outcomes of the steel making process have certain natural fluctuations even when the same production parameters have been used. Therefore, analysis of a single sample may lead to misconceptions if that sample is not representative or is an outlier. By grouping multiple samples, there is a high probability that the resulting statistics will yield a general truth that is less susceptible to outliers and will predict the features of future production. This prediction depends on breadth of the range of fluctuations. These variances may depend on some process parameters and are important factors. This kind of analysis is known as uncertainty or variance analysis. Steel experts want to know the boundaries of the range of possible outcomes. They want to be able to identify best- and worst-case samples as well as the range of most probable outcomes.

Aggregating samples with similar production parameters helps in the analysis of the range of possible outcomes, while aggregating samples with similar outcomes helps in the analysis of the range of process parameters suitable to achieve particular outcomes. For example:

- **To guarantee similar production conditions, the following parameters are important:** steel type, temperatures, amount of oxygen and timings.
- **Similar outcomes are grouped based on:** amount and types of defects and total cleanliness.

After identifying the tasks for the steel analysis, we imposed additional requirements on the system in order to cope with the challenges familiar in visual analytics:

Requirement #1: Visualization of uncertainty. As shown in previous publications, each sample is visualized with the help of different statistical visualizations. For instance, a trend graph showing the influence of the defect diameter on the defect eccentricity in a particular sample. Larger defects may be more spherical. When dealing with a whole group of samples, we get a family of curves (one curve for each sample) [KLM*12]. These multiple curves can be aggregated and then visualized as a single trend chart with uncertainty bars or box plots.

Requirement #2: Flexible parameter set. The concrete set of data attributes for samples and defects may change from time to time. There are thousands of process parameters and measurements stored for each sample and defect on different independent database systems. For security reasons, however, the database for the analysis will be separated and process parameters will be added and removed regularly based on current analysis goals.

Requirement #3: Fast response times. There are expected to be more than 10 billion defects over the next 10 years. Statistics over such huge amounts of data items cannot be calculated quickly enough to allow interactive analysis with continuously changing user demands. Scalability to huge datasizes is one of the current challenges in visual analytics in the research agenda [TC06].

3. Related Work

The visual analysis of ensemble data sets is relatively new [WP09]. Wilson et al. published a general overview of challenges of ensemble data sets. We compared our general approach with their paper about Ensemble-Vis [PWB*09] in a previous paper [TBL14].

To enable fast querying in large databases, a great deal of research is being conducted in the area of database optimization [KIML11]. There is a great demand on standardized implementations at the database level, because optimizations at this level are beneficial for all visual analytics tools [TC06]. However, such techniques are not yet mature or available for standardized databases.

A recent publication in this area is BlinkDB [AMP*13]. Instead of querying every single data item, it uses a small sample of the database to get an approximate answer. There are a lot of sampling based solutions available. The challenge is to find an optimal sample that best approximates the precise answer. BlinkDB manages multiple sets of uniformed and stratified samples for that purpose. The precalculated sample selection is based on predictions of the most likely future queries. In their case, they predict the columns involved in the query. Their work is very impressive, as they developed an approximate query engine with both error and response time constraints. The disadvantage of systems based on prediction and pre-calculation is a certain degree of inflexibility when compared to incremental, approximate analysis. Some answers can be given very quickly, while others cannot. This is a problem in explorative analysis, where all answers should be provided interactively. In our case, the flexible parameter set is an additional burden to pre-aggregation. In BlinkDB's taxonomy of four different workload models, the most flexible one is the online aggregation model. On the negative side, online aggregation does not offer solutions to increase efficiency. It is only supposed to return intermediate results so that users are involved in the calculation process. They can monitor interim results and intervene accordingly. An optimal way to gain efficiency and flexibility at the same time is to use hybrid systems, which may arise when mature implementations of each technique are available. Systems can evaluate query costs and choose the optimal corresponding technique.

Until then, we have chosen to implement the most flexible technique, online aggregation, which was called **incremental, approximate analysis** in a recent paper in visual analytics [FPDs12]. The same technique is also used by the CONTROL project [HHW97]. Incremental, approximate analysis or "online incremental analysis" has many advantages over other techniques. The idea is to calculate answers on a small sample of the data first. The approximate answer is visualized, while another sampling of the data is obtained that does not include the data items of the previous sample (sampling without replacement). After that, the two approximate answers are merged to obtain a better approximation. At the last data iteration, all available data has been taken into account, which leads to either precisely or closely approximated results. What was missing in previous publications was a discussion of the algorithms used. While [FPDs12] only named *average*, *sum* and *count*, more sophisticated statistical functions, like *quantile estimation*, are needed for a versatile analysis tool. Below, we will discuss and categorize some requirements of statistical algorithms.

Visualizing uncertainty when dealing with summary statistics is commonly done with box plots or similar techniques [PKRJ10]. Instead of analyzing individual values, the average value can be used. Depending on the desired precision of the information, more precise summaries for data distributions are available. One of these is the five-number summary, which consists of the min and max values, the upper and lower quartiles and the median. The family of curves provides a good example. When a trend graph visualization has multiple data points per bin (one for each curve), the bin can be summarized with a corresponding summary. The trend graph shows a trend including its possible variations just as we need it for the analysis of groups of steel samples and defects (Fig. 3).

4. The Data Model

Generating groups of objects can be understood differently. When dealing with more complex algorithms that try to group objects on nontrivial aspects, literature refers to clustering. We cluster data offline only and make results available as attributes in the database. The IPFViewer groups objects based on intervals of their attributes in real time. This means, for instance, that we group the samples based on melting temperature intervals. This is also known as data binning. On the other hand, we might group samples according to categorical data dimensions, like the steel type. We allow the definition of an unbound number of dimensions for clustering. That means that users can first create clusters based on attributes such as defect diameter, for example, small and large. The resulting clusters can be further divided by defect type, such as cracks and pores. In this way, we obtain four clusters — small cracks and small pores, large cracks and large pores — which then are visualized.

4.1. Algorithm Requirements

First of all, we require **limited memory usage**. Depending on the system's memory capacities, the amount of data items and distinct data dimensions required for one or multiple visualizations, we can run into a memory overflow quite fast. As a result, we want to keep only the current iteration in memory. Data items of older iterations have to be summarized in such a way that their meaning distributes to the overall calculation without saving each item separately. An example is the calculation of the average. We can calculate and store the average of the complete first iteration without saving each individual item. In the next iteration, we can merge

the results to get a more precise average. This requirement makes quantile calculation difficult. To calculate quantiles with a naive algorithm, we need all data items to be sorted in a single list. A lot of research regarding quantile estimation is available [GK04].

The second requirement concerns **execution behavior**. Algorithms can only work batch-wise, iteratively or completely. Batch-wise means that the data is divided into parts, or batches, which are handled one after the other. When working with batch-wise algorithms, we assume the ability to merge results (Mergeable Summaries [ACH*12]). For instance, we could calculate the quantiles of several batches. However, these batch quantiles would not reveal the quantiles of the overall data. While it is possible to use the median of medians, which fulfills the batch-wise behavior, this method is not as precise as needed. On the other hand, the count function can be executed batch-wise, and simply summarizing the results of the batches results in an outstanding ability to merge them. Other examples are *sum, count, min, max* and *avg*. In contrast, iterative algorithms handle data items one after the other. Here, we also assume the single pass criterion, which means that each data item is viewed only once. If that were not the case, we would overburden the limited memory available in order to store the data items for subsequent passes. While batch-wise and iterative algorithms are appropriate for very large data sets, algorithms that require access to the complete data set without the ability to provide interim results are completely unsuitable, as discussed above.

Data handling is divided into multiple steps. At each step, execution behavior can vary. From a database we normally get multiple data items batch-wise to reduce overhead. After that, a subsequent algorithm with an independent execution behavior may run within the application. Sometimes, batch results can be calculated directly within the database system. We assume that this was the way CONTROL [HHW97] and [FPDs12] worked. When dealing with more complex algorithms, especially quantile estimation and data mining, we receive all data items batch-wise and calculate results within the application.

Another requirement deals with the **precision of results**. A statistical calculation can be precise or it can yield an estimation with precise bounds or an estimation with probabilistic bounds. While trying to achieve as much precision as possible, we work with all these precisions and make sure to tell the user the degree of precision currently available. As stated in the introduction, a high degree of precision is not necessarily needed for analyzing general trends or creating hypotheses. Related to precision is the need for a priori knowledge. Some algorithms achieve good results only for certain data distributions. Due to the nature of explorative analysis, we cannot foresee such data distributions. Other a priori knowledge often needed is the number of data items. When dealing with many data items that are highly filtered, we would have to use a previous pass to count them. Therefore, we require algorithms without any a priori knowledge.

4.2. Algorithms based on query costs

	Itemcount: multiple billion	Itemcount: multiple thousand
Standard func.: count/ max/min/ sum/avg/ std. binning (e.g. histogram)	precise Batch-wise calculation by DB, merging by application	precise One-step by DB
Statistical func.: quantiles variance skewness kutosis mode stat. binning (e.g. box plot)	approximation in some cases Batch-wise retrieval of values from DB, iterative calculation by application	precise One-step by DB

Table 1: *The data processing steps depend on table sizes and desired statistics*

We differentiate between low and high numbers of data items in the expected results. As mentioned above, the exact number of data items is not known a priori when filters are involved. However, the typical item count for tables is familiar; thus, we know the limits for some types of queries:

Query all samples: max. 100 thousand items
Query defects from one sample: max. 100 thousand items
Query defects from multiple samples: up to 10 billion items

For a small number of items, we calculate the statistics completely in one step within the database system and achieve a precise result in an acceptable amount of time (Tab. 1). For large numbers of items, we differentiate between standard functions, which are very well suited for batch-wise merging [ACH*12], and more complex functions, which are calculated by the application iteratively. For the latter, we receive all data items in batches from the database and then iteratively calculate the desired statistics using the Boost Accumulator library †. These calculations, however, are not necessarily precise, but may be approximations in some cases. For instance, the quantile estimation uses the extended P-Square algorithm [JC85]. When binning is involved, which means a statistical calculation for each

† Boost Accumulator: www.boost.org

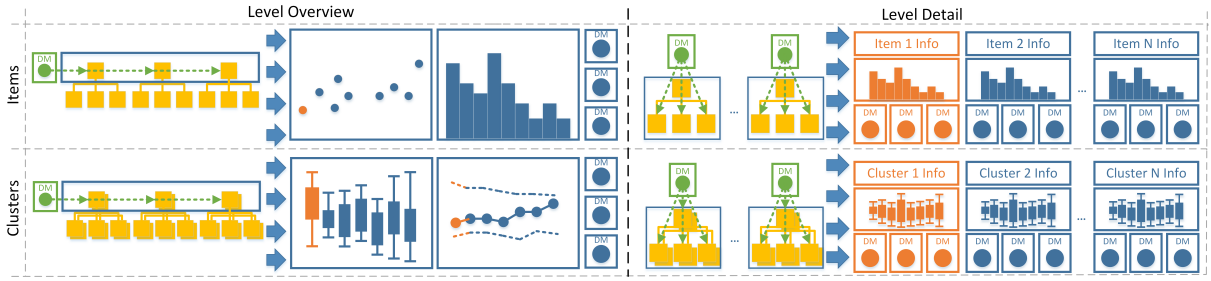


Figure 1: *The data model is divided into two parts. On the left is a level overview showing an overview of the level of interest in the data tree, which is selected by the user. Each data point in the overview corresponds to a sub-tree shown on the right. The sub-trees are visualized side by side in detail in the level detail view, using a multiple view system. The color orange indicates connections between the nodes. Grouped data is visualized with uncertainty to represent fluctuations of group members.*

bin is needed, we also differentiate between standard functions (histogram based on *count*) and statistical binning (box plots). Our implementation regarding incremental data mining is not mature yet. We mine and recommend data items based on most occurrences (*mode*) and extremes (*min* and *max* of attributes) of pre-calculated item descriptors. Once we have found suitable items, more corresponding data dimensions are received and presented as recommendations, such as the most dangerous defect.

	small table	large table
small workload per item (count, avg, etc.)	no limit	limit scanned items
large workload per item (aggregates, joins)	limit returned items	limit returned and scanned items

Table 2: *Query limit depends on table size and expected workload*

One of the problems we ran into was **paging within the database**. Paging is the operation of limiting the data to small portions, or pages, and receiving them one after the other using multiple queries utilizing LIMIT and OFFSET. As the database does not necessarily cache previous operations, a large offset means rearranging all items below the offset in the same order as before (the order must be unique), which can be a long operation. This was solved using the indexed ID as order and filter queries to the first ID that is a potential match. In this way, previous entries do not have to be scanned again. On the negative side, this prevents an optimization of the intermediate results based on sampling order, which is one of the most common optimizations. However, this problem can be solved by creating additional indices. Our solution is shown in Tab. 2. Using these limitations, the workload is distributed relatively equally among the queries, independent of the number of items that match the filters.

5. The Visualization Model

We now present the visualization model, which is shown in Fig. 1. First, a level of the main data tree is chosen as the level of interest. For instance, one might want to analyze the samples by conducting a comparison and a trend analysis. Another level of interest would be the hierarchical level containing the defects. We present all the data items on that level of the data tree in a visualization, the *level overview*, and also visualize each data item in detail side by side in the *level detail*. Each data item or cluster in the level overview is represented by a single visual item within a visualization, like a point or a bar. Multiple visualizations can be used as the level overview. In the level detail, each data item or cluster is represented by a comprehensive multiple-view system [WBWK00], showing various data dimensions, modalities and visualization forms simultaneously. Users can specify the data and visualization forms, like histograms, bar charts, graphs, scatter plots, pie charts and textual information. Since each visual item in the level overview relates to one comprehensive visualization layout in the level detail (Fig. 1, orange), sorting, filtering and clustering the data level affect both visualization methods. This is very effective for trend analysis [TBL14]. To sum up, we provide a small multiples visualization [Tuf90] (actually small multiples of multiple views) with an additional overview, as shown in Fig. 2.

The upper part of Fig. 1 deals with non-clustered data. We focus on the lower part regarding visualization of clusters. A cluster is a group of nodes on the data tree. The **precision of the summary statistic** can be selected. Instead of showing a single value to represent a large cluster, such as an average, multiple values representing boundaries have advantages when examining uncertainty [MTL78]. Here are some examples:

1 value: avg, median, count, etc.
2 values: min+max, upper/lower quartiles, etc.
3 values: min+avg+max, upper/med/lower quart., etc.
5 values: five-number summary (boxplots), etc.

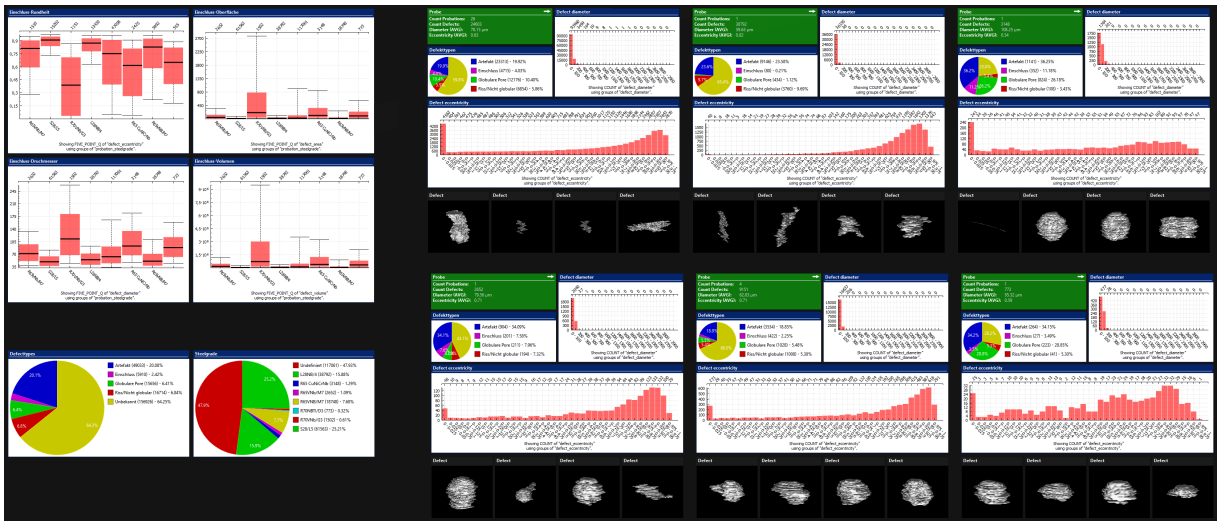


Figure 2: *A screenshot of the main window of the tool. In the overview on the left, uncertainties of four different outcome measurements are shown for different steel grades (bar charts). Two pie charts show the distribution of two categorical attributes. On the right, each steel grade is visualized in greater detail. Each steel grade uses the same layout, which enables easier comparison. Additionally, for every steel grade, four of the most significant defects are shown.*

For categorical data dimensions, like the steel type, we cannot use these numerical statistics. Instead, statements about most occurrence are given as textural information, utilizing the statistical function *mode*.

Besides pure processing of data by humans based on information visualization, we also added machine-based **Data Mining** [KMS*08] in the level overview via a tree traverse through the level of interest, thus horizontally. While data from lower and higher levels can be used during the mining process, the mining result will be an item or cluster in the level of interest (Fig. 1, green). The mining results are visualized as a recommendation to the user. This is very valuable in combination with level filters. For instance, the user may choose to analyze samples from the current week only. At first sight, exceptional samples from the current week are seen. That approach was also chosen for the level detail. Each sample or cluster has a large list of defects attached to it. The data mining process here is done vertically. Exceptional defects belonging to the current sample are found and recommended to the user, as are exceptional samples in the case of clustered data (Fig. 2, four exceptional defects recommended for each cluster).

6. Use Case

This paragraph describes the sequence of a typical analysis process:

Users choose one of three categories of objects of interest. These are *defects*, *samples* or *aggregates*. After choosing aggregates, they specify one or more data attributes that define

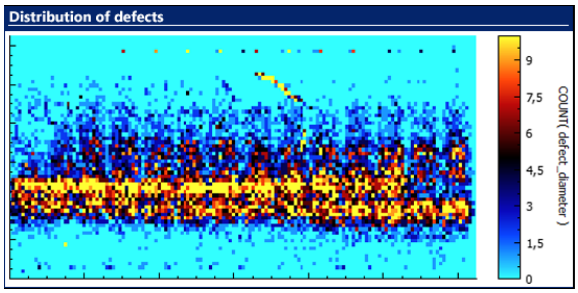


Figure 4: *Visualizing the defect bands for various steel types.*

the aggregates. They then select the categorical data attribute *steel type*, so that samples are grouped by their steel type. In this way, users are now comparing all the steel types available in the database.

Users may also choose to focus on a smaller set of steel types. To do so, they employ filtering on some of the process parameters, such as selecting steel types with a melting temperature of about 1500 °C. and additional oxygen blown into the melt. During all interaction with the GUIs the screen is constantly updating to address the user's requests. The overview and the detail view use some standard layouts that have been defined in a previous analysis session (Fig. 2).

In the detail view, users scroll through the different steel types and see how greatly they differ in the defect sizes they contain (histogram of diameter) and defect types that occur

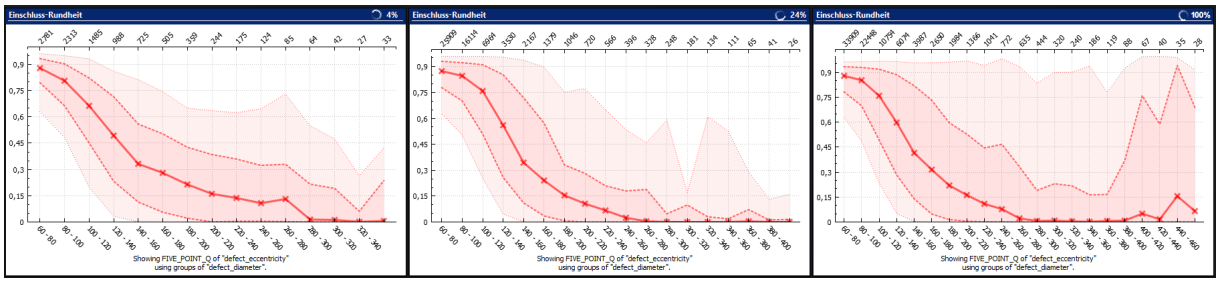


Figure 3: *Visualizing the iterative loading process of a statistical plot, showing the correlation of defect diameter and defect eccentricity: 4%, 28% and 100% of the data used.*

in them (pie chart of defect type). To determine whether there are also differences in the locational distribution of the defects within the sample, the user changes the layout to *distribution of defects*, as created in a previous session. A binned scatter plot, representing the rectangular sample format, displays a band where most of the defects are located (Fig. 4). The defect bands differ from steel type to steel type. Sometimes it is broader or wider; it may contain holes or be located differently within the sample. The user sorts the steel types according to different process parameters to see if there is a correlation between the defect band and certain process parameters.

In another analysis, the hypothesis was examined that the largest defects are more spherical and thus have a high degree of eccentricity. To do so, in the overview area, the user creates a graph showing the defect eccentricity against the defect diameters. The graph shows the average defect eccentricity for defects diameters in steps of 10μm. It seems that the hypothesis was correct: On average, the largest defects are more spherical.

Wondering why the variance is very high for large defects, the user then decides to undertake a variance analysis and edits the graph to show not only the average, but also a five-point summary. Even in between the upper and lower quartiles, where 50% of the defects having an eccentricity value around the middle are located, there are defects with no eccentricity and full eccentricity (Fig. 3). The user decides to create groups based on the defect type. This visualization strengthens and refines the hypothesis. The large defects with low eccentricity are the longish cracks, while the non-metallic inclusions are, in fact, spherical with a low variance (Fig. 5).

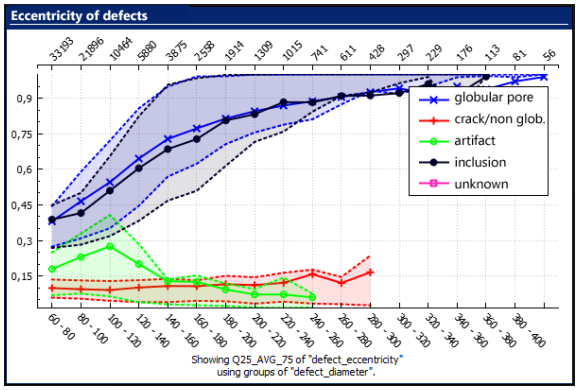


Figure 5: *Showing a correlation for different defect types.*

7. Discussion

The speed of the analysis tool is much better when compared to older versions of the software, which calculated everything in one step by database. Performance problems occurred when users changed input very quickly and ongoing calculations could not be completed. This was noticeable especially when users scrolled through a large number of clusters in the side-by-side views. By calculating the statistics incrementally, it is possible to cancel further unneeded calculation steps, greatly improving the degree of interactivity. By implementing the seek operation as described, there is only negligible overhead for paging. Calculating the quantiles with the naive algorithm in the database was significantly slower than the new method of retrieving the data batch-wise and approximating the answer. In general, by examining the data size and the statistical function of interest, we can adapt the way we process the data to meet the optimal requirements for execution behavior, speed and precision. There are algorithms which guarantee ε-approximate answers and fulfill our requirements. These algorithms guarantee an error in the approximation with an upper limit of ε. They will be implemented in future work. With these kinds of algorithms, as described, for instance, in [ZW07], we can more finely adjust the system with regard to the tradeoff between speed and precision in each iteration.

We collected **user feedback** from the steel experts who will use the tool and researchers involved in the project. Most wanted statistical function to be the quantile estimation. Compared to variance, user can see the distribution of

the values above and below the average independently. Variance only shows a general uncertainty in both directions. Feedback about performance was ambiguous. The steel facility employees work with systems that have very slow response times. Accidental user inputs are penalized by long waiting times before these inputs can be corrected. On the one hand, our new way of dealing with data was accepted. On the other, there was concern because its degree of reliability during the loading process was unknown. However, this was counterbalanced by the system's greater responsiveness. In many cases the approximations of visualizations hit the final visuals very fast due to high probability of similar data to come. We are exploring new input methods, so that all the available data is used only when explicitly wanted. This innovation would safeguard the infrastructure. Showing some data in comparison with reference data was adopted and desired for all kind of visualizations that show a single sample. For instance, when visualizing a sample, experts do not just see the facts about the collected data, but can evaluate and rate the sample compared to a reference. By defining the reference data dynamically — including rules to define similarity — great opportunities arise. Reference data is not as much of interest when clusters are visualized. Comparing multiple clusters side by side was preferred.

8. Acknowledgments

This work was funded by the European Regional Development Fund within the *Ziel 2 programme 2007-2013*.

References

[ACH*12] AGARWAL P. K., CORMODE G., HUANG Z., PHILLIPS J., WEI Z., YI K.: Mergeable summaries. In *Proceedings of the 31st Symposium on Principles of Database Systems* (New York, NY, USA, 2012), PODS '12, ACM, pp. 23–34. 4

[AMP*13] AGARWAL S., MOZAFARI B., PANDA A., MILNER H., MADDEN S., STOICA I.: Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems* (2013), ACM, pp. 29–42. 3

[BBPL14] BÜRGER F., BUCK C., PAULI J., LUTHER W.: Image-based object classification of defects in steel using data-driven machine learning optimization. *Proceedings of VISAPP 2014 - International Conference on Computer Vision Theory and Applications* (1014). 1

[FPDs12] FISHER D., POPOV I., DRUCKER S., SCHRAEFEL M.: Trust me, i'm partially right: Incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2012), CHI '12, ACM, pp. 1673–1682. 3, 4

[GK04] GREENWALD M. B., KHANNA S.: Power-conserving computation of order-statistics over sensor networks. In *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (New York, NY, USA, 2004), PODS '04, ACM, pp. 275–285. 4

[HBT*12] HERWIG J., BUCK C., THURAU M., PAULI J., LUTHER W.: Real-time characterization of non-metallic inclusions by optical scanning and milling of steel samples. *Proceedings SPIE* (2012). 1

[HHW97] HELLERSTEIN J. M., HAAS P. J., WANG H. J.: Online aggregation. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1997), SIGMOD '97, ACM, pp. 171–182. 3, 4

[JC85] JAIN R., CHLAMTAC I.: The p2 algorithm for dynamic calculation of quantiiles and histograms without storing observations. *Communications of the ACM (CACM) 28*, 10 (1985), 1076–1085. 4

[KIML11] KERSTEN M. L., IDREOS S., MANEGOLD S., LIAROU E.: The researcher's guide to the data deluge: Querying a scientific database in just a few seconds. *PVLDB Challenges and Visions* (2011). 3

[KLM*12] KONYHA Z., LEŽ A., MATKOVIĆ K., JELOVIĆ M., HAUSER H.: Interactive visual analysis of families of curves using data aggregation and derivation. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies* (New York, NY, USA, 2012), i-KNOW '12, ACM, pp. 24:1–24:8. 2

[KMS*08] KEIM D. A., MANSMANN F., SCHNEIDEWIND J., ZIEGLER H., THOMAS J.: Visual analytics: Scope and challenges. *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics, Springer, Lecture Notes In Computer Science (lncs)* (2008). 6

[MTL78] MCGILL R., TUKEY J. W., LARSEN W. A.: Variations of box plots. *The American Statistician 32*, 1 (1978), 12–16. 5

[PKRJ10] POTTER K., KNISS J., RIESENFELD R., JOHNSON C. R.: Visualizing summary statistics and uncertainty. *Computer Graphics Forum (Proceedings of Eurovis 2010) 29*, 3 (2010), 823–831. 3

[PWB*09] POTTER K., WILSON A., BREMER P.-T., WILLIAMS D., DOUTRIAUX C., PASCUCCI V., JOHNSON C.: Ensemble-vis: A framework for the statistical visualization of ensemble data. In *Data Mining Workshops, 2009. ICDMW '09. IEEE International Conference on* (2009), pp. 233–240. 3

[TBL14] THURAU M., BUCK C., LUTHER W.: IPFViewer - A Visual Analysis System for Hierarchical Ensemble Data. *International Conference on Information Visualization Theory and Applications, IVAPP 2014* (1014). 1, 2, 3, 5

[TC06] THOMAS J. J., COOK K. A.: A visual analytics agenda. *Computer Graphics and Applications, IEEE 26*, 1 (2006), 10–13. 2, 3

[Tuf90] TUFTE E.: *Envisioning information*. Graphics Press, Cheshire, CT, USA, 1990. 5

[WBWK00] WANG BALDONADO M. Q., WOODRUFF A., KUCHINSKY A.: Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2000), AVI '00, ACM, pp. 110–119. 5

[WP09] WILSON A. T., POTTER K. C.: Toward visual analysis of ensemble data sets. In *Proceedings of the 2009 Workshop on Ultrascale Visualization* (New York, NY, USA, 2009), UltraVis '09, ACM, pp. 48–53. 2, 3

[ZW07] ZHANG Q., WANG W.: A fast algorithm for approximate quantiles in high speed data streams. In *Scientific and Statistical Database Management, 2007. SSBDM '07. 19th International Conference on* (July 2007), pp. 29–29. 7

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

Using Curvilinear Grids to Redistribute Cluster Cells for Large Point Clouds

D. Schiffner[1], M. Ritter [2,3], D. Steinhauser [3] and W. Benger [2]

[1]Professur für Graphische Datenverarbeitung, Goethe Universität Frankfurt, Germany
[2]AirborneHydroMapping GmbH, Technikerstr 21a, Innsbruck, Austria
[3]Universität Innsbruck, Technikerstr 13/25, Innsbruck, Austria

Abstract

Clustering data is a standard tool to reduce large data sets enabling real-time rendering. When applying a grid based clustering, one cell of a chosen grid becomes the representative for a cluster cell. Starting from a uniform grid in a projective coordinate system, we investigate a redistribution of points from and to neighboring cells. By utilizing this redistribution, the grid becomes implicitly curvilinear, adapting to the point cloud's inhomogeneous geometry. Additionally to pure point locations, we enabled data fields to influence the clustering behaviour. The algorithm was implemented as a CPU and a GPU code. The GPU implementation uses GLSL compute shaders for fast evaluation and directly manipulates the data on the graphics hardware, which reduces memory transfers. Data sets stemming from engineering and astrophysical applications were used for benchmarking. Different parameters dependent on the geometric properties were investigated and performance was measured. The method turned out to reach interactivity for medium sized point clouds and still good performance for large point clouds. The grid based approach is fast, while being able to adapt to the point cloud geometry.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Hierarchy and geometry transformations

1. Introduction

Large point cloud data sets are produced by observations and simulations. Today, laser light detection and ranging (LiDAR) applications easily generate several billions of points measurements [PMOK14, OGW*13], similar amounts of particle based data are generated by state-of-the-art astrophysical simulations, i.e. by smooth particle hydrodynamic codes [SWJ*05, Spr05]. Interactive rendering of data becomes important, i.e., when semi-automated algorithms are applied. The classification of LiDAR data, e.g., requires a quality check and 'hand-made' corrections done by users. Here, interactive response and rendering is important for an efficient work-flow. Such large amounts of geometry data do not fit into the graphic hardware's (GPU's) memory as they easily reach hundreds of giga-bytes. Thus, data has to be prepared to support out-of-core rendering, for example in spatially sorted data fragments. But, more geometry data can be loaded onto the GPU's memory than the GPU can display at interactive frame rates.

Here, our approach aims at geometry reduction on the GPU to still achieve interactive frame rates per out-of-core data fragment. We want to avoid any additional data preprocessing, but can enhance the reduction when using pregenerated information. We cluster the incoming vertices to reduce the amount of data being displayed by creating an implicit curvilinear grid originating from an affine transformed uniform grid. The cluster process consists of two steps: a grid cluster operation and a move operation. The cluster operation is simple as it operates on an initial uniform grid. The move operation uses accumulated information from the first step and processes indices only. This allows a fine grained control over individual cells and their number of contained vertices enabling to manipulate the details of the rendered data while not needing to preserve it for further processing. In the context of out-of-core rendering and big data sets, this becomes ever increasing in importance.

Our method aims to be:

fast. Utilization of the GPU and preparing the clusters via GLSL compute shaders to reduce memory transfers and unleash parallelism available in standard workstations.

simple. Data organization is linear and no hierarchical data structure is required. It is directly used for rendering.

flexible. Besides the vertices, additional data fields such as scalar, vector, or tensor fields, may be taken into account for clustering. The method can be combined with a coarse Level of Detail technique on out-of-core data fragments.

versatile. Since the grid is being adjusted to the point clouds's geometry, the method is applicable to different kinds of data. The approach makes no assumption on how the point cloud originated or if it represents a specific kind of geometry. The points may or may not describe lines, surfaces, volumes, or other geometrical distributions, available at many time steps or just at one.

We chose application motivated data sets to do benchmarks. We study different parameters of the approach and the influence of data set properties on the performance and the applicability.

After having provided some background information, we gather related and previous work in section 2. The approach is described in-depth in section 3. Data sets stemming from LiDAR and from astrophysical particle simulations, see section 4, are the basis for benchmarks in section 5. Here, the main results regarding to timing and visualization are presented and discussed. The article is concluded in section 6, and closes with thoughts on future work in section 7.

2. Related and Previous Work

The application of vertex clustering recently has grown in interest due to its fast processing capabilities. Linear methods, such as grid based clustering methods, are especially well suited for large data sets that may contain several million or even billion data points. By reducing the input set, such as presented by DeCoro [DT07] or Willmot [Wil11], the rendering of large data is possible again with a little overhead at the initial clustering phase. In the latter case, individual attributes of an input data set are stored separately to increase detail after reduction.

Promising results have also been achieved by Peng and Cao [PC12], as they are able to provide frame-to-frame coherence when applying their reduction method. Their method is based on an edge collapse algorithm, which was presented by Garland and Heckbert [GH98]. They apply the computation of the quadric error metric in parallel and then decide where to reduce and restructure the output triangles.

The selection of individual level of details is also a crucial part and often includes offline processing methods. In [SK12] we used a parallel approach to dynamically update the current representation while retaining interactivity. This could be done by computing a raw estimate of the object that is being iteratively refined.

An comparison of two clustering algorithms has been presented by [PGK02]. In this case, a hierarchical and an incremental clustering method are applied to reduce point-set-surfaces [ABCO*01], where cells were iteratively refined. Both approaches showed good results regarding reduction quality and run-time performance. Clustering especially in the context of SPH data sets has been utilized by [FAW10] with a perspective grid. They include a hierarchy (octree) in the data organization and apply texture based volume rendering in front to back order of the perspective grid for drawing.

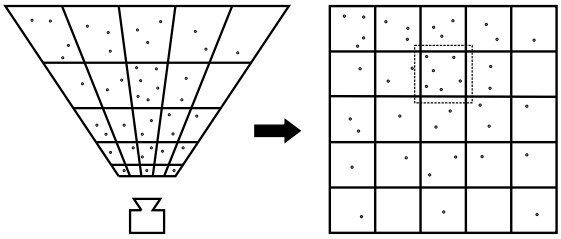


Figure 1: Points are transformed into a local coordinate system of the camera view frustum. Initial cells are defined by a uniform grid. The clustering algorithm operates in this coordinate system. The grid's geometry preserves more detail close to the camera and reduces detail far way.

[PGK02] use a covariance technique in the point neighborhood to compute a reconstructed 'surface normal' and to measure a distance from a cluster point to the original surface. A similar method based on the same dyadic product, called the point distribution tensor, was introduced in our previous work [RB12]. However, the product also includes distance weighting functions and the analysis based on the tensor's Eigenvalues is different. Three scalar fields are derived from the second order tensor called linearity, planarity, and sphericity. These describe the geometric point neighborhood and are normalized between 0 and 1. If points are distributed on a straight line, linearity is high, and if points are distributed on a plane planarity is high, respectively. We precomputed the planarity for some of the data sets used in the benchmarks and include it in the clustering process, such that variations in planarity are preserved and homogeneous planar regions are clustered.

3. Our Approach

The main idea behind our approach is to re-size individual cells based on their internal data. The less points contribute to an individual cell, the better the quality once a reduction is applied. This applies, as long as the representative is being computed using the values taken from a single cell.

The most basic scenario for shrinking a cell is that it contains more points than their neighbors. This can be achieved by reducing the cell extents. Note, that this reshaping does not alter the actual data but is only used internally to derive a new cell. More elaborate results can be achieved, by using

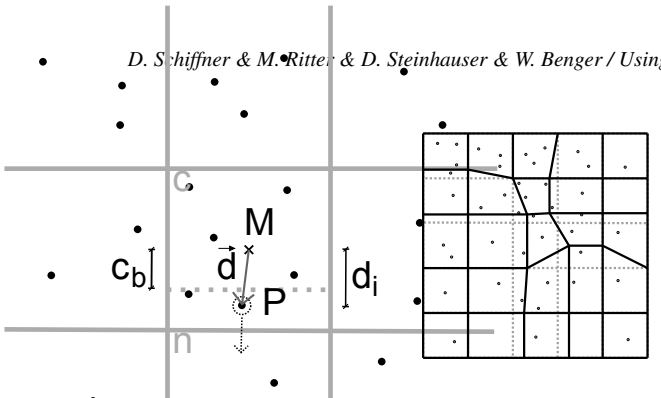


Figure 2: **Left:** Detail of the marked cell in Figure 1 illustrates how a point is "moved" from one cell to its neighbor below. A point P of the cell c is assigned to a different cell if the largest component d_i of the direction vector $\vec{d}$ from the cell center M to P is larger than a certain cell bound c_b which depends on parameters of cell c and the neighboring cell n. **Right:** Curvilinear grid after moving the points. Note that the curvilinear grid is not computed explicitly, but indirectly defined by the points assigned to each cell.

geometric properties, such as curvature or tensor data, that may have been computed in advance.

Also, this approach can be combined with classical warping techniques. In these cases, a non-uniform transformation is applied prior to the clustering, see Figure 1.

3.1. Overview

We apply a 3 step method to create the reduced input set: *cluster*, *move*, and *reduce*. The first step applies a classical clustering, but we also accumulate information needed for the second step. The incoming vertices are mapped to a grid that may can be warped via an affine transformation. The resulting position is converted to an index that is used for further computations.

The second, i.e. the *move*, identifies, whether a data point needs to be placed in a neighboring cell. It uses the accumulated information from step 1 and local information of the current data point to compute new, temporary, cell bounds. If the point is located outside the temporary cell bound it is *moved* to its neighbor. This renders the cell bounds curvilinear, as the actual shape is being altered. Figure 2 illustrates the involved geometrical objects of the method, which formally is described by:

$$\vec{d} = P - M \tag{1}$$

$$\Delta_i = \max_{j=1..3} \{|d_j|\} \tag{2}$$

$$\mathrm{w}(c,n) = \min(lb, \left(\frac{\mathrm{density}(c)}{\mathrm{density}(c) + \mathrm{density}(n)} \right)^{\gamma}) \tag{3}$$

$$c_b = w(c,n) \tag{4}$$

$$\Delta_i > c_b \begin{cases} true, & move\ P \text{ to } n \\ false, & \text{skip} \end{cases}, \tag{5}$$

with M the center point of the current cell c, P a point in c, i the index of the maximal component of vector $\vec{d}$, n the neighbor cell, lb a lower bound of the cell size of c assuring its minimal size, c_b the cell boundary in direction of the component i, and γ a non linear scaling factor.

For a point P its direction vector from the cell's center is computed. Then, the maximal absolute component of this vector is chosen and compared to the according component of

Both, step 1 and step 2 scale with the size of the input data $\mathcal{O}(N)$. Each cell, identified by the index, is processed and the according data is accumulated to compute a representative data point.

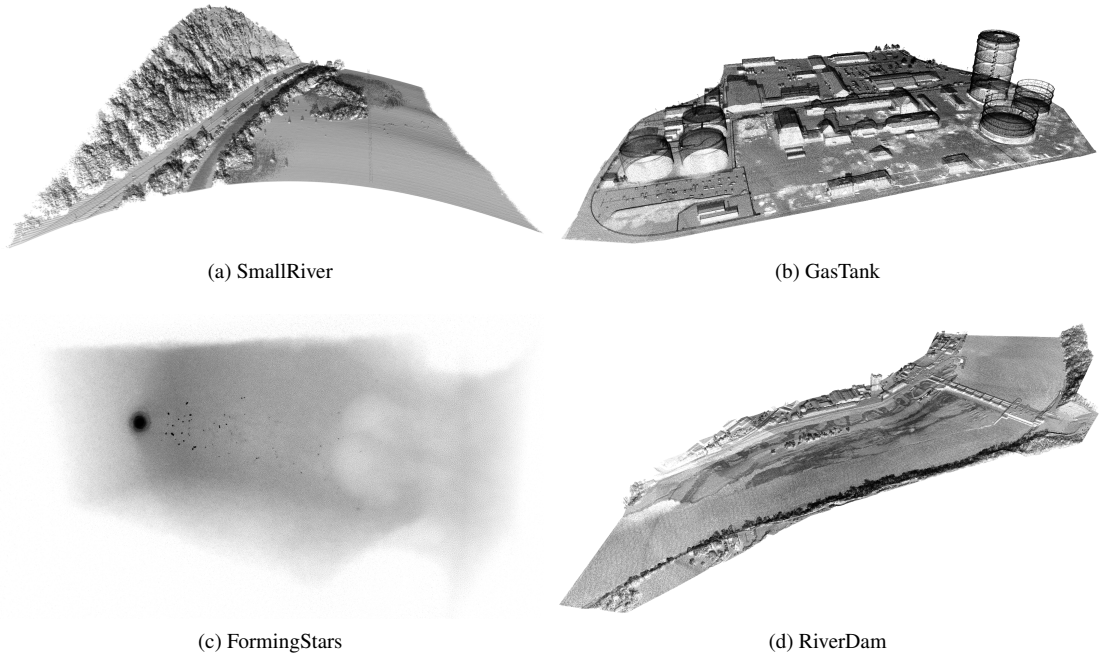
The last step simply reduces the input point set by emitting the previously averaged cell position. More sophisticated methods such as median or a quadratic error minimization could be utilized to derive the representative. As the single cells are iterated in this case, the time complexity is bound linearly with the number of cells $\mathcal{O}(C)$. The final output is a reduced point set, that can be visualized. To allow further displaying of additional data, the accumulated data of the *cluster* or *move* steps can be emitted as well.

3.2. Computation Details

The processing flow of our method can be described as follows. On each call, the input position from the raw data set is being warped by a given projection matrix. This may be identity, if no warping should be applied. The resulting position is in normalized device coordinates and is matched to the underlying grid by multiplying it with the grid size. Finally, a grid index is derived by performing a 3D to 1D mapping. From this point, the individual shaders diverge and different operations are performed.

In the *cluster* operation, a scalar value is read from an additional buffer that is aligned with the input positions. This value represents the individual weight of an input point and is atomically added to an internal counter. We also store a maximum value to allow visualizations regarding the overall weight.

In case of the *move* operation, each cell is compared to their immediate neighbors. The previously summarized counter is used to extract the total weight of a cell. Weights that are less than the derived neighbor's weight are not taken into account and the processing is aborted. Otherwise, the new cell bounds for the current active cell are computed by applying formulas (4) and (3). We use the internal counter from the first step as the density(c) and set used a default parameter set for all tests ($lb = 0.1, \gamma = 1.0$). If the current position exceeds the new cell bound, the current point is emitted to that neighboring cell used in the computations.



(a) SmallRiver (b) GasTank

(c) FormingStars (d) RiverDam

Figure 3: Four point cloud data sets were used for testing. Different sizes and different geometrical distributions are benchmarked. The points in the LiDAR data sets are mainly distributed on surfaces with small volumetric regions in vegetation and water. The point density varies relatively little over the whole data set. The SPH simulation of forming stars is fully volumetric and has small regions of much higher point densities.

Name	Nr. of Points	Scalar Field
SmallRiver	2.075.993	Planarity
GasTanks	11.133.482	Planarity
FormingStars	16.250.000	Type / Density
RiverDam	26.212.555	Planarity

Table 1: Data set sizes used for the benchmarks.

4. Data Sets

We use data sets stemming from LiDAR measurements and an astrophysical particle simulation to test our algorithm, see Figure 3. Table 1 lists the data sets, its sizes, and an available scalar field on the points.

LiDAR: For the LiDAR data three airborne scans with increasing complexity were chosen. The data was captured with a green laser system by Riegl, the VQ820g, specialized for bathymetric scanning. The laser system has an especially high pulse rate of up to 520 kHz and a wide footprint optimized for capturing shallow water regions. The *RiverDam* data set was enriched by additional sonar measurements and, thus, includes ground echos of the deeper (>3m) river sections, besides the shallow water regions of the fish ladder (<3m) [DBS*13]. Such high density bathymetric laser scans are used for hydraulic engineering, planing water related building structures, and environmental engineering. Grids for numerical hydraulic computations can be generated, e.g., for flooding simulations or morphological studies. To generate such grids from a point cloud several processing steps are required. Points are filtered and geo-referenced. Then, they are classified into, at least, the two classes: water and non-water points. Next, the water surface is extracted and non-water points are corrected to eliminate the effect of the water's refraction. Especially, the step of classification needs control and corrections by human users to support automatic algorithms. For all the LiDAR data sets the planarity was pre-computed, an attribute given per point, describing a geometrical property of the surrounding neighborhood [RB12]. It was computed via a the point distribution tensor and describes how closely points are distributed towards a fitting plane in the neighborhood. The radius of the neighborhood was set to 2 meters.

Astrophysics: The *FormingStars* data set represents one time step of a combined N-Body/Hydrodynamic simulation of a galaxy undergoing ram-pressure stripping [SHKS12]. Such simulations are performed in order to understand the evolution of galaxies in dense environments in the universe. In galaxy clusters, the largest gravitationally bound structures in the universe, galaxies move in their mutual gravitational field. Besides the galaxies and dark-matter, such clusters consist of a very hot and thin gas, the intra-cluster

medium (ICM). The galaxies are encountering this gas and feel its ram pressure, nonetheless it is very thin. This induces enhanced star formation within the galaxy at first, and leads to the stripping of the inter-stellar medium (ISM), the gas within a galaxy, reservoir for forming new stars. As a consequence, star formation in the galaxy ceases, but stars can be formed from stripped gas in the wake of the galaxy. The mass distribution of different components in GADGET-2 [Spr05] (gas(type 0), dark-matter(1), old stars(2), bulge stars(3), newly formed stars(4)) is discretized and sampled using a Monte Carlo method. Except gas, all other types of matter are then modeled as a collision-less fluid, interacting only via gravity. To solve the resulting N-Body problem, a tree code is used (e.g. [BH86]). The hydrodynamic equations for the gaseous component are solved via SPH (smoothed particle hydrodynamics [Mon92]). Initially, the density estimate of each particle is calculated using a kernel interpolation technique. Consequently, the momentum and thermal energy equation can be integrated in time, the continuity equation is implicitly fulfilled.

The points of the LiDAR data sets reside mostly on surfaces, such as measured ground or building structures. Only a few points captured in vegetation and water regions represent volumes. However, in the star forming simulation the points describe a volume. We want our algorithm to perform well in all cases and want to investigate its behaviour. All data sets still fit into 1GB of GPU memory, but only the smaller ones can be displayed at interactive frame rates.

5. Results

To create test results, we have implemented our approach with OpenGL using compute shader capabilities that are available since version 4.3. We did not use an OpenCL approach, as the data is going to be rendered directly after the processing. This way, we can directly control the outcome of the cluster algorithm when altering the individual parameters.

In the core specification, no floating point atomic operations are specified but can be added by using an extension from nVidia. When using other vendors, one could emulate this feature, by converting the float value to an integer. For further details, the reader may be referred to [CCSG12].

As our approach consists of two steps, we can simply omit the second one (and the additional computations) to allow an evaluation of the overhead generated by our additional *move* operation. Thus, this algorithm applies a basic clustering to the input data set.

A CPU implementation has been realized for sake of completeness. Obviously, the CPU variant will not be able to compete with the GPU implementation.

As stated before, we want to avoid any pre-computations, e.g. computation of tensors or connectivity, on the available data sets. The algorithm is able to perform a reduction without planarity information, but can produce better results with them.

5.1. Timing

Based on our applications, several benchmarks have been conducted. They vary in terms of input size, grid size and used graphics card. In general, a test has been repeated 10 times and the mean time values are given. Timings are reported in milliseconds. Each test was run with varying input parameters, i.e. the object and the grid size. These benchmarks were executed on 3 different PC's, running on Windows 7 and Linux. The results are listed in table 2. The first machine (1) consists of an i5-3450 and a nVidia GeForce GTX 460 with 1GB RAM. The second system (2) uses an i5-670 and a nVidia GeForce 680 GTX. The last configuration (3) contains an Intel Xeon-X5650 and a nVidia Quadro 5000. (1) and (2) operate on a MS-Windows platform while (3) runs a Linux system.

Model	Sys	Our[ms]	Cluster[ms]	CPU[ms]
SmallRiver	1	68.9	49.7	700.0
	2	14.6	10.2	831.0
	3	93.9	52.0	879.0
GasTanks	1	298.1	239.5	3780.0
	2	65.4	34.8	4445.4
	3	480.0	256.5	4758.5
FormingStars	1	648.8	479.9	5751.0
	2	129.8	88.6	6858.3
	3	749.0	434.4	7146.2
RiverDam	1	950.3	671.5	8670.0
	2	206.7	146.7	10292.0
	3	1228.6	719.9	11062.7

Table 2: Benchmark results of our GPU algorithm, a basic cluster approach and a CPU implementation. All shown tests have been performed with a grid size of 75x75x15. This grid was chosen due to the planar point distribution.

The individual timings indicate an overhead due to the additional processing step of our approach. Yet, we only have an increase of roughly 50% despite the additional computations performed in the *move* operation. Note that our compute shader has not been optimized and leaves room for further improvements. A visualization of the presented timings using a different grid size can be seen in Figure 4.

The influence of the grid size is in all computation steps very small. This is due to the fact that the individual steps mostly depend on the data input size, while only the last step scales with the size of the grid. As one can see in Figure 5, the GeForce 680 outperforms the older graphics cards.

5.2. Visual Results

The visualization technique in the OpenGL demo simply draws equally sized non-transparent splats. Color is con-

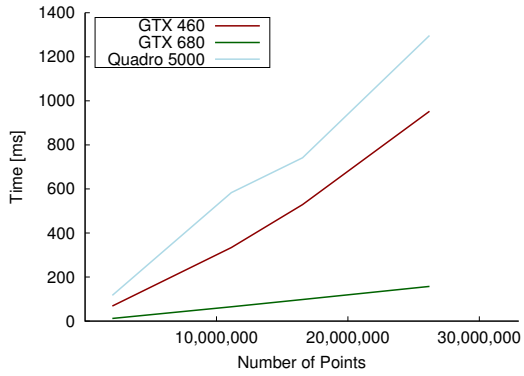


Figure 4: Timing values generated by processing each object repeatedly. The reported values are the mean of all runs. For all objects, a grid size of 200x200x100 has been used.

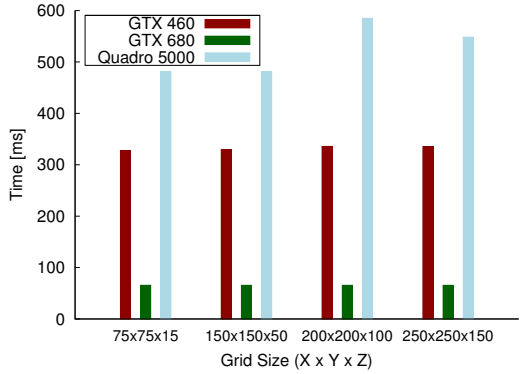


Figure 5: The influence of the grid size on the overall performance of our algorithm. The GeForce 680 GTX outperforms the other graphics cards. The Quadro, despite its larger memory, is not able to compete with the GeForce 460 GTX. We used the GasTank data set for computation.

trolled by a scalar value via a red to green color map. As presented in section 5.1, the impact of the additional *move*-step is acceptable, as the computation times are within interactive response times. The following Figures show several images that were created with both the curvilinear and a classic cluster algorithm with different grid sizes. The color map either illustrates changes based on the relative movement from the cells or the cluster cell density.

In Figure 6 some results generated with our method are shown. We used the prior mentioned data sets to apply a clustering. The colors indicate the density of the represented cell. The more red-ish the color, the more data points have been collected in this cell.

Especially with larger grid sizes, the reduction quality is increasing. In Figure 7, the cell density of each step is used for the color mapping. After application of the *move* operation, the global average is reduced, which results in the red color, as the same maximum is used for the mapping. The lower image visualizes the differences regarding the additional *move* operation. The curvilinear grid matches the underlying source more closely, as can be seen via the cluttered splats at the top right of the image.

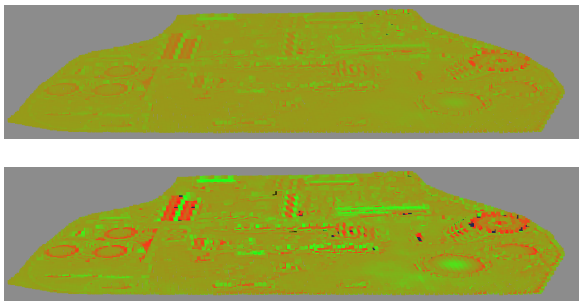


Figure 8: GasTank data set visualized clustered on a perspective grid. **Top:** *move* operation based on cell densities only. **Bottom:** *move* operation including the scalar field planarity which was computed in a pre-processing step. Smaller cells are created in regions of low planarity (e.g. edges) and, thus, preserving more detail. Dense cells are created in regions of homogeneous planar regions, were less detailed information is necessary for a good visual representation. Geometric features of the point cloud are enhanced, when taking the planarity into account.

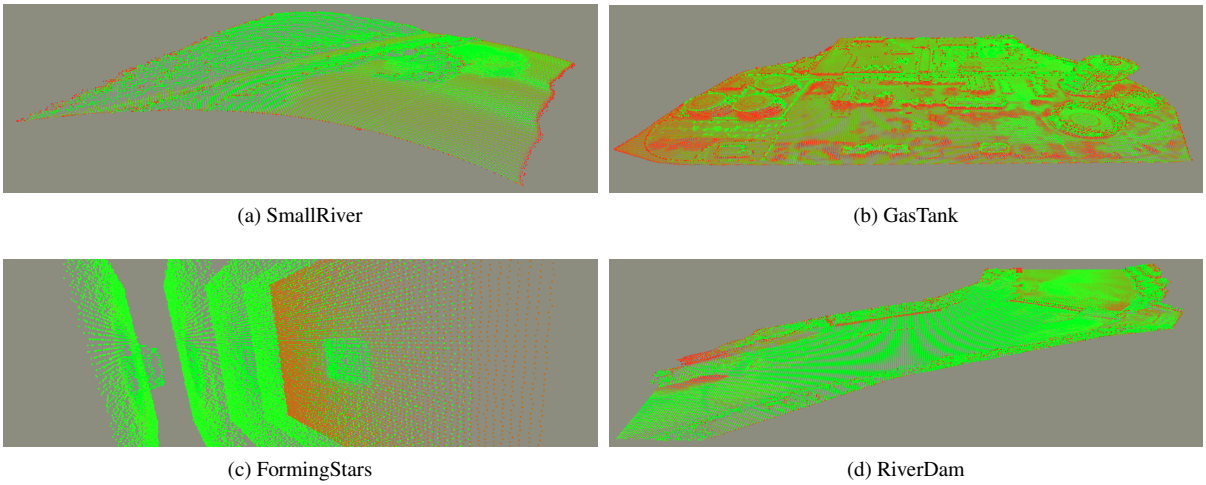
By introducing precomputed information, our algorithm can perform even better. As one can see in Figure 8, regions where edges are present are better fitted as smaller cells are used. This is indicated by the more distinct color values present in the individual cells, e.g. it the lower right of the image.

6. Conclusion

We have presented a new approach to apply a non-linear clustering to arbitrary objects. We are able to use multiple information from the current geometry and are not limited to scalar field properties. The applied reduction is made selectively, due to a restructuring of individual cells. Currently, our data sets are point based and do not incorporate connectivity information. However, an extension to triangles or polygons can easily be achieved, as shown by other researchers ([PC12, Wil11]).

The computation times of the *move* operation has been shown to be interactive for medium sized point clouds and has a good performance with large data sets. Our implementation has not been optimized and leaves room for further enhancements. For example, the calculation of cluster indices is performed in both the *cluster* and the *move* operation, which is not necessary.

We have shown the differences between classical cluster-



(a) SmallRiver (b) GasTank

(c) FormingStars (d) RiverDam

Figure 6: Visual results of the clustering for the different data sets. Color represents the cell density. The number of points per cell is illustrated by a green to red color map going from many (red) to one (green) point. Grid size varies from $150 \times 150 \times 25$ to $300 \times 300 \times 100$ in (a), (b), and (d), which yield good results for reduced overview visualizations. In (c) the grid resolution in z-direction was reduced to 5 slices allowing to see inside the volume. When inspecting the leftmost slice one can see how the representing points are pulled toward the high point density region of the galaxy, thus emphasizing a region of interest. The simple non-transparent splat rendering prevents better insights into the volume.

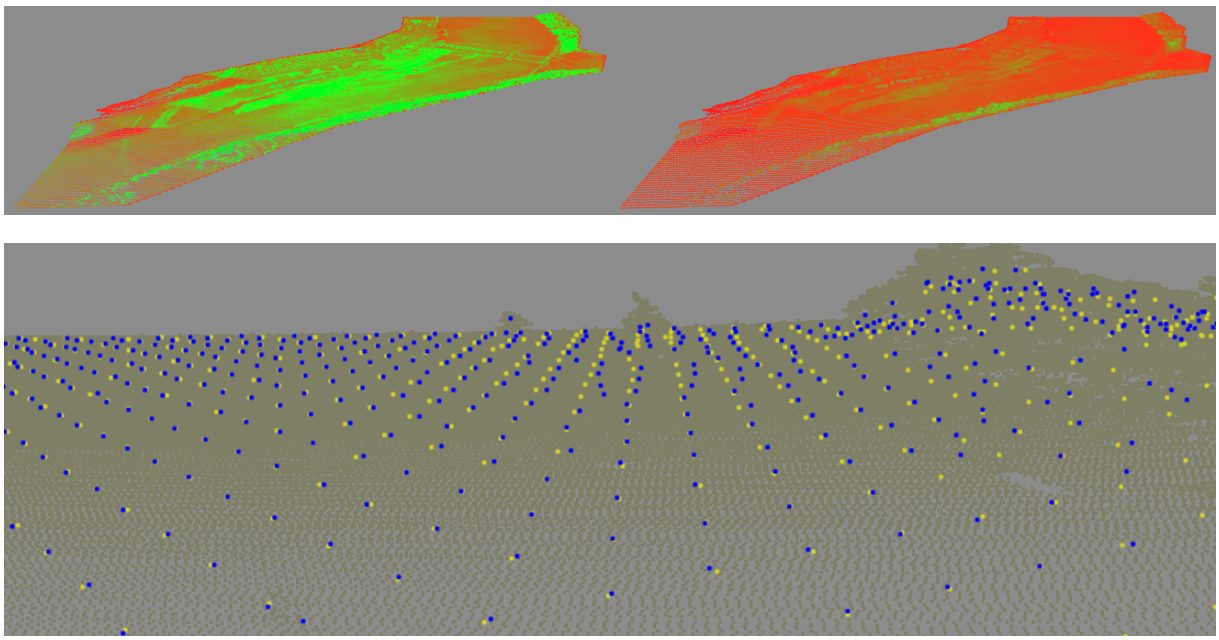


Figure 7: The differences due to the application of the proposed method. In the first picture of the top row, the green-ish regions indicate cells with high density. These are reduced by rescaling the cell sizes, which results in a more even distribution, as seen on the right top. The image below shows a detailed view, where and how the *move* operation modifies the positions of the resulting cells. The yellow cells are created by the clustering while blue ones are the result with the additional *move* operation. Note that the latter produces a splat at the tree in the top of the image.

ing and our curvilinear implementation. Due to the dynamic cells, details in an object are more likely to be preserved. This preservation of features during a rendering increases the quality and topology of the basic object, while still reducing the input data set. Thus, we have made another step towards interactive rendering of large, unprocessed data sets.

7. Future Work

The high performance of the compute shader drives us to further investigate streaming of big data. This includes a fast discard of unnecessary data, as well as selective reloading of individual fragments of a rendered object. Especially, the efficiency of the *move* allows repetitive execution (more iterations) or more complex grid modifications. We intent to use several reconstruction methods to enable the visualization of closed surfaces as well as available geometric properties, such as the point distribution tensor or the planarity. This will allow an identification of interesting regions within the large scale object. Tensor analysis may also be computed on the fly on the GPU.

The visualization can be enhanced by displaying the individual cell sizes. This way, a user could visually control, whether the implicitly generated curvilinear grid matches the expectations. Also, the information within a cluster cell could be visualized showing the influence of the available parameters to the effectively computed grid.

We also want to investigate, whether we could use the fast approximation to create a fingerprinting of these large data sets. To compare large data sets for equality, the accumulated information could be used instead of the raw data. However, it remains to be shown, if the generated data is unique enough for a clear identification.

8. Acknowledgments

This work was supported by the Austrian Ministry of Science BMWF as part of the Konjunkturpaket II of the Focal Point Scientific Computing at the University of Innsbruck and as part of the UniInfrastrukturprogramm of the Research Platform Scientific Computing at the University of Innsbruck and funded by the Austrian Science Fund (FWF) DK+ project Computational Interdisciplinary Modeling, W1227-N16. We like to thank Frank Steinbacher [ahm] to provide the LiDAR data sets.

References

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point Set Surfaces. In *IEEE Visualization* (2001), Ertl T., Joy K. I., Varshney A., (Eds.), IEEE Computer Society.

[ahm] http://ahm.co.at.

[BH86] BARNES J., HUT P.: A hierarchical 0 (N log iV) force-calculation algorithm. *Nature* (1986).

[CCSG12] CYRIL CRASSIN, SIMON GREEN: Octree-Based Sparse Voxelization Using the GPU Hardware Rasterizer. In *OpenGL Insights*, Cozzi P., Riccio C., (Eds.). CRC Press, July 2012, pp. 303–319. http://www.openglinsights.com/.

[DBS*13] DOBLER W., BARAN R., STEINBACHER F., RITTER M., NIEDERWIESER M., BENGER W., AUFLEGER M.: Die Zukunft der Gewässervermessung: Die Verknüpfung moderner und klassischer Ansätze: Airborne Hydromapping und Fächerecholotvermessung entlang der Rheins bei Rheinfelden. *Wasser-Wirtschaft 9* (2013), 18–25.

[DT07] DECORO C., TATARCHUK N.: Real-time Mesh Simplification Using the GPU. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2007), I3D '07, ACM, pp. 161–166.

[FAW10] FRAEDRICH R., AUER S., WESTERMANN R.: Efficient High-Quality Volume Rendering of SPH Data. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2010) 16*, 6 (November-December 2010), to appear.

[GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization* (1998), pp. 263–269.

[Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual review of astronomy and astrophys. 30* (1992), 543–574.

[OGW*13] OTEPKA J., GHUFFAR S., WALDHAUSER C., HOCHREITER R., PFEIFER N.: Georeferenced Point Clouds: A Survey of Features and Point Cloud Management. *ISPRS International Journal of Geo-Information 2*, 4 (2013), 1038–1065.

[PC12] PENG C., CAO Y.: A GPU-based Approach for Massive Model Rendering with Frame-to-Frame Coherence. *Comp. Graph. Forum 31*, 2pt2 (May 2012), 393–402.

[PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient Simplification of Point-sampled Surfaces. In *Proceedings of the Conference on Visualization '02* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 163–170.

[PMOK14] PFEIFER N., MANDLBURGER G., OTEPKA J., KAREL W.: OPALS - A framework for Airborne Laser Scanning data analysis. *Computers, Environment and Urban Systems 45*, 0 (2014), 125 – 136.

[RB12] RITTER M., BENGER W.: Reconstructing Power Cables From LIDAR Data Using Eigenvector Streamlines of the Point Distribution Tensor Field. *Journal of WSCG 20*, 3 (2012), 223–230.

[SHKS12] STEINHAUSER D., HAIDER M., KAPFERER W., SCHINDLER S.: Galaxies undergoing ram-pressure stripping: the influence of the bulge on morphology and star formation rate. *Astronomy & Astrophysics 544* (July 2012), A54.

[SK12] SCHIFFNER D., KRÖMKER D.: Parallel treecut-manipulation for interactive level of detail selection. In *20th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2012), vol. 20.

[Spr05] SPRINGEL V.: The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society 364*, 4 (Dec. 2005), 1105–1134.

[SWJ*05] SPRINGEL V., WHITE S. D. M., JENKINS A., FRENK C. S., YOSHIDA N., GAO L., NAVARRO J., THACKER R., CROTON D., HELLY J., PEACOCK J. A., COLE S., THOMAS P., COUCHMAN H., EVRARD A., COLBERG J., PEARCE F.: Simulating the Joint Evolution of Quasars, Galaxies and their Large-scale Distribution. *Nature* (2005).

[Wil11] WILLMOTT A.: Rapid Simplification of Multi-Attribute Meshes. In *High-Performance Graphics 2011* (August 2011).

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

Towards a Model for the Integration of Time into a Graph-based Key Performance Indicator Analysis

Stefan Hesse[1] and Rainer Groh[2]

[1]SAP AG, Walldorf, Germany
[2]Professur für Mediengestaltung, TU Dresden, Germany

Abstract

The analysis of relationships between key performance indicators is one of the challenging tasks in modern business applications. On the one hand, a complex network of key performance indicators, based on sensor data and calculations, is obviously available in technical systems, but on the other hand, the final human decision is based on the information provided by visualization types like dashboards. But in most cases dashboards only cover static information and neglects temporal dependencies. In this paper, we present an approach for the integration of a temporal perspective into a graph-based visualization for the analysis of key performance indicators using multi-level graphs.

Categories and Subject Descriptors (according to ACM CCS): H.1.1 [Systems and Information Theory]: Information theory—H.1.2 [User/Machine Systems]: Human factors—

1. Introduction

Analysts and decision makers are facing continuous changing dashboards displaying information about multiple individual key performance indicators (KPIs) in each level of the enterprise hierarchy in a global distributed enterprise. KPIs are information sources, which provide a quick overview about status and changes of e.g. business or production processes and their results [FG90]. They are significant predefined measures that provide businesses with the information they need to assess previous performance and to create profound decisions. KPIs define targets and provide individuals with the ability to assess past information [Wu02]. KPIs can be used to control systems, to observe the quality of services, notice issues and start properly actions. For example, KPIs in a production site might help to support the goal of the improvement of the product quality. With the definition based in business analysis and controlling, business KPIs can be standardized, e.g. in [ISO14] or proposed by organizations like VDMA, while others are created for special adapted scenarios. The dependencies and influences between them may be derived from a proposed key performance indicator system, such as the Balanced Scorecard [KN92] for business related KPIs. Other approaches try to derivative relationships for entire key performance indicator systems by statistical analysis [RSB09].

The interconnected and integrated source systems, e.g. sensor-actuator-systems, production lines or logistic areas, are often connected to few common dashboards for the final analysis . In addition to that, daily or weekly KPI reports are generated for the users. Using this setting, the user normally accesses the dashboard for his analysis and view tables, pie charts, line charts or bar charts of a set of individual KPIs. For example, the user might see KPIs such as 'Overall Equipment Effectiveness (OEE)', 'Throughput time', 'Re-Work time count' or 'Direct Run Ratio'. While the calculation of such KPIs become increasingly complex and interconnected, the current way of presenting the information as single KPIs on a dashboard is not sufficient for the analysis of inter-dependencies. Furthermore dashboards might not satisfy the emerging need for the visualization to display temporal inter-dependencies between KPI data objects.

The mere display of individual, unconnected KPIs in dashboards does not reflect entirely the dependent and temporal nature of data during the analysis. We observed that the missing visibility of relationships between KPIs in the dashboard leads to personal interpretation of the relationship by the user based on individual knowledge and observation. For example, events (and therefore KPI results) from an earlier date in time may influence the KPI results with mathemati-

cal and logical relationships to each other, but this influence is not visible in current production dashboards. This means, if some issue is visible in a KPI result from the assembly line of an automotive production site, the root cause for this might be caused in an issue some days before in the body shop.

While decision makers must explore causality, the implementation shows that an analysis task for KPIs might be more focused with a graph-based visualization. A graph can provide a clear hierarchy and direction information as a common ground for the interpretation. Thus, Keim states that a new kind of visualization is required to face the challenges on complexity and interaction [KKS*09]. Typical graph-based visualizations are able to present information objects of key performance indicators, such as names, values, units like dashboards before and additionally their relationship to other KPIs using the edges. But while the data source, like a business data warehouse or an in-memory database system, can provide multidimensional data sets for every time period required, only few information items out of these sets can be finally visualized without overloading the user. The contributions of this paper are twofold. Building on the ideas of [HVRH13] and [HVNK13] for static graph-based visualizations, we propose a model for integrating temporal information on top of a KPI dependency graph. Whereas in our previous work we did not include temporal information between KPIs, this work enriches the graph-based approach by defining a model how to provide a variability of temporal information.

The following section 2 will provide a sketch view into the most important related work before presenting our idea of multi-level graphs in section 3 including some detailed description of the informational and temporal characteristic of the concept. We conclude the paper in section 4 with a brief outlook to our future work.

2. Related work

The section of related work is split into two areas. First, we will take a look into the current approaches for visualization of KPIs, second, we will summarize the work in regard of integrating temporal information into graphs.

The analysis concentrated visualization of KPIs is one noticeable topic in terms of business intelligence. The main research focus is based on the improvement of dashboards and the optimization in terms of usability or style of common data visualization elements, such as tables, pie charts, line charts or bar charts [EB12, GRC04, Hil12] and [RC13] whereas the request for a more coherent visualization was been postulated long before [DeS84]. Only few authors have investigated the content that is relevant for business analysis. Most recommendations are geared towards typical information items of individual KPIs without any consideration about dependencies, which are regarded as obsolete in this way by [Kei02]. In the same context, some researchers seem to shift to a plain graph-based visualization for dependencies between business information objects [BHPS12, DT11] and key performance indicators [ALA07, WLR*11, HTB*11].

For the integration of temporal information into general graph-based visualization we have identified several approaches, which can be mapped partly to four categories of dynamic graphs. These four categories, highlighted by [BdMM08] are: 1) all nodes and edges remain fixed, but the values of attached attributes vary, 2) the sole visualization of the additions or deletions of edges over time, 3) the variation of the edge and their visualization attributes over time and 4) the visualization of additions or deletions of nodes over time. The first set of approaches proposes the generation of one unique graph for the entire time period and the visualization of the entire temporal data (e.g. using a time series visualization) within each node entirely [SKM06, BBD08, SLN05]. A quite similar approach can be seen in the works of [APP11, PS06] where a unique graph contains all temporal information for all time steps but nodes or edges are visualized in a different way for the single time step. The second area of related work covers the integration of temporal information by visualizing an individual graph for each step in time and the combination of these single graphs into one unique display whereas each graph is displayed simultaneously and applies visual differences [KNC*11, The06]. The third set of research activities focuses on the generation of dedicated graphs for each step in time [SDM12, LBD07] and the visualization of changes of values and attributes for the nodes and edges between each step in time [DGK01, FSC99, EHK*04, GBD09, FWSL12]. Following the description in the previous section none of these approaches is fully applicable for the analysis of key performance indicators, where we face the observation of the separation between large set of theoretic but enterprise wide valid KPIs as well as the application and specification of few KPIs out of this set for a dedicated use case and the analysis within a dedicated time span. We will specify this observation in the following chapter.

The proposal presented in this paper solves this challenge by formulating a general model for the graph-based visualization of time-dependent key performance indicators.

3. The concept of the multi-level graphs

3.1. Model-graph, view-graphs and nodes

While we researched the temporal setup of the different KPI dashboards used by our industrial partners, we gained insight into the composition and usage of these dashboards. First we noticed that every KPI dashboard displayed an inherited and modified subset of KPIs from a larger, but unique set of KPIs. The larger set of KPIs provided specifications, such as calculation formulas, units of measures, descriptions or identifier. While an applied subset of the KPIs was only

valid for a dedicated time span, the underlying KPI specification didn't specify any temporal parameters or maturity. Furthermore, we saw that every inherited KPI value on the dashboard represented information for a dedicated step in time. This observation of the structure, composition and temporal nature of KPI dashboards for the transformation into a graph-based visualization of KPIs leads to the necessity of a concept as described as follows.

First we assume one or more graphs representing KPI networks as a 'model-graph', where all necessary specification of KPIs and their dependencies are stored. This 'model-graph' can include multiple graphs or just individual nodes, if dependencies between some KPIs are not described or not included. Using this 'model-graph' one or multiple sub-graphs can be inherited for a dedicated analysis, monitoring or measuring purpose. Each sub-graph is called 'view-graph'. Within every 'view-graph', single 'nodes' represent the information, such as value, identifier or semantic context for a single KPI. While the analysis, monitoring or measuring task of the responsible user might overlap, multiple 'view-graphs' may contain identical 'nodes'. The following figure 1 illustrates this approach as multi-level graphs. In figure 3 an example can be seen. This figure contains one 'model-graph' with 22 nodes, where the selected nodes for the 'view-graph' are selected for illustrative purpose. The figure also contains a 'view-graph' view A with six nodes, where value of the nodes change from time step t to time step t+1.

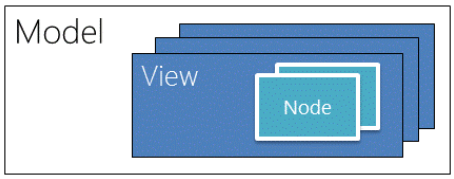


Figure 1: *The concept of multi-level graphs describes one single 'model-graph' for the generation of multiple 'view-graphs'.*

3.2. Information perspective within the multi-level graphs

The separation between 'model-graph', 'view-graph' and 'nodes' facilitates the partitioning of graph related information, graph drawing behavior and interaction. The 'model-graph', in regards to the data structure, might contain three types of information: a generic KPI specification such as formulas, a description, related business areas, all applicable units of measurement (e.g. kilowatt hour, megawatt hour, kWh/yr), the dependencies between the KPIs and information about default graph drawing algorithm or default graph aesthetic rules. In addition to that general specification from the 'model-graph', the 'view-graph' might include information about the subset of the KPIs, the maturity and the granularity of the 'view-graph' and interaction information, which is applied to the entire 'view-graph' and for the time span. This may include filters, search parameters or sub-graph folding information. Each 'node' within the 'view-graph' contains one dedicated information item, which is inherited and refined from the information provided in the 'model-graph'. The refinement can include the specific unit of measurement (kilowatt hour) or some intra-production site responsible person. The data for each time step, allocated to each node includes the value of the KPI for this time step, the identifier, the thresholds and pre-calculated state information.

3.3. Temporal characteristic of the multi-level graphs

Beside the information perspective and the separation (and inheritance) of information within the multi-level graphs, the approach permits the integration of time into each level. In our approach, the 'model-graph' is time-independent. The KPI nodes and the description of dependencies between the KPIs within the 'model-graph' are valid until modifications to the 'model-graph' occur, such as addition or deletion of nodes or edges. The 'view-graph' indicates a defined time span and the granularity of the time steps for the containing nodes. This means multiple 'view-graphs' can exist in parallel describing different time spans and time steps. The time granularity and period is selected up on creation of the 'view-graph', but might be also bound to the time steps of the underlying data source. Inside each 'view-graph', a set of information per 'node' is valid for one dedicated step in time. Hence, while the graph of the 'view-graph' (in regards to the position of nodes and composition of edges) is stable during the entire time span, the only values and states of each individual node per time step change from time step to time step. In our approach, the dependencies between the KPIs are not time dependent to avoid a destruction of the users mental map from the relationships between KPIs as described in [ELS91, MELS95]. The position of nodes and the composition of edges will not change due the life span of the 'view-graph'. This temporal characteristic is illustrated in the following figure 2. In this figure the 'model-graph' is valid between time t and t+1. The view A defines a different time span and time steps for the nodes as view B with some overlap in the time span. The cyan boxes between view A and nodes A (as well as view B and nodes B) are introduced for an illustrative purpose to link the presentation of the nodes clearly to the view.

4. Application

The approach of multi-level graphs has been implemented successful in a .NET/WPF based prototype. The prototype is connected to an enterprise service bus (ESB) for retrieving and transforming data. We refer for more information

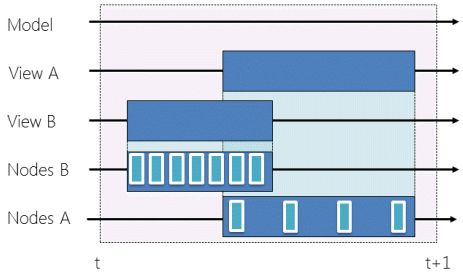


Figure 2: *The temporal characteristics of the multi-level graphs.*

about the distributed architecture to the work proposed in [VHM*12] and extended in [DKM*13]. Both provide detail to the underlying architecture in regards to the back-end implementation. The ESB extends the framework of Apache ServiceMix and integrates heterogeneous components as adapters e.g. for the access machine sensors, to web services, business data warehouses and in-memory database systems. The ESB gathers the external data using these adapters, processes this data and transforms it into a unified JMS based message exchange format for consumption by our prototype. For this paper we generated a 'model-graph' with 150 nodes. The amount of nodes represents the number of supervised KPIs within a medium scale enterprise in the manufacturing area. We know that the creation of such KPI network represents the most intellectual effort since the knowledge of dependencies between KPIs is highly distributed and available in most cases only on individual level by experts. A common expert might be the responsible users for the analysis at the production site. This means, the responsible user knows from experience about three or four dependencies between KPIs in his local work area. But often there is no enterprise wide KPI network defined. We see the non-formalized knowledge of dependencies by KPIs as one major outcome of the usage of KPI dashboards. The figure 3 illustrates this 'model-graph' with 150 nodes. The nodes in the 'model-graph' provide information about the related business-id, business-units, the formalized description of the KPI, an identifier, a data source template, the list of responsible persons for the technical implementation and possible units of measurement. Furthermore the node includes information about the relationship to other KPIs. These information can be seen as basic information per KPI for inheritance to the 'view-graphs'. Beside the KPI relevant information, the 'model-graph' provides default settings for the graph-drawing algorithm and a pre-defined set of graph-aesthetics rules per graph-drawing algorithm.

Each 'view-graph' inherits and specifies this information. For example the data source template is replaced by a data source query entry for accessing the values and the list of technical responsible persons is extended to responsible per-

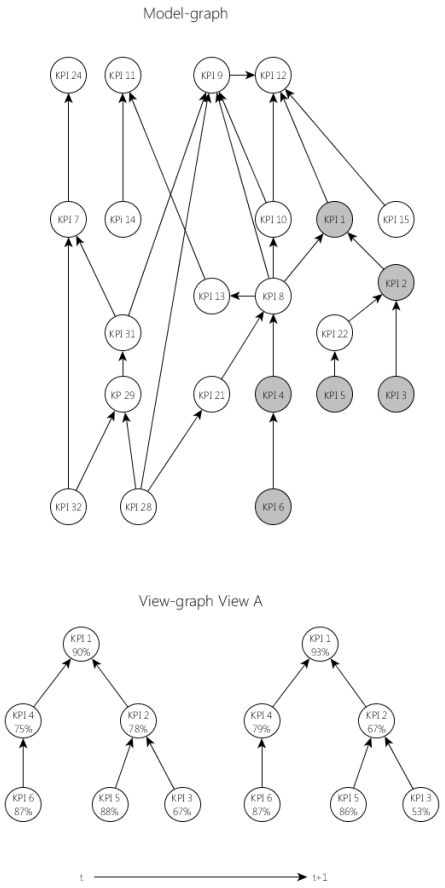


Figure 3: *Example illustration: a) on the left side a 'model-graph' with 22 nodes. nodes which have been selected for the 'view-graph' are highlighted. b) a 'view-graph' view A six nodes. The values of the nodes change between time step t and t+1.*

sons for execution of the associated business process. Typical tasks for the user using the 'view-graph' might be:

- the analysis of previous states and values of the KPIs,
- a drill down into related KPIs to detect root causes,
- an early detection of present issues which might affect other departments in the coming time period,
- and training on relationships and influences between business processes.

The selection of the KPI nodes per 'view-graph' is bound to the tasks of the user. The user can compose each view by himself using drag and drop functionality. The creation of the 'view-graphs' starts with an empty 'view-graph', where the user defines time span and period, and proceeds with drag and drop of the necessary KPIs from a list of KPI models. This means, a specialist for product quality analysis will include all daily KPIs in regard to the quality audit to his

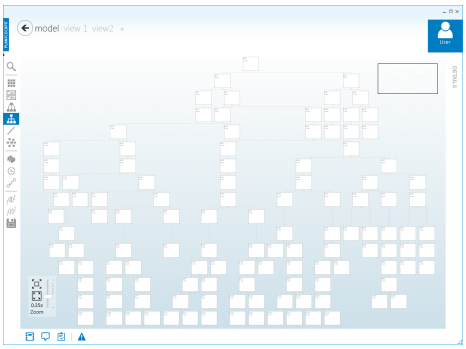


Figure 4: *The implementation of the 'model-graph'.*

'view-graphs', while an executive from the senior management may want to visualize quarterly KPIs related to budget, performance or effectiveness from distributed sources (e.g. all production sites of the enterprise) . We propose an average amount of nodes per 'view-graph' of 15 to 20 nodes, according to cite{Brown1997.

While figure 4 presents a 'view-graph' containing 16 nodes in the time span 01. July 2013 to 31. August 2013 (as view 1), the figure 5 illustrates a second 'view-graph' with four nodes for the time span between 22. October 2013 and 30. October 2013. Both 'view-graphs' inherits the dependency information from the same 'model-graph' as presented in figure 3. The user can access the single time steps for each 'view-graph' individually by using a slider control. On the left side of the screen the start of the time span is controlled, on the right side of the screen the user can control the end of the time span. We extracted one 'node' and two time

Figure 5: *A 'view-graph' named as "view 1", containing 16 nodes from the 'model-graph'; Time span: 01. July 2013 to 31. August 2013.*

steps as figure 6 for illustration. This 'node' is included into 'view-graph' named 'view 1'. For part a of the figure 6 the

Figure 6: *A 'view-graph' named as "view 2", containing 4 nodes from the 'model-graph'; Time span: 22. October 2013 to 30. October 2013.*

value is 15.17 for the 06. July 2013 and for part b the value is 23.80 for the 31. July 2013.

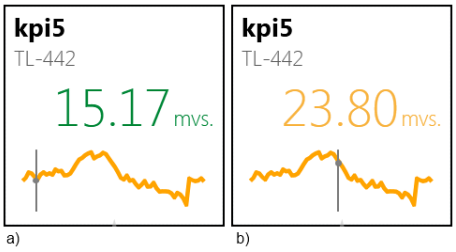


Figure 7: *Comparison of two time steps from one 'node' out of "view 1". time step for part a) 06. July 2013, time step for part b) 31. July 2013*

5. Conclusion and outlook

This work presented a novel approach for integrating multiple temporal aspects into a single graph-based visualization using multi-level graphs. This idea can be used (but is not limited to) for the complex graph-based visualization for KPI analysis where multiple time spans per analysis task and time steps between KPIs information have to be considered. Our proposal provides deep insight in temporal and non-local dependency information between KPIs assuming a valid model-graph in the back-end. The future work will include further research on the specification of common characteristics of all 'view-graphs' for proposing a minimal set of valid criteria for efficient processing of graph-based KPI analysis tasks.

Currently a validation of the concept, using a focus group with six experts from automotive industry, has been finished as a first part of a combined qualitative and quantitative research approach. First insights point to the necessity of an

enterprise wide valid KPI model for the creation of time-dependent 'view-graphs' as described in the paper to provide a common ground for the interpretation of dependencies between KPIs. The second part, a quantitative usability study is 'work-in-progress'. The study is part of a thesis and results will be expected soon.

Two questions concerning update mechanism and the granularity of time remain to be addressed: We waived the consideration about the handling of modifications to the 'model-graph' during run-time and the effects to the 'view-graphs'. In the scope of this issue, we will focus on a notification and modification inheritance concept, which can be integrated into the multi-level graphs. Furthermore our approach doesn't consider different granularity in time for each node yet. For the current state, we have implemented a uniform time-line per 'view-graph' because a secondary 'view-graph' with a different time-line might solve this issue. Nevertheless, multiple changes in the granularity of the time-line within the 'view-graph' are imaginable. For this matter we have to focus on a further refinement of the currently temporal uniformity of the 'view-graphs'.

6. Acknowledgment

Thanks to the anonymous reviewers for their extensive feedback on previous versions of this work. The work presented in this paper was supported by the RES-COM project funded by the Federal Ministry of Education and Research in Germany.

References

[ALA07] ARNOLD A., LIU Y., ABE N.: Temporal causal modeling with graphical granger methods. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2007), KDD '07, ACM, pp. 66–75. 3

[APP11] ARCHAMBAULT D., PURCHASE H. C., PINAUD B.: Difference map readability for dynamic graphs. In *Proceedings of the 18th international conference on Graph drawing* (Berlin, Heidelberg, 2011), GD'10, Springer-Verlag, pp. 50–61. 3

[BBD08] BURCH M., BECK F., DIEHL S.: Timeline trees: visualizing sequences of transactions in information hierarchies. In *Proceedings of the working conference on Advanced visual interfaces* (New York, NY, USA, 2008), AVI '08, ACM, pp. 75–82. 3

[BdMM08] BENDER-DEMOLL S., MORRIS M., MOODY J.: Prototype packages for managing and animating longitudinal network data: dynamicnetwork and rsonia. *Journal of Statistical Software 24*, 7 (5 2008), 1–36. 3

[BHPS12] BASOLE R. C., HU M., PATEL P., STASKO J. T.: Visual analytics for converging-business-ecosystem intelligence. *IEEE Computer Graphics and Applications 32* (2012), 92–96. 3

[DeS84] DESANCTIS G.: Computer graphics as decision aids: Directions for research. *Decision Sciences 15*, 4 (1984), 463–487. 3

[DGK01] DIEHL S., GOERG C., KERREN A.: Preserving the mental map using foresighted layout. In *Data Visualization 2001*, Ebert D., Favre J., Peikert R., (Eds.), Eurographics. Springer Vienna, 2001, pp. 175–184. 3

[DKM*13] DENNERT A., KRAUSE J., MONTEMAYOR J. A. G. I., HESSE S., LASTRA J. L. M., WOLLSCHLAEGER M.: Advanced concepts for flexible data integration in heterogeneous production environments. In *11th IFAC Workshop on Intelligent Manufacturing Systems (IMS 2013)* (2013). 5

[DT11] DOGANATA Y., TOPKARA M.: Visualizing meetings as a graph for more accessible meeting artifacts. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems* (New York, NY, USA, 2011), CHI EA '11, ACM, pp. 1939–1944. 3

[EB12] ELIAS M., BEZERIANOS A.: Annotating bi visualization dashboards: needs and challenges. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems* (New York, NY, USA, 2012), CHI '12, ACM, pp. 1641–1650. 3

[EHK*04] ERTEN C., HARDING P., KOBOUROV S., WAMPLER K., YEE G.: Graphael: Graph animations with evolving layouts. In *Graph Drawing*, Liotta G., (Ed.), vol. 2912 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 98–110. 3

[ELS91] EADES P., LAI W., SUGIYAMA K. M. K.: Preserving the mental map of a diagram. *Research Report Iias-rr-91-16e, International Institute For Advanced Study Of Social Information Science, Fujitsu Laboratories Limited,* (August 1991). 4

[FG90] FITZ-GIBBON C. T.: *Performance Indicators*. BERA Dialogues Series. Multilingual Matters Ltd, 1990. ISBN:1853590924. 2

[FSC99] FERNANDES SILVA S., CATARCI T.: Graphical interaction with historical databases. In *Scientific and Statistical Database Management, 1999. Eleventh International Conference on* (1999), pp. 184–193. 3

[FWSL12] FENG K.-C., WANG C., SHEN H.-W., LEE T.-Y.: Coherent time-varying graph drawing with multifocus+context interaction. *Visualization and Computer Graphics, IEEE Transactions on 18*, 8 (aug. 2012), 1330 –1342. 3

[GBD09] GREILICH M., BURCH M., DIEHL S.: Visualizing the evolution of compound digraphs with timearctrees. *Computer Graphics Forum 28*, 3 (2009), 975–982. 3

[GRC04] GOLFARELLI M., RIZZI S., CELLA I.: Beyond data warehousing: what's next in business intelligence? In *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP* (New York, NY, USA, 2004), DOLAP '04, ACM, pp. 1–6. 3

[Hil12] HILGEFORT I.: *Reporting and analysis with SAP BusinessObjects*, 2. ed., updated for release 4.0 fp3 ed. Galilep Pr., Bonn [u.a.], 2012. 3

[HTB*11] HORSFALL F., TANEV S., BONTCHEV B., GIGILEV T., GRUEV A.: Visualization of complex data relationships and maps: using the bloom platform to provide business insights. In *Proceedings of the 12th International Conference on Computer Systems and Technologies* (New York, NY, USA, 2011), CompSysTech '11, ACM, pp. 266–272. 3

[HVNK13] HESSE S., VASYUTYNSKYY V., NADOVEZA D., KIRITSIS D.: Visual analysis of performance indicators and processes in modern manufacturing. In *11th Global Conference on Sustainable Manufacturing (GCSM 2013), Berlin, Germany* (Berlin, Germany, September 2013), Seliger G., (Ed.), Technische Universitaet Berlin, Institut of Machine Tools and Factory Management, UniversitÃd'tsverlag der TU Berlin, pp. 455–460. 3

[HVRH13] HESSE S., VASYUTYNSKYY V., ROSJAT M.,

HENGSTLER C.: Modeling and presentation of interdependencies between key performance indicators for visual analysis support. In *Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services*, Emmanouilidis C., Taisch M., Kiritsis D., (Eds.), vol. 398 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2013, pp. 281–288. 3

[ISO14] ISO: *Automation systems and integration - Key performance indicators (KPIs) for manufacturing operations management - Part 2: Definitions and descriptions*. Tech. rep., International Organization for Standardization, 2014. 2

[Kei02] KEIM D. A.: Datenvisualisierung und data mining. *DatenbankSpektrum 2*, 1 (2002), 30–39. 3

[KKS*09] KEIM D. A., KOHLHAMMER J., SANTUCCI G., MANSMANN F., WANNER F., SCHAEFER M.: Visual analytics challenges. In *Proceedings of eChallenges 2009* (2009), p. 8. Echallenges 2009, Istanbul, Turkey. 3

[KN92] KAPLAN R. S., NORTON D. P.: The balanced scorecard - measures that drive performance. *Harvard Business Review January-February* (1992), 71–79. 2

[KNC*11] KHURANA U., NGUYEN V.-A., CHENG H.-C., WOOK AHN J., CHEN X., SHNEIDERMAN B.: Visual analysis of temporal trends in social networks using edge color coding and metric timelines. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)* (oct. 2011), pp. 549 –554. 3

[LBD07] LOUBIER E., BAHSOUN W., DOUSSET B.: Visualization and analysis of large graphs. In *Proceedings of the ACM first Ph.D. workshop in CIKM* (New York, NY, USA, 2007), PIKM '07, ACM, pp. 41–48. 3

[MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout adjustment and the mental map. *Journal of Visual Languages & Computing 6*, 2 (1995), 183 – 210. 4

[PS06] PERER A., SHNEIDERMAN B.: Balancing systematic and flexible exploration of social networks. *Visualization and Computer Graphics, IEEE Transactions on 12*, 5 (2006), 693–700. 3

[RC13] RAINER R. K., CEGIELSKI C. G.: *Introduction to information systems*, 4. ed., international student version ed. Wiley, Singapore ; Hoboken, NJ, 2013. 3

[RSB09] RODRIGUEZ R. R., SAIZ J. J. A., BAS A. O.: Quantitative relationships between key performance indicators for supporting decision-making processes. *Computers in Industry 60*, 2 (2009), 104 – 113. 2

[SDM12] SCHMIDT B., DOEWELING S., MÜHLHÄUSER M.: Interaction history visualization. In *Proceedings of the 30th ACM international conference on Design of communication* (New York, NY, USA, 2012), SIGDOC '12, ACM, pp. 261–270. 3

[SKM06] SCHRECK T., KEIM D. A., MANSMANN F.: *Regular treemap layouts for visual analysis of hierarchical data*. Spring Conference on Computer Graphics (SCCG'2006), Acm Siggraph, 2006, April 20-22, Casta Papiernicka, Slovak Republic,, 2006. 3

[SLN05] SARAIYA P., LEE P., NORTH C.: Visualization of graphs with associated timeseries data. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on* (2005), pp. 225–232. 3

[The06] THERÓN R.: Hierarchical-temporal data visualization using a tree-ring metaphor. In *Smart Graphics* (2006), Butz A., Fisher B., KrÃijger A., Olivier P., (Eds.), vol. 4073 of *Lecture Notes in Computer Science*, Springer, pp. 70–81. 3

[VHM*12] VASYUTYNSKYY V., HENGSTLER C., MCCARTHY J., BRENNAN K., NADOVEZA D., DENNERT A.: Layered architecture for production and logistics cockpits. In *Emerging Technologies Factory Automation (ETFA), 2012 IEEE 17th Conference on* (2012), pp. 1–9. 5

[WLR*11] WETZSTEIN B., LEITNER P., ROSENBERG F., DUSTDAR S., LEYMANN F.: Identifying influential factors of business process performance using dependency analysis. *Enterprise Information Systems 5*, 1 (2011), 79–98. 3

[Wu02] WU J.: Visualization of key performance indicators. http://www.information-management.com/news/5229-1.html?zkPrintable=true, May 2002. 2

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

Towards a Massively Parallel Solver for Position Based Dynamics

Marco Fratarcangeli and Fabio Pellacini

Sapienza University of Rome

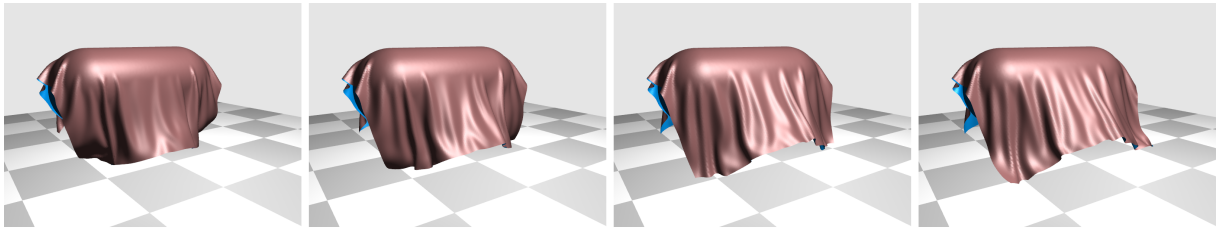


Figure 1: Interactive animations with our novel GPU-based implementation of the Position Based Dynamics approach: highly-detailed cloth model composed by 67K stretch constraints and 66K bending constraints.

Abstract

Position-based dynamics (PBD) is an efficient and robust method for animating soft bodies, rigid bodies and fluids. Recently, this method gained popularity in the computer animation community because it is relatively easy to implement while still being able to synthesize believable results at interactive rate. The animated bodies are modeled by using a large set of linearized geometrical constraints which are iteratively solved using a sequential Gauss-Seidel method on a single core CPU. However, when the animated scene involves a large number of objects, solving the constraints sequentially one after the other makes the computation of the motion too slow and not suitable for interactive applications. In this paper, we present a massively parallel implementation of position based dynamics which runs on the local GPU. In the initialization phase, the linearized geometrical constraints are divided in independent clusters using a fast, greedy coloring graph algorithm. Then, during the animation, the constraints belonging to each cluster are solved in parallel on the GPU. We employ an efficient simulation pipeline using a memory layout which favor both the memory access time for computation and batching for visualization. Our experiments show that the performance speed-up of our parallel implementation is several orders of magnitude faster than its serial counterpart.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically Based Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation and Virtual Reality

1. Introduction

The interactive animation of soft and rigid bodies is still a challenging problem for the Computer Graphics community. The final users expect outstanding quality and for this reason, the mathematical models defining the animated objects have become more and more sophisticated through the years.

Position-Based Dynamics (PBD) [MHHR06] is a widely spread method for interactively animating rigid bodies, soft bodies and fluids. Its popularity is due to its robustness, speed and easiness of implementation. In PBD, each object is modeled as a particle system, and the relationship between particles is expressed using geometrical constraints. There exists several kinds of constraints. For example, the *stretch* constraint imposes the distance between two particles to an input distance. Another example is the *tetrahe-*

dral constraint, which imposes the volume of the tetrahedron formed by four particles to an input value. The set of constraints associated to an object forms a system which must be solved for each animation frame.

The Gauss-Seidel method is particularly suitable for solving such systems of linear equations. However, the underlying algorithm is sequential and thus unsuitable for parallel implementations. Previous works exploited the sparsity of the system to extract parallelism [KW03], or proposed parallel approaches requiring synchronization primitives [CA09], or aimed at the general case of sparse vector-matrix products [WBS*13]. In this paper, we propose a parallel scheme for the systems of linear, geometrical constraints employed in Position Based Dynamics for modeling animated objects, taking advantage of commodity parallel local architectures such as multi-core CPUs and GPUs.

The main feature of the graphics processing units (GPUs) is their high computational throughput. They offer a great speed-up allowing the execution of thousands of lightweight threads in parallel according to the SIMD (single instruction multiple data) paradigm: the threads execute the same instruction on different data. The introduction of the NVIDIA's Compute Unified Device Architecture (CUDA) [KH13] allowed to abstract from the underlying graphics hardware and use GPUs for general purpose programming (GPGPU).

In the initialization phase of our method, the geometrical constraints are divided in independent clusters using a fast, greedy coloring graph algorithm. Then, during the animation, the constraints belonging to a cluster are solved in parallel on the GPU. We employ an efficient simulation pipeline using a memory layout which favor both the memory access time for computation and batching for visualization.

The presented method allows for real-time animations of complex deformable bodies. To demonstrate the performance gain in practice, we present animation of deformable bodies, like cloth and volumetric objects. We compare between implementations on the single core CPU, multi-core CPU and GPU. We observe that in some cases the performance speed up of the GPU solver is 10x w.r.t. the other platforms.

Our contributions:

- A novel algorithm which divide the set of constraints in independent clusters using a greedy coloring algorithm; all the constraints are then solved using a GPU-based parallel Gauss-Seidel method.
- A novel GPU data structure storing all the objects in a single particle system. The data structure is used for both the animation and visualization minimizing the frame update time.

This parallel implementation allows the animation of soft bodies composed by tens of thousands of constraints in real-time.

2. Related Work

Position Based Dynamics has been employed in a broad range of applications from knot simulation [KPFG07] to face animation [Fra12] and automatic body skinning [DB13, RF14]. Its original formulation considered just soft bodies, like cloths and inflatable balloons. Recently several works have been proposed to include both rigid bodies [DCB14] and fluids [MM13]. An extensive description of this method and its derivatives can be found in [BMOT13]. Popular available implementations are included, among others, in the open-source *Bullet* physics engine [Cou10] and in the proprietary *PhysX SDK* [NVI13].

One of the main issues of PBD is the intrinsic slow convergence of the employed serial Gauss-Seidel solver. The geometrical constraints are solved one by one several times in an iterative way (see Sec. 3). Each time an iteration is completed, the difference between the current solution and the optimal one decreases. In order to reach a satisfying solution, usually just a small number of iterations is needed $(2-4)$, which is suitable for interactive applications. However, for complex scenes where a substantial number of constraints is involved (e.g., several hundred of thousands), the convergence is too slow, the number of iterations increases and the performance is not suitable anymore for real-time animations.

In [MÖ8], a hierarchical *ad-hoc* position-based approach for clothes is devised in order to accelerate the convergence of the solver. In [BB08], a red-black parallel Gauss-Seidel schema is used for animating inextensible clothes using a force-based system. While providing excellent performance, this method is restricted to meshes with a regular grid topology. The mesh is subdivided into strips of constraints. The strips that have no common particle are independent from each other and can be solved in parallel. Both the solvers presented in [MÖ8] and [BB08] lack the generality needed to simulate generic, volumetric objects with arbitrary topology.

3. Position Based Dynamics

In the Position Based Dynamics approach, a soft body is represented by a set of N particles and a set of M constraints. Each particle i is defined by its position $\mathbf{p}_i \in \mathbb{R}^3$. A geometric constraint j is a mathematical relationship between particles $C_j(\mathbf{p}) = 0$. When an external force is applied to the particles, like the gravity, or some particles are displaced for some reason, e.g., by user manual interaction, the geometrical constraints may be not satisfied anymore. For each animation frame, the system of constraints must be always satisfied, or at least, the error between the current and the optimal solution must be small enough to produce believable motion.

During the simulation, if the particles configuration $\mathbf{p}^k$ in

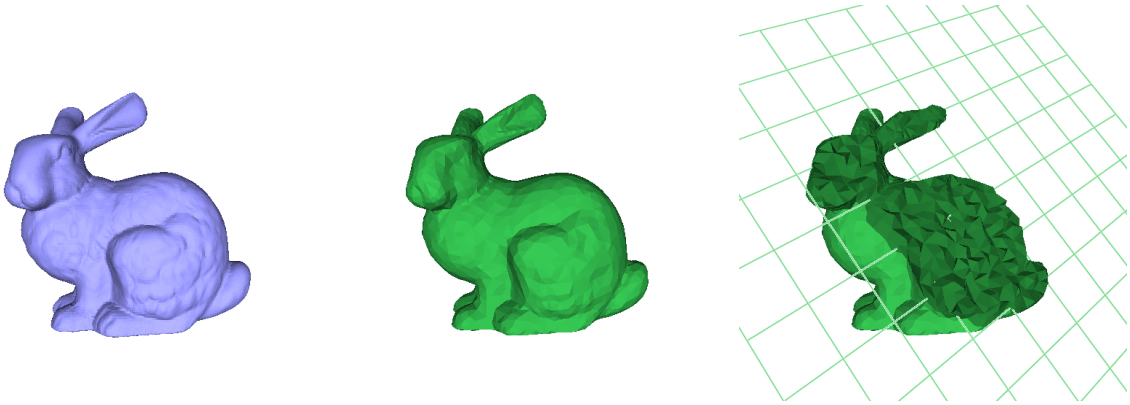


Figure 2: *Left:* An input surface mesh, composed by 34K vertices and 70 K faces. *Middle:* The corresponding tetrahedral mesh composed by 4K vertices and 16K tetrahedrals obtained using [BDP*02]. *Right:* Cross-section view showing the internal tetrahedrons of the input mesh.

a state k does not satisfy the set of constraints, then the iterative solver *projects* the particle positions in a valid state by finding a displacement $\Delta\mathbf{p}^k$ such that $C(\mathbf{p}^k + \Delta\mathbf{p}^k) = 0$. All the constraints in our method are functions $C_i(\mathbf{p}) = 0$. In this context, the constraints express a relationship (usually geometrical) between the particles. *Projecting* a set of particles according to a constraint means moving the particles such that their positions satisfy the constraint. The motion of the particles is computed inside a simulation loop. The particles are initially at rest state. This state can be perturbed by external conditions such as a force, like the gravity. The objective of the solver is to update the positions of the particles in order to keep the system of constraints satisfied.

Given $\mathbf{p}$, we want to find a correction $\Delta\mathbf{p}$ such that $C(\mathbf{p} + \Delta\mathbf{p}) = 0$. This system of non linear equations is *linearized*:

$$C(\mathbf{p} + \Delta\mathbf{p}) \approx C(\mathbf{p}) + \nabla_{\mathbf{p}} C(\mathbf{p}) \cdot \Delta\mathbf{p} = 0 \tag{1}$$

and then iteratively solved. If $\Delta\mathbf{p}$ is chosen to be parallel to $\nabla_p C(\mathbf{p})$ (which is perpendicular to rigid body modes), then both linear and angular momenta are conserved. For a full mathematical description on how to solve this system, the interested reader can refer to [MHHR06,BMOT13]. For consistency, we briefly report the equations for solving distance and volume constraints in the following section.

3.1. Geometric Constraints

Starting from an input mesh like the one depicted in Fig. 2, we create one particle $\mathbf{p}_i$ for each vertex, one stretch constraint for each edge (including the internal ones), and one volume constraint for each tetrahedron. These constraints are described in the following subsections.

3.1.1. Stretch Constraint

We define one *stretch* constraint for the particles $(\mathbf{p}_1, \mathbf{p}_2)$ at the end points of each edge of the mesh, including the edges of the internal tetrahedrons:

$$C(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d = 0 \tag{2}$$

where d is the rest length of the edge.

Given the configuration $\left(\mathbf{p}_1^k, \mathbf{p}_2^k\right)$ of two particles in the state k connected by a distance constraint, the corrections to the positions (Eq. 1) in order to satisfy a single constraint are:

$$\mathbf{p}_1^{k+1} = \mathbf{p}_1^k - \left(\left|\mathbf{p}_1^k - \mathbf{p}_2^k\right| - d\right) \frac{\mathbf{p}_1^k - \mathbf{p}_2^k}{\left|\mathbf{p}_1^k - \mathbf{p}_2^k\right|} \tag{3}$$

$$\mathbf{p}_2^{k+1} = \mathbf{p}_2^k + \left(\left|\mathbf{p}_1^k - \mathbf{p}_2^k\right| - d\right) \frac{\mathbf{p}_1^k - \mathbf{p}_2^k}{\left|\mathbf{p}_1^k - \mathbf{p}_2^k\right|} \tag{4}$$

3.1.2. Tetrahedral Volume Constraint

We define one *volume* constraint for the particles $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$ at the corners of each tetrahedral of the mesh:

$$C(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = \frac{1}{6}(\mathbf{p}_{2,1} \times \mathbf{p}_{3,1}) \cdot \mathbf{p}_{4,1} - V_0 \tag{5}$$

where $\mathbf{p}_{i,j}$ is the short notation for $\mathbf{p}_i - \mathbf{p}_j$ and V_0 is the rest volume of the tetrahedral. The projection of each particle belonging to a tetrahedron is:

$$\Delta\mathbf{p}_1^{k+1} = -\frac{s}{6} \cdot \left((\mathbf{p}_{2,1}^k \times \mathbf{p}_{3,1}^k) + (\mathbf{p}_{3,1}^k \times \mathbf{p}_{4,1}^k) + (\mathbf{p}_{4,1}^k \times \mathbf{p}_{2,1}^k)\right) \tag{6}$$

$$\Delta\mathbf{p}_2^{k+1} = \frac{s}{6}(\mathbf{p}_{2,1}^k \times \mathbf{p}_{3,1}^k) \tag{7}$$

$$\Delta\mathbf{p}_3^{k+1} = \frac{s}{6}(\mathbf{p}_{3,1}^k \times \mathbf{p}_{4,1}^k) \tag{8}$$

$$\Delta \mathbf{p}_4^{k+1} = \frac{s}{6}(\mathbf{p}_{4,1}^k \times \mathbf{p}_{2,1}^k) \tag{9}$$

where s is the scaling factor:

$$s = \frac{\frac{1}{6}(\mathbf{p}_{2,1} \times \mathbf{p}_{3,1}) \cdot \mathbf{p}_{4,1} - V_0}{\Sigma_{i=1}^4 \mid \nabla_{\mathbf{p}_i} C(\mathbf{p}_i) \mid^2} \tag{10}$$

and the gradients with respect to the particles are [MHHR06]:

$$\nabla_{\mathbf{p}_2} C(\mathbf{p}_2) = \frac{1}{6}(\mathbf{p}_{2,1} \times \mathbf{p}_{3,1}) \tag{11}$$

$$\nabla_{\mathbf{p}_3} C(\mathbf{p}_3) = \frac{1}{6}(\mathbf{p}_{3,1} \times \mathbf{p}_{4,1}) \tag{12}$$

$$\nabla_{\mathbf{p}_4} C(\mathbf{p}_4) = \frac{1}{6}(\mathbf{p}_{4,1} \times \mathbf{p}_{2,1}) \tag{13}$$

$$\nabla_{\mathbf{p}_1} C(\mathbf{p}_1) = -(\nabla_{\mathbf{p}_2} C(\mathbf{p}_2) + \nabla_{\mathbf{p}_3} C(\mathbf{p}_3) + \nabla_{\mathbf{p}_4} C(\mathbf{p}_4)) \tag{14}$$

4. Parallel Iterative Gauss-Seidel Solver

We consider scenes where thousands of constraints must be solved in real-time at least once every 16 ms to guarantee interactivity. During the animation, these constraints may be not satisfied due to external conditions, for example the user interacts with the model and move arbitrarily a set of particles, or an external force like gravity or wind is applied.

To solve the system, PBD employs a Gauss-Seidel solver. The constraints are solved iteratively one after the other, from the first to the last one. Then, the process starts over again and it is repeated a number of times, n_{its}. Increasing n_{its} leads to more precise solutions of the systems, sacrificing performance. Usually we employ a number of iterations between 2 and 24, depending on the topology of the system.

To speed-up the solving process, we implemented the Gauss-Seidel solver in a parallel fashion. We define a graph with a node for every constraint in the system. Two nodes of the graph are connected if the corresponding constraints share at least one particle. Each color corresponds to a cluster of constraints. We solve all the constraints belonging to a cluster in parallel: we instantiate a thread for each constraint within the same cluster. This way, the system is solved in less steps than the sequential approach. Fig. 3 depicts this mechanism for a simple mesh composed by stretch constraints (Fig. 3.*Left*). The corresponding colored graph is shown on the right together with the corresponding clusters, one for each color.

The graph coloring problem, in its simplest form, involves the assignment of colors to each node in the graph, such that two connected nodes do not share the same color. In other words, given a graph $G(V,E)$ and a set S of colors, a proper coloring is a map $c : V \to S$ s.t. $\forall \langle u,v \rangle \in E, c(u) \neq c(v)$.

Finding the minimal amount of colors for coloring a generic graph G (the *chromatic number*) is known to be NP-hard [GJ79]. Usually, efficient greedy heuristics are employed to find an approximate solution. A widely-known approach is the following: let $v_1, v_2, \ldots, v_n$ be an ordering of the vertices of the graph $G = (V,E)$, for $k = 1,2,\ldots,n$ the sequential algorithm assign v_k to the smallest possible color. In general, an arbitrary ordering may perform very poorly but it is possible to show that, for any G, there exists at least one ordering of vertices for which the sequential algorithm produces an optimal coloring.

In our system, we used the *smallest-last* ordering defined in [MB83, CM83], which guarantees a coloring with at most

$$max\{1 + \delta(G_0) : G_0 \text{ is a subgraph of G }\} \tag{15}$$

colors where $\delta(G_0)$ is the smallest degree of the vertices in G_0.

5. Implementation

During the design of an algorithm for the GPU, it is critical to minimize the amount of data that travels on the main memory bus. The time spent on the bus is actually one of the primary bottlenecks that strongly penalize the performance [nvB13]. In fact, the transfer bandwidth of a standard PCI-express bus is 2-8 GB per second, while the internal bus bandwidth of a modern GPU is approximately 100-150 GB per second.

We designed our system in order to minimize the amount of data which travels on the PCI bus and keep the data on the GPU as much as possible. In the initialization phase, we load all the data required for the animation on the video memory. Then, during the animation phase, we update the data structures directly on the GPU. In this way, the CPU is not involved in the animation process (besides being responsible for calling the GPU kernels), and any data exchange using PCI bus is avoided.

In order to advance the animation step, the current state of the particle system must be stored in the video memory of the GPU. The current state is given by the following attributes:

1. the current position of each particle;
2. the previous position of each particle;
3. the external forces influencing the particles;

A different static array is created for each attribute during the initialization phase. Then, each array is loaded into video memory. Each array stores the attributes for all the particles. The size of each array is equal to the size of an attribute (4 floating-point values) multiplied by the number of the particles.

We employ the so-called *ping-pong* technique [Dro07] that is particularly useful when the input of a simulation step is the outcome of the previous one, which is the case in most

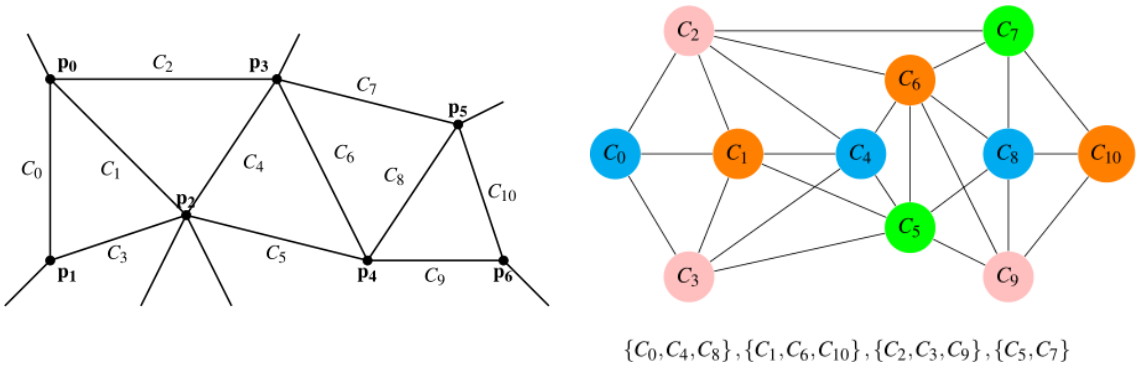


Figure 3: *Left.* A connected mesh of particles. Each edge corresponds to a stretch constraint. *Right.* Constraint graph. Each node corresponds to a constraint and two nodes are connected if the corresponding constraints share at least one particle. Nodes of the same color forms a cluster of constraints which can be solved in parallel.

of the animations based on particle systems. The basic idea is rather simple. In the initialization phase, two buffers are loaded on the GPU for each attribute, one buffer to store the input of the computation and the other to store the output. When the computation ends and the output buffers are filled with the results, the pointers to the two buffers are swapped such that in the following step, the previous output is considered as the current input.

The current results data are then stored in a Vertex Buffer Object (VBO), which is employed to render the current state of the deformable object; in this way the data never leaves the GPU achieving maximal performance. This mechanism is illustrated in Fig. 4.

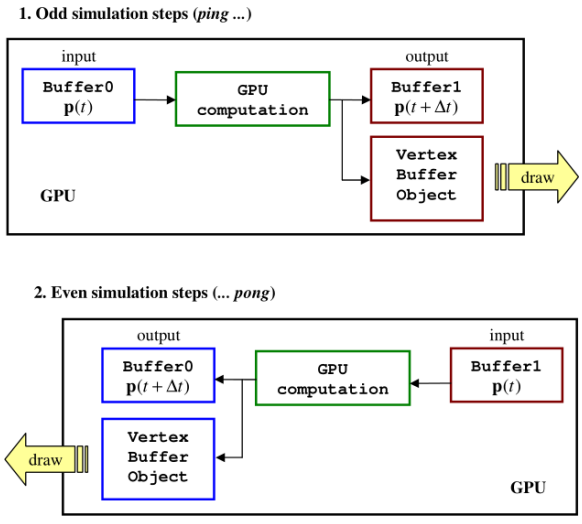


Figure 4: The ping-pong technique on the GPU. The output of a simulation step becomes the input of the following step. The current output buffer is mapped to a Vertex Buffer Object for visualization.

6. Results

We tested our proposed technique on three different scenes (Fig. 5). The first scene represents fixed cloth modeled using stretch and bending constraints under the influence of wind. The second scene represent a stack of clothes which falls under the gravity force and collides with a capsule. The third scene represents a volumetric character modeled using both stretch and tetrahedral volume constraints and deformed by user intervention. The tetrahedral meshes used in the experiments are obtained using [BDP*02].

To advance the animation, we used a 10 ms time step and 16 iterations per frame. We measured the time performances on the set of test scenes on a mass-market laptop equipped with an Intel i7 2.30 GHz processor (4 cores), 4GB RAM and a graphics card GeForce 610M with one multiprocessor and 2 GB VRAM.

We implemented standard Position Based Dynamics on a single core CPU, and the parallel version both on multithread CPU and GPU. The code of the kernels for solving the constraints is the same for each kind of platform. We used Intel Thread Building Blocks technology for implementing the multithreaded version on the CPU. The mean computation times are reported in Table 1. The corresponding animations are shown in the accompanying video.

7. Conclusions

In this paper, we introduced an early implementation of a massively parallel solver which can be used to speed up the computation of the popular Position Based Dynamics approach. It allows to simulate soft bodies, both cloths and volumetric ones, and it can be easily extended to involve rigid bodies, fluids and the two-way interactions among them. Table 1 shows the computational superiority of the GPU (even with a single multiprocessor!) against single and multi core CPUs.

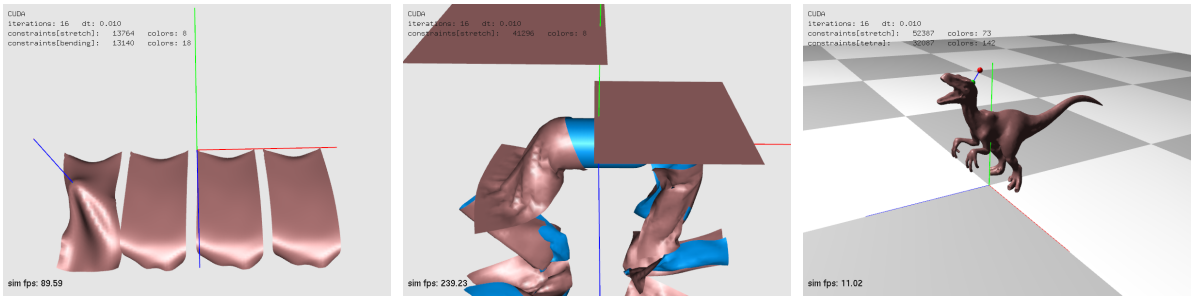


Figure 5: Test beds for our experiments. The number of constraints involved into the animation is reported in Table 1. *Left.* Fixed cloths under the influence of wind. Cloths are modeled with stretch and bending constraints. *Middle.* Cloths falling under the gravity force and colliding a capsule. *Right.* Volumetric deformation of a volumetric body.

	particles	constraints/colors			iterations	avg. computation time [ms]		
		stretch	blending	tetrahedral		CPU Single	CPU Multi	GPU
scene 0	4800	13764/8	13140/8	/	16	151.6	102.1	40.6
scene 1	14400	41296/8	/	/	16	104.2	48,8	9,7
scene 2	10452	52387/73	/	32087/142	16	256.4	122,1	82.0

Table 1: Quantitative results of our experiments on three different scenes depicted in Fig. 5. For each scene, it is reported the number and type of constraints. For each type of constraint, it is reported the number of colors, which corresponds to the independent clusters in which the constraints are grouped. We run the animations on a single core CPU, a multi core CPU and a GPU with a single multiprocessor and report the average computation time over 500 frames.

The performance of the GPU solver is bounded by the number of times the kernels are run, rather than the number of particles involved into the animation. This is depicted in the case of *scene 1* in Table 1, where the performance speed-up of the GPU is higher than in the other cases because, despite the big number of particles, the number of clusters is smaller than in the other cases.

A kernel call is required for solving each cluster. Each kernel is run a number of times equal to the number of clusters multiplied by the number of iterations. The number of clusters is equal to the number of colors required to color the graph constraint. The problem of maximizing the performance can be thus expressed as finding the minimal number of colors required to cover the graph, which is known to be NP-hard [GJ79].

We noted that the minimal number of colors is lower bounded by the size of the biggest clique in the constraint graph. In the future, we would like to explore the possibility to color the constraint graph with an arbitrarily low number of colors, allowing minimal changes in the topology, in order to maximize the performance on the GPU.

References

[BB08] BENDER J., BAYER D.: Parallel simulation of inextensible cloth. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (November 2008). 2

[BDP*02] BOISSONNAT J.-D., DEVILLERS O., PION S., TEILLAUD M., YVINEC M.: Triangulations in cgal. *Comput. Geom. Theory Appl. 22*, 1-3 (May 2002), 5–19. 3, 5

[BMOT13] BENDER J., MÜLLER M., OTADUY M. A., TESCHNER M.: Position-based methods for the simulation of solid objects in computer graphics. In *EUROGRAPHICS 2013 State of the Art Reports* (2013), Eurographics Association. 2, 3

[CA09] COURTECUISSE H., ALLARD J.: Parallel dense gauss-seidel algorithm on many-core processors. In *High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on* (June 2009), pp. 139–147. 2

[CM83] COLEMAN T. F., MORE J. J.: Estimation of sparse jacobian matrices and graph coloring problems. *Journal of Numerical Analysis 20* (1983), 187–209. 4

[Cou10] COUMANS E.: *Bullet 2.76 physics sdk manual*, 2010. 2

[DB13] DEUL C., BENDER J.: Physically-based character skinning. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (2013). 2

[DCB14] DEUL C., CHARRIER P., BENDER J.: Position-based rigid body dynamics. In *Computer Animation & Social Agents (CASA)* (2014). accepted. 2

[Dro07] DRONE S.: Real-time particle systems on the gpu in dynamic environments. In *ACM SIGGRAPH 2007 Courses* (New York, NY, USA, 2007), SIGGRAPH '07, ACM, pp. 80–96. 4

[Fra12] FRATARCANGELI M.: Position-based facial animation synthesis. *Computer Animation and Virtual Worlds 23*, 3-4 (2012), 457–466. 2

[GJ79] GAREY M. R., JOHNSON D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. 4, 6

[KH13] KIRK D. B., HWU W.-M. W.: *Programming Massively*

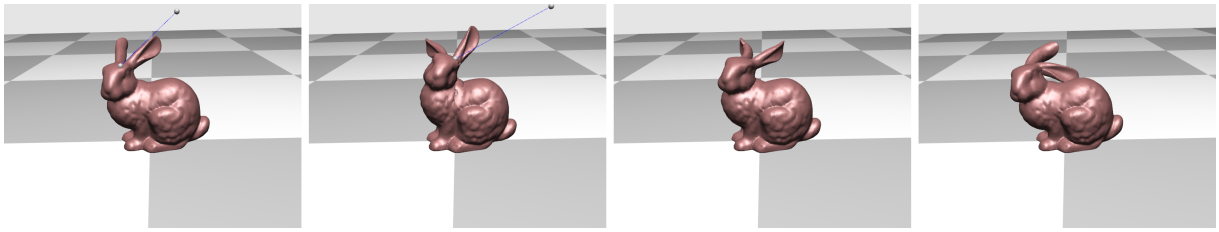


Figure 6: volumetric deformation of the Stanford Bunny model, composed by 80K stretch constraints and 50K tetrahedral constraints.

Parallel Processors: A Hands-on Approach, 2 ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2013. 2

[KPFG07] KUBIAK B., PIETRONI N., FRATARCANGELI M., GANOVELLI F.: "A Robust Method for Real-Time Thread Simulation". In *ACM Symposium on Virtual Reality Software and Technology (VRST)* (New Port Beach, CA, USA, November 2007), ACM, (Ed.). 2

[KW03] KRÜGER J., WESTERMANN R.: Linear algebra operators for gpu implementation of numerical algorithms. *ACM Trans. Graph. 22*, 3 (July 2003), 908–916. 2

[MÖ8] MÜLLER M.: Hierarchical position based dynamics. In *Virtual Reality Interactions and Physical Simulations (VRIPhys2008)* (Grenoble, November 2008). 2

[MB83] MATULA D. W., BECK L. L.: Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM 30*, 3 (July 1983), 417–427. 4

[MHHR06] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Madrid, November 6-7 2006). 1, 3, 4

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (July 2013), 104:1–104:12. 2

[nvB13] : *NVIDIA CUDA Compute Unified Device Architecture - Best Practices Guide*, version 5.5 ed., 2013. 4

[NVI13] NVIDIA: *Physx sdk 9.13 documentation*, 2013. 2

[RF14] RUMMAN N. A., FRATARCANGELI M.: Position-based skinning. In *Eurographics/ACM Spring conference on Computer Graphics (SCCG)* (2014). accepted. 2

[WBS*13] WEBER D., BENDER J., SCHNOES M., STORK A., FELLNER D.: Efficient gpu data structures and methods to solve sparse linear systems in dynamics applications. *Computer Graphics Forum 32*, 1 (2013), 16–26. 2

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

Fire fighting and related simulations in a CAVE using off-the-shelf hardware and software

Frank Poschner

Department of Computer Engineering
Faculty of Electrical Engineering and Computer Science
University of Kassel, Germany

Abstract

The growing interest in using Serious Games in education has also reached the field of fire brigade training. By the use of simulations, firefighters can train in virtual worlds on the computer and learn about processes and prepare real fire drills. It is for example possible to extinguish a virtual fire or to set up a water supply in such simulations. Played on a common computer and with input devices like mouse and keyboard, the games normally do not offer a very realistic impression on the user. For an immersive experience, those virtual training games must be used in Virtual Reality environments like a Cave Automatic Virtual Environment (CAVE), which is usually associated with high costs. In this paper the setup of a CAVE used for simulations in the field of fire brigade training will be presented as well as the use of input devices and their coupling with real firefighting equipment. Since almost exclusively low-cost and commercially available elements such as common projectors, the Microsoft Kinect and the Nintendo Wii controllers were used, at the same time an approach for bringing simulations like these into VR environments without large costs is demonstrated. As the presented simulations are developed with a free but powerful Game Engine, a wide range of possibilities for the development of training games for a CAVE environment is given.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture—Input devices, I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism —Virtual reality

1. Introduction

A fire has broken out in a garbage can on the street. How should a civilian behave now, what would a firefighter do? The matching behavior of the latter are taught and trained in the exercises of fire brigade training.

The use of games as a basis for training and learning can be basically described with the expression of "Serious Games" [MC05]. Such games exist in many different types, special attention is recently given to training simulations on the computer. They are widely used in different educational sectors like health and safety, but even in political education [Pre01].

The use of such games is also of interest in the field of fire brigade training and for teaching in preventive fire protection. The effective support of education through preparation, follow-up and support of practical exercises is tried to be achieved here. Furthermore, the objective of saving time of presence by giving the trainees the opportunity to be trained in the practical processes and their rules by simulations on a personal computer at home or even on mobile devices is pursued. Expenses in time, money and organisation can be saved. Such Serious Games can be of very different types to the point of those consisting of complex, virtual worlds in which avatars of the player can be independently moving in an environment to solve problems and maybe even being connected to others via network [ES14]. Simulations like these belong to the area of Virtual Reality applications [Rhe91]. The player should be able to immerse himself into the virtual world, also to support the learning effect. It is known that the creation of immersion for a player goes hand in hand with a realistic representation of the game scene on matching displays and the use of ergonomically sensible and appearingly realistic input devices. It would therefore be most useful for the player of a simulation in the field of fire brigade training, when he gets the impression of a game

scene in a realistic size and could use realistic equipment. A solution would be to use a CAVE (Cave Automatic Virtual Environment) and input devices that create the impression of real equipment to the user.
Operating such a CAVE means to set up a well cooperating configuration of hardware and software to link multiple projection surfaces or screens to a virtual environment for an immersive impression of the user. Originally, the operation of a CAVE required specially developed hardware and software. In particular, the operation of parallel computer systems, the use of appropriate software and the connection of special input devices was often necessary. This special equipment usually made it a costly affair to operate a CAVE and would therefore not meet the needs of the fire brigades. In Chapter 2 of the presented work the operation of CAVEs will be discussed in general as well as related works that benefit from the progressive development in the field of gaming technology by using commercial products in a CAVE. In the further chapters a solution for the development of a CAVE with 3 walls is presented as well as simulators from the field of fire brigade training and preventive fire protection which were developed for the CAVE with a freely available game engine. Furthermore, the use of input and output devices that are meant to create the impression of realism to the player is described. For example, the Microsoft Kinect and the Nintendo Wii controller, which was coupled with a fire extinguisher, were used. Almost all elements presented here are common products in the field of game industry and game development, so that this work also represents a proposal for the low-cost design of a VR environment.

2. Related work

The concept of a CAVE was first presented in 1992 by Carolina Cruz-Neira et al. [CNSD*92] as an environment for virtual reality applications. Based on this proposed approach many CAVEs were built, which were equipped with special hardware and software configurations. During that time, the hardware required for a CAVE such as projectors, the computers and graphics hardware for the processing of the images as well as the matching input devices was expensive. Magnetic or optical tracking systems were used for recognizing the user's movement while the interaction with virtual objects in the scene could for example be realized by the use of data gloves. Special software such as VRJuggler [VR 14] was developed to support parallel rendering, image processing and the use and management of input devices. Due to the enormous development in the field of computer and graphics hardware, computer games and game consoles as well as input devices for gaming there are new ways for the use of these technologies by including them into CAVE systems. Jung et al. [JKS11] use the Microsoft Kinect, which is originally an optical input device with depth sensor for the Microsoft Xbox console, or even two of them [JS12] to track the motions of the user for navigating a virtual scene by gestures and even for grabbing objects. For the process of grabbing itself a Nintendo Wii controller is used. Settgast et al. also use the kinect for navigating in an "'affordable"' CAVE-like environment and compare its use for navigation to that of a joystick [SLBF12]. Furthermore a work for the integration of the Wii controller into VR applications is presented, in which the Wii is used for different purposes like playing drums or recognizing the user's gestures [HPL07]. While gestures can be used for navigation like raising an arm for stepping forward as shown by Jung et al., this might not be satisfying from an ergonomic point of view. One attempt for a more realistic way of using input devices for walking and running that uses common games hardware is presented by Hilsendeger et al. [HBTF09]. The Nintendo Wii Balance Board which was originally built for fitness applications is used here. The player is able to navigate a scene by standing and stepping on the buttons of the board. There are new attempts in this field of navigation like the Virtuix Omni [Vir14], which is again specially designed VR hardware and therefore expensive or not yet available. As one goal of our work was to build CAVE simulators with low cost components, we used almost common hardware only. In a similar project a CAVE was built and connected to planning simulations in the field of factory planning [Ins13].
This work presents solutions for firefighting and related simulations in a CAVE. There are many simulations in this area, which are typically developed for personal computers or mobile devices. One of those is a simulation which is used as a basis for our work and has been developed in the project "'KATIE"'. It is a network game in which the player can cooperate with others in a virtual environment. He can also use virtual equipment of fire brigades for certain operation purposes [Pos13].

3. CAVE setup

In this chapter the setup of the CAVE used for fire fighting simulations is described. This covers a short description of the basic hardware components as well as the idea and the steps of how to create the CAVE view.

3.1. Hardware components

The hardware components consist of two different parts: The CAVE frame and the components used for processing the CAVE images.

3.1.1. The frame

The most basic elements of the CAVE are their walls and screens. The CAVE has three walls which are ordered rectangular and have a size of 2.5m x 2.5m each. The frame of each wall was built out of wooden pillars between which canvas for back projection was stretched, the left part of figure 1 shows the CAVE out of action. As the building process itself is not of interest at this point it can be reviewed in the

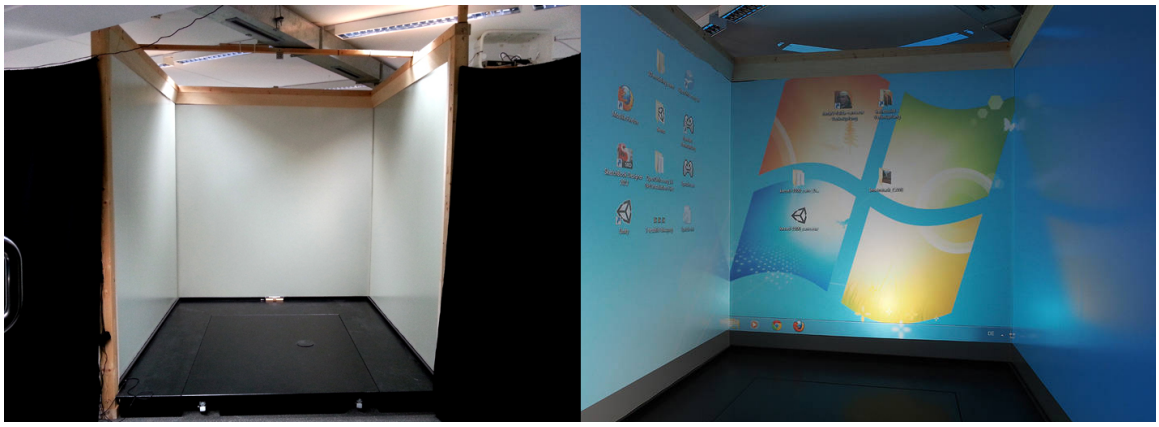


Figure 1: *Left: CAVE turned off. Right: CAVE showing a desktop view.*

thesis under which it was developed [Wol02]. All the described solutions could also be used on a single wall or a projection screen.

3.1.2. Components for image processing

As already pointed out only standard components are used for operating the CAVE. In former times typically a lot of computers had to be connected for a good performance in a CAVE. For this CAVE only one computer is used, which is a normal PC with an Intel i7 processor, 12GB of RAM and an Asus P6T S1366 mainboard. Three graphics boards of the type NVIDIA GeForce GTX 285 with average performance are used.
Three projectors of the brand Optoma are used for the image projection. The fact that the targeted image resolution can be represented in 120Hz is important here. In our case this resolution is 1024 x 768. As the CAVE walls are quadratic we would need a resolution that has the same width as height. This is normally not supported by the projectors available and so 256 pixels of each projection width are cut off by considering this in our simulation. The projectors need to be placed in a certain distance to the CAVE walls, in our case this is 2,965 meters. In case the room of the CAVE is not big enough, mirrors for reflecting the images can be used [Wol02].
For the stereoscopic rendering an active stereo system with shutter technology developed by NVIDIA is used, which is described in the next section.
In order to bring 3D sound into the CAVE an active 5.1 speaker system from "'Logitech"' is used, which is connected to the built-in sound card of the computer mainboard.

3.2. CAVE view and stereoscopic rendering

To create a CAVE view, as many images as the CAVE has walls have to be projected in a way that the images are seamlessly combined. For this reason we combine our three graphics boards as an SLI combination connecting each projector to one card. NVIDIA offers the matching technology for their cards in the graphics driver, which is named "'NVIDIA surround"'. While this works with monitor displays it is also possible to connect projectors. The configuration is a simple process in the supplied system control. The right side of figure 1 shows the CAVE with working NVIDIA surround showing a Windows desktop.
The next step is to establish stereoscopic rendering in the CAVE. As already mentioned, the technology of active stereo with shutter glasses is used here. NVIDIA offers a USB infrared emitter which has to be connected to the computer and sends shutter signals to the matching shutter glasses [NVI14]. This system can only be used with NVIDIA graphics boards and represents a simple and low-cost solution if the user doesn't want to implement the stereoscopic rendering himself. The NVIDIA driver recognizes the start of a 3D application like a game and renders the images ready for shuttering. There are parameters that can be set like the depth of the 3D effect. As all these components work together well in the CAVE, serious games can be developed or adapted for the use in this environment, which is described in the next chapter.

4. Developing CAVE simulations with a free available game engine

This chapter describes the use of a game engine for creating simulations for the presented CAVE system. Since the generation of simulations with variable and rich content was always a difficult task in conjunction with CAVEs and their specialized software, it was our goal to try using today's gaming technology in this regard. The approach was to combine simulations developed with the game engine Unity [Uni14] with the setup of a CAVE like it is described in chapter 3.
Unity is a free game engine with a wide range of functions

for the development of games and simulations. The basic idea was to be able to use Unity simulations that have been developed for a variety of devices in the CAVE. This chapter describes the necessary adjustments for the CAVE view. Necessary adjustments for the use of special input devices are discussed in Chapter 5. The CAVE installation includes

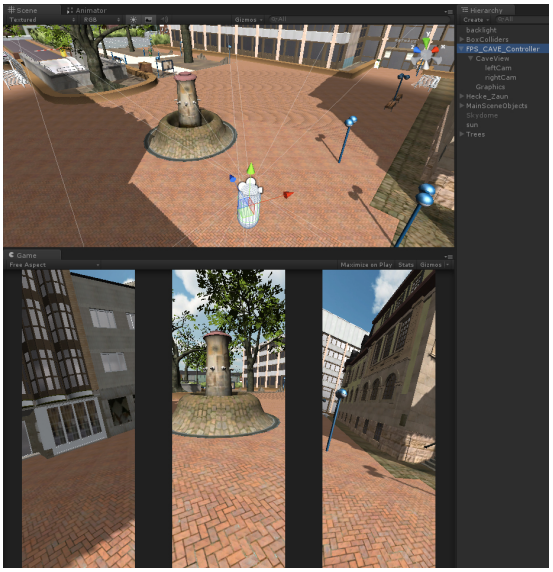


Figure 2: *Development view in Unity for CAVE simulations.*

three square screens with rear projection. A game scene must be represented accordingly by three images in a suitable arrangement. For this purpose, three virtual cameras must be used in the game scene and be structured to fit and work with the appropriate viewport. Therefore a package of Unity elements is created, which represents a controller to the corresponding cameras. This can be used for navigating a Unity game scene. If other controllers or representatives of the user have to be developed for the CAVE, this controller must be adjusted accordingly. The camera settings and positions can then be applied.

Figure 2 shows a picture of the First Person Controller in a game scene and the arrangement of the cameras for the CAVE view. In the Hierarchy view the hierarchy of the used components can be seen. The GameObject "FPS_Cave_Controller" at the highest level is responsible for controlling the FPS object, this must be adapted for an adaptation of the controller to other needs. The attached GameObject "CaveView" has a script "CAVE", which is responsible for setting the camera values at runtime. The attached GameObjects "leftCam" and "rightCam" are also generated at runtime, which can be seen in the screenshot. "CaveView" itself is responsible for the camera of the center wall.

Regarding the arrangement of the CAVE screens at right angles and taking into account that seamless image transitions between the screens have to be created, the field of view of the cameras has to be adjusted. It must also be noted that supernatants of the images would arise on the lateral borders of the screens, because of their square format and the restrictions in the resolution settings of the projectors. Those can only provide images in either 16:9 or 4:3 format. Since 4:3 format will be used in the CAVE, the camera images have to be cropped by using black stripes to prevent picture supernatants between the images when using the full height of the screens. Figure 2 also shows the game view of the scene at runtime using the described stripes.

The camera producing the image of the left wall must be rotated by -90 degrees about the y-axis to that of the center wall, while the camera for the right wall must be rotated by 90 degrees to the camera of the center wall. In figure 3 the corresponding settings for the Unity camera that captures the image for the right CAVE wall are shown. While the settings of the clipping planes can be adjusted to the different simulations, the field of view should be 90 degrees for each camera. The height of the Normalized View Port Rect should be 1 for all cameras, the width of the viewport should be 25 percent each, which was determined in tests in the CAVE but can also be calculated. To generate the stripes described earlier, the x-component of the viewport, which describes the starting point of the camera picture from the left to the right border of the screen, depends on the respective camera position: There must be black stripes to the left and the right of each wall. This results in six similar stripes, which means a width of $(100 - (3 * 25))/6$ for each of the stripes. The

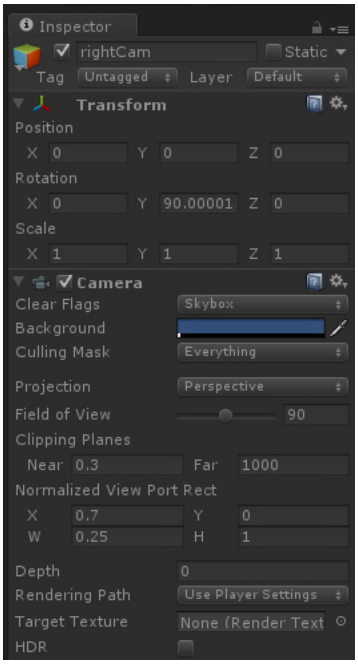


Figure 3: *Exemplary settings for the Unity camera that produces the image for the right CAVE wall.*

camera of the left wall starts with the width of one of these stripes as offset in the x-component of its viewport, while the center camera starts with an offset of 0.25 plus three times the width of a stripe. Figure 3 shows the right cam starting at 0.70 due to some rounding, which is compensated by the position of the projector.
The Unity Input Manager should be used to operate the FPS_CAVE_Controllers by Mouse, Keyboard, Joystick and XBOX360 controller.
Sound in the CAVE results directly from the use of sound in Unity, the settings of the Windows sound driver and the sound hardware. Unity offers a range of settings for audio sources, surround sound is possible and must be set in Unity, too.
To link CAVE simulations via network with other applications, for example to develop a multi-player mode, the Unity network functions can be used. For this purpose the CAVE computer must be connected with an appropriate network, as it is common for PCs with a Windows operating system.
Applications for the CAVE must be exported as a PC standalone application from Unity3D with the projector's resolution while the width is taken times 3. In our case the resolution is 3072 x 768.

5. Input devices

An important reason for using Unity simulations in a CAVE is the ability to use various commercial but low-cost input devices in simulations. This chapter describes some of the connected devices and how they are used. Of course standard devices like mouse, keyboard and common joysticks can be used, especially when connected wireless. Although this may be not of great interest, it should still be mentioned.

5.1. Using Microsoft Kinect for head-tracking

Detecting the position of the player in a CAVE can give developers the possibility of directly reacting on the human player's physical behavior in a VR application and connect him directly to the virtual world. Especially the detection of the position of his head has an important meaning, because the eye position of people can be important for the proper recognition of 3D effects. In Unity the player looks into the virtual world through a camera. If he bends down, the camera can also be moved down in the same amount like he moved his head. Thus, the user will get a more realistic impression of the scene.
Originally expensive special hardware was needed for this head-tracking. This often had to be fixed to the head of the player. An example is the "Flock of birds"', a magnetic position detection system [Fif14]. Since the release of the Microsoft Kinect, a camera system that has a depth sensor and can be connected to a PC, the detection of movements of a player can also be done with this system. It is connected via USB and a matching SDK to use the system with Unity is provided. The advantages of using the Kinect are the elimination of body-worn parts for head-tracking and the possibility of tracking and distinguishing between different body parts, not just the head.
In our CAVE the Kinect is mounted on the open side of the CAVE on the opposite of the center wall at a height of 2.80 m, by that it is behind the usual sight of the CAVE user. Since the Kinect has a specified minimum distance of 1.80 m to the tracked person and because of negative results in tests with lens attachments to reduce this distance, a fixture at another location was out of the question because of the height of the CAVE frame. The distance to the users would be too short and the angle at which the camera focuses the users would be too large to grant a reliable recognition of the user's head. By the actual mounting the tracking of movements, carried out by the player towards the center screen in front of his body, is not possible though. In general, the sense of using the Kinect is mainly directed to head-tracking, although body-tracking is also possible. Figure 4 shows the current position of the Kinect in the CAVE.
The Kinect is connected to the CAVE computer via USB.

Figure 4: *The Kinect sensor mounted on the rear end of the CAVE.*

The following problem may appear with the use of the Kinect: If many USB devices are connected to the CAVE computer, it may lead to utilization of the computer's USB bandwidth. The utilization of the Kinect is classified as very high. Symptomatic of this is a high latency in the motion detection by the Kinect, which results in jittering in the movements of the tracked player. Since the CAVE projection is not supported by external light, but only the light from the projector itself is used, there may be an insufficient illumination of the CAVE in simulations with dark scenes. Tracking errors may occur due to the Kinect then, and the player's position in the simulation might not be affected any more. On Windows operating systems the Kinect can be used by installing the Microsoft Kinect SDK which is used here, too. To use the Kinect in Unity, a matching software package must be integrated. A package of " Carnegie Mellon University " has been used successfully here [Car14].

5.2. Nintendo Wii controller and real equipment

The Nintendo Wii Controller or Wii Remote is a wireless input device, which was originally developed for the Nintendo Wii console. It has a number of buttons, a directional

pad and acceleration sensors, which make it possible to detect the movement of the player who holds the controller. Furthermore, a goal can be targeted, which is done via an infrared source to be sighted. The controller can be extended with the so-called Nunchuck controller, which has an analog stick, two buttons and also acceleration sensors. Since the Wii controller can be connected to a PC, it is also suitable as an input device for PC-based games. It is used as an input device with the extension of the Nunchuck controller via bluetooth connection in the CAVE. In order to use the Wii controller with Unity, a suitable framework must be used. The free framework UniWii has been used here.
The advantage of using the Wii controllers in a CAVE is the possibility of attaching it to real equipment. It can, for example, be coupled to a fire extinguisher. It's acceleration sensor and the possibility of targeting infrared sources allow the use of sensor data in Unity simulations. For example the user can target a virtual fire in a simulation by using the hose of the fire extinguisher. Subsequently a button of the controller, which is attached to the appropriate place of the fire extinguisher is pressed and a virtual jet of water can be released, which triggers the deletion of the virtual fire. Another possibility would be the attachment to an axe to simulate the hammering of a door or to use it with a jet pipe for firefighting. Figure 5 shows the coupling of the Wii controller to a real but empty fire extinguisher. An ergonomically sensible position was determined first and then the controller was tied to the extinguisher by using cable ties.
Since the Wii controller also has an analog stick and different buttons, it can be used for navigating a simulation scene and thus make the use of further controllers unnecessary.

Figure 5: *Using the Wii controller with a fire extinguisher.*

5.3. A Gamepad for navigation and handling

Gamepads can also be used as input devices in the CAVE. They can be integrated by using the Unity Input Manager and have their advantages in the easy and ergonomic operation, particularly for the rapid handling of equipment or to navigate through scenes. In the case of the presented CAVE an Xbox360 controller is used, which can be operated via the standard drivers in Windows. The mentioned FPS_CAVE_Controller can be controlled intuitively using the joysticks and buttons on the gamepad.

6. Simulations

The various input devices described in Chapter 5 were used and tested in various simulations for the training of fire brigades and fire protection, which is described in this chapter.

6.1. Turntable ladder vehicle tutorial

To use a turntable ladder vehicle requires an extensive knowledge. In order to explain the basics of controlling the ladder a Serious Game was developed. It allows the user to train in the CAVE and includes a user guide with explanations. It begins with explanations to the use of the vehicle and ends with the rescue of a man from the roof of a house. The game is controlled by the user via gamepad to interact with the car controls, but also head-tracking using the Kinect is implemented. The users get a realistic impression of the virtual world, because the environment reacts on their behavior. The games is played in a first-person mode, but also a mode for 3rd-person operation was implemented. In the 3rd person mode the movements of the avatar that represents the player are tracked, too. This was only used as a test of the Kinect, because the appearance of the player's avatar in the simulation is not necessary: The player should feel as a part of the virtual world and not see his body virtually represented. Figure 6 shows the simulation in the 3rd person view with the avatar in front of the ladder vehicle. It is shown in the Unity development view while the user raises his arm. The Kinect works well in this simulation, sometimes there can be a few jittering effects due to the light conditions. The basis for this simulation were extracted from the literature and discussions with firefighters. It is planned to make it available for testing by experts from the fire brigades soon.

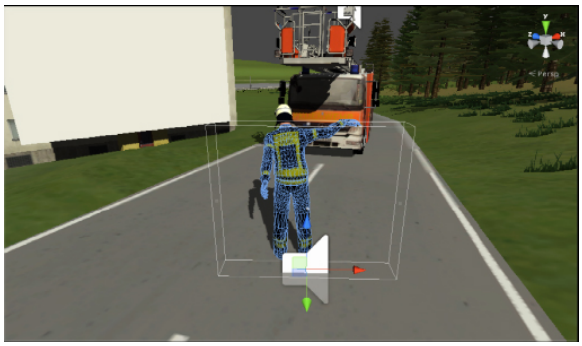


Figure 6: *User is tracked by a Kinect in the simulation of a turntable ladder vehicle.*

6.2. Fire in a residential building

In this simulation, the Nintendo Wii controller is used. At the start the player is located in a bathroom of a residential building and can navigate in first-person mode with the joysticks and buttons of the Wii controller, that is mounted to the fire extinguisher controller. He can open doors and needs to find a fire that has broken out somewhere in the building. The game is equipped with instructions for the player. He has to take a fire extinguisher from the basement of the virtual house and extinguish the fire with it by using the extinguisher like he would do in the real world. A particle system simulates the extinguishing media and interacts with the virtual fire if used correctly: The fire extinguisher can be operated by targeting the corresponding IR-emitter with the Wii controller. This simulation shows small accuracy problems when targeting the IR emitters, but generally the integration of fire extinguishers and Wii controller and the connection to the Unity simulation work well.

6.3. Using a turntable ladder vehicle and deletion of fires in an industrial area

In this simulation, the player starts in an industrial area, in which a turntable ladder vehicle and a burning fire tank can be found. At the beginning the player's movements are controlled by the Xbox360 controller. The user has to find a fire extinguisher and take it. When he takes the virtual fire extinguisher the controls change to the Wii controller, mounted to the real fire extinguisher: The analog joystick of the Nunchuck can be used for walking while also the infrared sensor can be used to change direction. Also the player can change his line of sight by using the Wii controller to point in a certain direction so he can target a fire. The game contains a tutorial: The fire at the tank truck must be deleted first, then a turntable ladder vehicle must be operated correctly. The user can drive the ladder to a fire on the roof of a hall, being in the basket of the ladder. That fire has to be extinguished, too. Figure 7 shows a user in the CAVE extinguishing the fire at the tank truck using the fire extinguisher in combination with the Wii controllers. The simulations where the Wii controller is used were tested by a fire safety expert of the fire brigades Kassel and classified as appropriate for use in preventive fire protection. A revision and expansion of the content to different scenarios is in development. It can be noted that the use of the Wii controllers for navigation and targeting a fire has been implemented successfully and works well in this simulation .

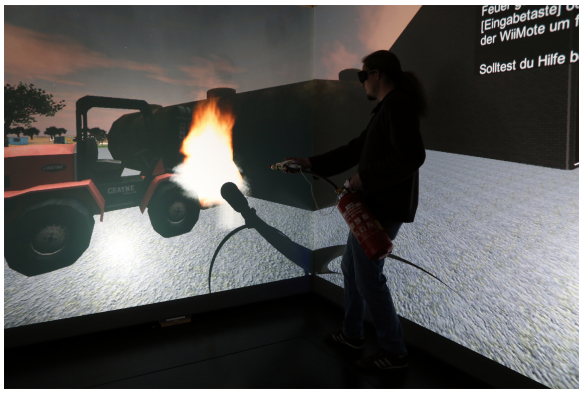


Figure 7: *Extinguishing a virtual fire with real equipment.*

6.4. Using virtual equipment and setting up a water supply

As already pointed out there exists a simulation that was used as a basis for creating applications in the CAVE with the presented setup. By using the Xbox360 controller virtual equipment of fire brigades can be used for certain operation purposes here, like setting up a water supply. The simulation itself is described in [Pos13]. The contents of the simulation was developed in cooperation with experts from the County Fire Services Association Fulda. A PC version which is controlled via mouse and keyboard was evaluated and revised by intensive tests with firefighters of voluntary fire brigades. However, the CAVE version shows ergonomic problems being controlled by the gamepad, especially if the user is inexperienced. While not all operable objects of the simulation can be represented by real equipment, the usability of the presented input devices comes to a limit here and a more natural interface has to be found. In figure 8 two people can be seen using the simulation in the CAVE.

7. Conclusions and future work

In this paper we have presented the work of developing simulations for the field of fire brigade training in a CAVE, using common hardware and software like the Nintendo Wii game controller and a free game engine. With the game engine

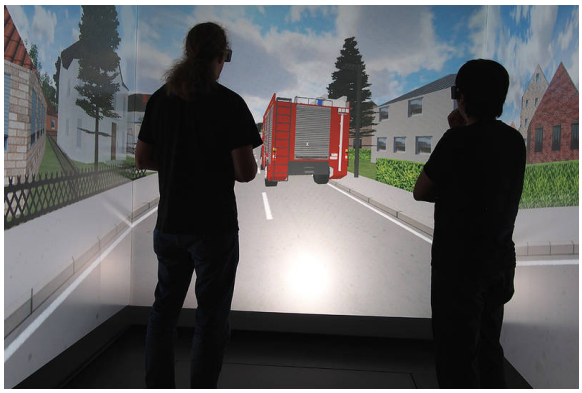


Figure 8: *A simulator for fire brigade training.*

Unity it is possible to create simulations that can be run using various input devices in the CAVE. With the solution shown, it is also possible to generate stereoscopic projections on 3 CAVE walls for the applications. Furthermore, equipment like a fire extinguisher is coupled with the controllers and used in a way that real equipment becomes an input device. As the hardware and software used here is relatively inexpensive, this work can serve as a template to build affordable CAVE systems. With this approach also VR systems with a single projection are conceivable. Different kinds of simulations have been presented to prove that the range of use of this system is wide.

In the future simulations could be developed that include all the input devices in a sensible way as the integration of all the presented elements in one simulation has not yet been accomplished. This is mainly for the reason that problems such as the lack of USB bandwidth must be solved for the simultaneous use of all devices. Furthermore, a solution for the problem of moving in a way that simulates real movement should be developed in the CAVE. While head-tracking is already possible by using the Microsoft Kinect, to track walking and running with it would lead the user out of the range of the CAVE and so at present the Xbox360 controller or the Nintendo Wii controller are still used for navigation.

It is also planned to develop input devices with sensors based on components like Arduino products [Ard14] to use real equipment of for example fire brigades as input devices. Much better ergonomics could be the result as the small sensors could be integrated into the equipment much more inconspicuous and at the same time provide results that are more precise.

References

[Ard14] ARDUINO: Arduino - products, 2014. http://arduino.cc/en/Main/Products (28.04.2014). 8

[Car14] CARNEGIE MELLON UNIVERSITY: Microsoft kinect Ű microsoft sdk, 2014. http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Microsoft_SDK (26.04.2014). 5

[CNSD*92] CRUZ-NEIRA C., SANDIN D. J., DEFANTI T. A., KENYON R. V., HART J. C.: The cave: Audio visual experience automatic virtual environment. *Commun. ACM 35*, 6 (June 1992), 64–72. 2

[ES14] E-SEMBLE: Xvr in general, 2014. http://www.e-semble.com/en/Products/XVR/In_general/ (27.04.2014).

[Fif14] FIFTH DIMENSION TECHNOLOGIES: Ascension flock of birds, 2014. http://www.5dt.com/products/pfob.html (26.04.2014). 5

[HBTF09] HILSENDEGER A., BRANDAUER S., TOLKSDORF J., FRÖHLICH C.: Navigation in virtual reality with the wii balance board. 6. Workshop der GI-Fachgruppe VR/AR, Aachen, Germany: Shaker Verlag GmbH. 2

[HPL07] HAMMERL S., PREUSS T., LATOSCHIK M. E.: Wiinc - wii network control - einsatz des wii-controllers für vr-anwendungen. Virtuelle und Erweiterte Realität, 4. Workshop of the GI VR & AR special interest group, pp. 141–148. 2

[Ins13] INSTITUT FÜR PRODUKTIONSMANAGEMENT UND LOGISTIK: Virtuelle fabrikplanung in der 3d-cave (vifa 3d), 2013. http://www.i-p-l.de/leistungen/cave (28.04.2014). 2

[JKS11] JUNG T., KROHN S., SCHMIDT P.: Ein natural user interface zur interaktion in einem cave automatic virtual environment basierend auf optischem tracking. *Workshop 3D-NordOst* (2011), 93–102. 2

[JS12] JUNG T., SIMON F.: Interaktion in einem cave automatic virtual environment unter verwendung mehrerer tiefensensorkameras. *3D-NordOst* (2012), 199–208. 2

[MC05] MICHAEL D. R., CHEN S. L.: *Serious Games: Games That Educate, Train, and Inform*. Muska & Lipman/Premier-Trade, 2005. 1

[NVI14] NVIDIA: Nvidia 3d vision Ű the best stereoscopic 3d gaming experience for pcs, 2014. http://www.nvidia.co.uk/object/3d-vision-main-uk.html (27.04.2014). 3

[Pos13] :. *Eine virtuelle Multi-User Trainingsumgebung für die Feuerwehrausbildung* (2013), Fraunhofer Verlag. 2, 7

[Pre01] PRENSKY M.: *Digital Game-Based Learning*. McGraw-Hill, 2001. 1

[Rhe91] RHEINGOLD H.: *Virtual reality*. Summit Books, 1991. 1

[SLBF12] SETTGAST V., LANCELLE M., BAUER D., FELLNER D. W.: Hands-free navigation in immersive environments for the evaluation of the effectiveness of indoor navigation systems. In *VR/AR* (2012), Geiger C., Herder J., Vierjahn T., (Eds.), Shaker, pp. 107–118. 2

[Uni14] UNITY TECHNOLOGIES: Unity, 2014. http://www.unity3d.com (28.04.2014). 3

[Vir14] VIRTUIX: Virtuix omni, 2014. http://www.virtuix.com/ (28.04.2014). 2

[VR 14] VR JUGGLER: vrjuggler - vr juggler provides a cross-platform virtual reality application development framework, 2014. http://code.google.com/p/vrjuggler/ (26.04.2014). 2

[Wol02] WOLF M.: *Planung und Aufbau einer immersiven Multiprojektionsumgebung*. Master's thesis, University of Kassel, 2002. 3

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

Calibration of Depth Camera Arrays

Razmik Avetisyan, Malte Willert, Stephan Ohl and Oliver Staadt

Visual Computing Lab, University of Rostock, Germany

Abstract

Depth cameras are increasingly used for tasks such as 3-D reconstruction, user pose estimation, and human-computer interaction. Depth-camera systems comprising multiple depth sensors require careful calibration. In addition to conventional 2-D camera calibration, depth correction for each individual device is necessary. In this paper, we present a new way of solving the multi depth-camera calibration problem. Our main contribution is a novel depth correction approach which supports the generation of a 3-D lookup table by incorporating an optical marker-based tracking system. We verify our approach for the Microsoft Kinect and for the MESA Swiss-Ranger4000 time-of-flight camera.

Categories and Subject Descriptors (according to ACM CCS): I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—Camera Calibration

1. Introduction

The popularity of multi-camera acquisition setups in combination with depth cameras brought up new challenges for camera calibration.

A system with only one camera usually has several drawbacks. Due to the limited field of view, a single camera is not suitable for capturing large scenes, and when using one camera image, the scene and objects are visible only from one side. Furthermore, scene reconstruction suffers from object occlusion problems. Incorporating multiple neighboring depth cameras can overcome these limitations by using images from different viewpoints. The matching correspondence between different depth images can be used to achieve higher accuracy in measurements. Moreover, a good calibration of the camera system is the key requirement to achieve a high correspondence between the cameras.

Despite the camera registration, there is also particular need to calibrate the depth reported by a sensor in order to attain more accurate results. In numerous experiments [BKKF13, MCAPL13, MBPF12, SSBH11] it was observed that low-cost commodity depth cameras suffer from measurement inaccuracies when capturing over larger distances. As a result, the recorded datasets from different cameras cannot be matched to each other properly, thus, introducing inaccuracies in the reconstructed results.

In this paper, we offer a new depth correction approach incorporating an optical tracking system. Unlike other approaches [BKKF13, MBPF12], based upon complex mechanical setups for moving the camera, our approach incorporates the benefits of a tracking system. This facilitates the camera calibration in place, which means, that the cameras can remain fixed in their desired setup during the entire calibration procedure. Thus, intrinsic and extrinsic parameters possibly estimated beforehand remain valid. Another important advantage of our method is, that it allows to perform the depth correction of multiple cameras at once.

2. Related Work

There are three different categories of the calibration: (1) the intrinsic and extrinsic calibration, (2) intrinsic depth calibration, and (3) extrinsic calibration between color and depth cameras. Intrinsic calibration refers to estimating the internal camera parameters, for instance focal lengths. External parameters, such as the location and orientation of the camera, are determined in the extrinsic calibration To fully solve the camera calibration problem these tasks have be combined.

Intrinsic and extrinsic calibration: The basic idea of all approaches is to find the parameters by the help of an object with known geometrical properties. A large number of

algorithms use a planar checkerboard target for the calibration. The approaches in [AZD13, HCKH12, MCAPL13, MBPF12, Wen12] for example use the checkerboard for the intrinsic calibration. However, homography computed from the checkerboard pattern provides very suitable constrains for both intrinsic and extrinsic calibration. Extrinsic calibration approaches based on a checkerboard target are presented in [MBPF12, Wen12]. Liu et al. [LFZL12] describe an intrinsic calibration based on a cross shaped checkerboard target. Here, the authors use the missing corners of the checkerboard to visualize the error. Stürmer et al. [SSBH11] present an approach for aligning multiple ToF cameras based on a cubic reference object of known size. With the knowledge about angular and distance relations of the cubic sides the transformations between those cameras are calculated. Alexiadis et al. [AZD13] use a 1-D target based extrinsic calibration approach. Here, the authors establish point correspondences across all cameras for performing a pairwise stereo calibration based on epipolar geometry. Macknojia et al. [MCAPL13] describe a plane feature-based extrinsic calibration approach. The method consists of finding a normal vector and the center of the target plane for estimating the relative orientation and translation between the cameras. Another plane-based method is presented by Auvinet et al. [AMM12]. Their procedure involves moving a large rectangle in front of two cameras simultaneously. The intersection from three detected planes in each camera view is then used for constructing a virtual point. Having a cloud of virtual points synchronized in all camera views, the authors calculate the external relations between the cameras. Beck et al. [BKKF13] describe an extrinsic calibration approach based on a tracked box as reference. The transformation of each camera is estimated with respect to the constructed coordinate system of the box, which is the result of intersecting the detected box planes in the depth image.

Intrinsic depth calibration: The depth values reported by depth cameras are usually inaccurate. To correct these distances several approaches use manually measured data.

Maimone et al. [MBPF12] present a depth calibration approach based on a laser range finder. The authors fix the camera on a sliding rail and move it towards a wall. For every fixed interval they measure the real distance from the camera to the wall with a laser range finder. These measured distances along with the camera recorded depth values are used to determine a linear depth correction function. A more precise depth correction approach based on a 3-D lookup table is proposed by Beck et al. [BKKF13]. For generating this kind of table, a motor equipment is used, which slowly lowers the camera setup from the ceiling of the room towards the floor. During this process the lookup table is filled with tracking system measured and camera captured distances to the even floor. The resulting 3-D table provides a quite accurate per-pixel and per-distance mapping at every pixel in the depth image. An approach that does not need a mechanical pre-processing is offered by Herrera et al. [HCKH12]. They get improved depth values by adding a special varying offset to the distorted disparity, which decays exponentially with the increase in disparity.

Extrinsic calibration between color and depth cameras: Color to depth calibration is a quite well known problem, where the external relation between color and depth cameras has to be found. Kreylos [Kre13] replaces the planar checkerboard with a semi-transparent one. For this kind of target, checkerboard corners become visible in the color and in the depth image, which allows him to calibrate both kinds of cameras simultaneously with respect to each other, without need of exchanging the target-type. Macknojia et al. [MCAPL13] capture a fixed checkerboard target by both the color and IR image of Kinect camera simultaneously. They use a classical calibration method proposed by Zhang [Zha00] for finding the external relations. Other approaches like [HCKH12, LFZL12] calibrate cameras with respect to the same reference frame independently and then relate them to each other.

3. Our Calibration Approach

Depth correction is the main challenge for today's calibration approaches. Most cases use a correction table based on manually measured data. The drawback of the existing approaches is their incapability to generate this table for the cameras fixed in the desired setup. Furthermore the process usually needs special, quite elaborate hardware.

For this reason, we propose a new approach for correcting the depth values of a depth measuring device. The basic idea is to incorporate a 6DOF marker-based tracking system for measuring distances. The whole calibration procedure is separated into three steps. First, we carry out our new method of depth correction. Second, checkerboard-based intrinsic and extrinsic calibration between color cameras is followed. Finally, we execute extrinsic calibration between the color and depth cameras.

3.1. Depth calibration

To acquire tracking data we use a rigid body attached to a traditional checkerboard target (Fig. 1).

During the calibration procedure the rigid body is located by a tracking system, while at the same time the checkerboard is located in the camera images. Combining these two kinds of information allows us to calibrate the camera.

For every position of the checkerboard in space the system is capable to deliver coordinates (X_O, Y_O, Z_O) and orientation R of the rigid body with respect to our tracking coordinate system. We align the rigid body with the checkerboard target in such a way that the position of the body exactly matches with the center of the checkerboard, and the rotation of it

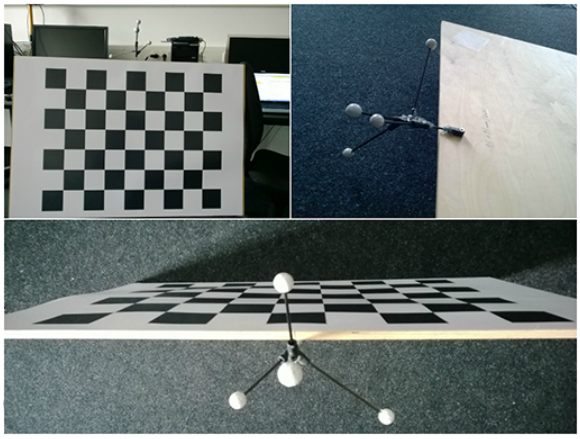


Figure 1: *The rigid body attached to the checkerboard.*

indicates the rotation of the checkerboard. To achieve this center adjustment, we use the tracking system's capability to specify offsets for the reported trackable coordinates.

While moving the target in space we can now take advantage of the reported 6DOF coordinates as follows:

We multiply the checkerboard center position with the rotation matrix R to rotate the checkerboard to its original orientation.

$$\begin{bmatrix} X_{R(O)} \\ Y_{R(O)} \\ Z_{R(O)} \end{bmatrix} = R \begin{bmatrix} X_O \\ Y_O \\ Z_O \end{bmatrix} \tag{1}$$

where X_O, Y_O, Z_O are the coordinates of the checkerboard center in the tracking coordinate system, R is the rotation matrix and $X_{R(O)}, Y_{R(O)}, Z_{R(O)}$ are the rotated coordinates of the checkerboard center. Having the coordinates of the rotated center, means having the checkerboard positioned orthogonal to the z-axis of the tracking coordinates system. By knowing the size of the edge of a checkerboard square it becomes possible to calculate 3-D coordinates $(X_{R(i)}, Y_{R(i)}, Z_{R(i)})$ of all inner corners by the following equations:

$$X_{R(i)} = X_{R(O)} + \left(i \bmod w - \left\lfloor \frac{w}{2} \right\rfloor \right) \alpha \tag{2}$$

$$Y_{R(i)} = Y_{R(O)} + \left(\left\lfloor \frac{h}{2} \right\rfloor - \frac{i}{w} \right) \alpha \tag{3}$$

$$Z_{R(i)} = Z_{R(O)} \tag{4}$$

where w, h are the number of inner corners in horizontal and vertical directions respectively, α is a constant indicating the size of the edge of a checkerboard square and $(X_{R(i)}, Y_{R(i)}, Z_{R(i)})$ are the coordinates of the i-th corner ($i = 0...[w \times h - 1]$). We count i starting from the top left corner of the checkerboard and continue row by row, column by column with an increment by one unit. Once the coordinates of the corners are calculated, we multiply all points with the inverse of the rotation matrix in order to obtain their real coordinates in the tracking coordinate system.

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = R^{-1} \begin{bmatrix} X_{R(i)} \\ Y_{R(i)} \\ Z_{R(i)} \end{bmatrix} \tag{5}$$

Since our tracking coordinate system does not coincide with the coordinate system of the camera, an extrinsic calibration has to be done to find the relation between those two coordinate systems. After obtaining this relation, we translate the 3-D positions of all checkerboard corners (X_i, Y_i, Z_i) to the camera coordinate system with the help of the following relation:

$$\begin{bmatrix} X_{K(i)} \\ Y_{K(i)} \\ Z_{K(i)} \end{bmatrix} = R_K \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + T_K \tag{6}$$

where R_K is the 3×3 rotation matrix and and T_K is the translation vector between those two coordinate systems respectively and $(X_{K(i)}, Y_{K(i)}, Z_{K(i)})$ are translated and rotated coordinates of the checkerboard corners. Having the coordinates of all checkerboard corners, we calculate the distances between the camera frame and each corner point according to the following equation:

$$D_{K(i)} = \sqrt{X_{K(i)}^2 + Y_{K(i)}^2 + Z_{K(i)}^2} \tag{7}$$

For each position of the checkerboard in space we have to measure $w \times h$ distance values. In general, this can be done easily, if a semi-transparent checkerboard is used. However, for the cameras used in this work, we provide a simpler approach that uses the IR images of the Kinect or SwissRanger4000 camera, respectively. We localize the inner corners of the checkerboard in the IR image in order to extract their positions in image coordinate space. As we know the relation between IR and depth cameras, we can map the corners onto the depth image in a quite accurate way.

After obtaining pairs of real measured distances and camera recorded distances, we generate a 3-D lookup table. The size of the table corresponds to the resolution of the depth image $T[resx, resy, raw]$ and the device depth sensitivity level as a third dimension. This makes it possible to map raw values to measured values for all pixels along the

depth image. To fill the lookup table we use the following equation:

$$T[x, y, depth(x, y)] = D_{K(i)} \tag{8}$$

where (x, y) are the pixel coordinates of i-th corner of checkerboard, $depth(x, y)$ is a depth measured by the camera and $D_{K(i)}$ is a distance of that point measured by a tracking system.

For each depth camera we construct a separate lookup table. To fill the lookup table and obtain precise depth mapping it is necessary to move the checkerboard in front of each camera with the objective to cover the whole volume in front of the device. Depending on the dimension of the depth image, the procedure may take several minutes to fill the lookup table with the real distance values. While moving the target in front of the camera, we make use of our visualization software providing real-time feedback, so that the user knows which parts of the volume are calibrated and which parts are not appropriately covered yet.

3.2. Intrinsic calibration and Extrinsic calibration

First, the intrinsic calibration procedure includes estimation of camera specific parameters for the RGB and the IR sensors. Such parameters include the focal length (f_x, f_y), the image center (u_0, v_0) and the lens distortion coefficients. To carry out this intrinsic calibration, we use a planar checkerboard pattern of a size 9×7. In average 15-20 images per camera were enough to ensure sufficient calibration quality. After obtaining the necessary amount of images with the checkerboard, we apply the standard calibration routine based on checkerboard homography. By doing so, the intrinsic parameters of the camera along with the distortion coefficients are estimated accurately.

Second, the procedure of extrinsic calibration involves positioning the checkerboard target in the overlapping region of two cameras. Having the target visible in both camera frames enables us to find the relative position and orientation for each of them. The standard routine for external calibration takes the already estimated intrinsic parameters along with the distortion coefficients to find the translation vector and rotation matrix for each camera pair.

3.3. Extrinsic calibration between color and depth cameras

Although the geometrical relation between the IR and RGB sensors of the Kinect is already embedded in the device internal memory by the manufacturer, it still may vary from one camera to another. For some tasks it is very important to have a more precise knowledge regarding to the external relation of these cameras. This is also true when we combine a depth camera, like the SwissRanger4000, with an external color camera, because in this setting we have no given knowledge about the geometrical relation. To find the relation, we fix the checkerboard target in front of the device in about 1.5 meters and capture it by both sensors. To enhance the detection rate for the IR image we apply a Median smoothing filter with the aperture size of 5 pixels.

After identifying the checkerboard corners on both images, we apply a standard routine for stereo calibration.

4. Prototype and Experiments

In the upcoming subsection a brief overview of hardware setup is presented. Then we discuss our prototype used for this work.

4.1. Experimental setup

The hardware setup of our system involves two kinds of depth cameras, the Kinect and the MESA SwissRanger4000, which are calibrated using 6DOF marker-based tracking system.

4.1.1. Depth cameras

The subject of our study consists of two Microsoft Kinect sensors and one MESA SwissRanger 4000 positioned in the room of a size about 4×4 meters. All cameras are positioned in such a way that they are pointing to the center of the working volume. The sensors are mounted and remain fixed during the entire calibration procedure. To avoid time and data synchronization issues over the network and make the data acquisition more simple we connect the depth cameras to a single portable PC.

4.1.2. Tracking system

For the depth correction we use a 6DOF marker-based tracking system from OptiTrack. Six V100:R2 cameras, which have a maximum latency of 10 ms, are surrounding the room and covering the calibration volume. The system is connected to the separate PC. The calibration of the tracking system is carried out beforehand, based on the routines provided by the manufacturer.

The two PCs used in the experiments are connected through a network. The tracking system machine streams the tracking data in real-time.

4.2. Prototype

Our prototype is a command line tool that combines the standard routines of OpenNI and OpenCV libraries for acquiring and processing RGB, depth and IR images in real-time. After collecting the necessary amount of images with a checkerboard target, we execute a standard routine for corner detection on both RGB and IR images. For IR images

the corner detection procedure is slightly unstable due to noise introduced by the IR projector. To overcome this problem, some authors suggest to block the IR projector with an overlapping mask and instead use powerful incandescent lamps [MCAPL13]. Another method for reducing the noise is to apply a smoothing filter [MCAPL13]. In our approach we apply a Median filter of a size of 5×5 pixels to reduce the noise and without blurring the images too much. To further enhance the detection rate we turned on the external IR lighting of the OptiTrack system. Fig. 2 illustrates enhanced IR image of a Kinect camera.

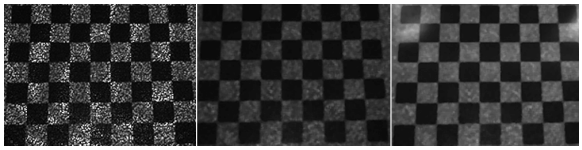


Figure 2: *The IR images of Kinect 1.) The noise IR image illuminated with IR projector 2.) Blurred IR image 3.) Blurred IR image with external IR lighting of the OptiTrack system.*

Since the SwissRanger 4000 has a relatively low resolution (176×144) compared to the Kinect, the standard checkerboard corner detection approach failed. This is due to the small size of the checkerboard visible in the amplitude image. Starting from 3 meters the detected corners were supposed to be very hard distinguishable, and from 3.5 meters onwards they were unrecognizable at all. To address this problem we scaled the amplitude images up to 3 times. Although the resulting images are a little bit blurry, the corner detection procedure works satisfying even on larger distances. Another problem with this device is the poor illumination of the target. For this reason, we increased the integration time of the camera up to 150 ms. This significantly enhancing the quality of images (Fig. 3), as the pixels could acquire more light.

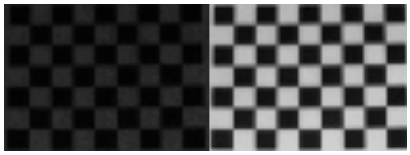


Figure 3: *The scaled amplitude image of SwissRanger4000. Left: Image captured with the default integration time. Right: Image captured with the integration time 150 ms.*

For our depth calibration procedure we make use the detected corners of the checkerboard in the IR image for finding corresponding corner coordinates in the depth image. According to Macknojia [MCAPL13] the offset between the IR image and the depth image are given by the following simple equations:

$$depth_x = ir_x + 5 \tag{9}$$

$$depth_y = ir_y + 4 \tag{10}$$

where ir_x and ir_y are the pixel coordinates in IR image, and $depth_x$ and $depth_y$ are the mapped pixel coordinates in depth image. The resulting images from Kinect and Mesa SwissRanger 4000 are presented in the Fig. 4 and Fig. 5 respectively.

Figure 4: *Left: The blurred IR image of Kinect camera. Right: Detected corners that are localized in the depth image.*

Figure 5: *Left: The amplitude image of SwissRanger 4000. Right: Detected corners that are localized in the depth image.*

4.3. Lookup table visualisation

Visualizing the 3-D depth lookup table is an essential step in the overall depth correction procedure. It allows the user to gain deeper knowledge regarding the status of the calibrated volume. Moreover, during the procedure of generating the table, the visualization tool may decrease the time needed for covering the whole volume by just providing a real-time feedback about which parts are filled and which parts still remain. This avoids of re-filling already covered parts.

To generate 3-D lookup table for the Kinect camera, we have to consider that the depth images of this device have 11 bits per pixel. Under the fact that one bit is reserved for invalid values only 10 bit remain. Hence, it is sufficient to have $2^{10} = 1024$ as the size of the third dimension. The resulting table has a size of $640 \times 480 \times 1024$. The presented lookup

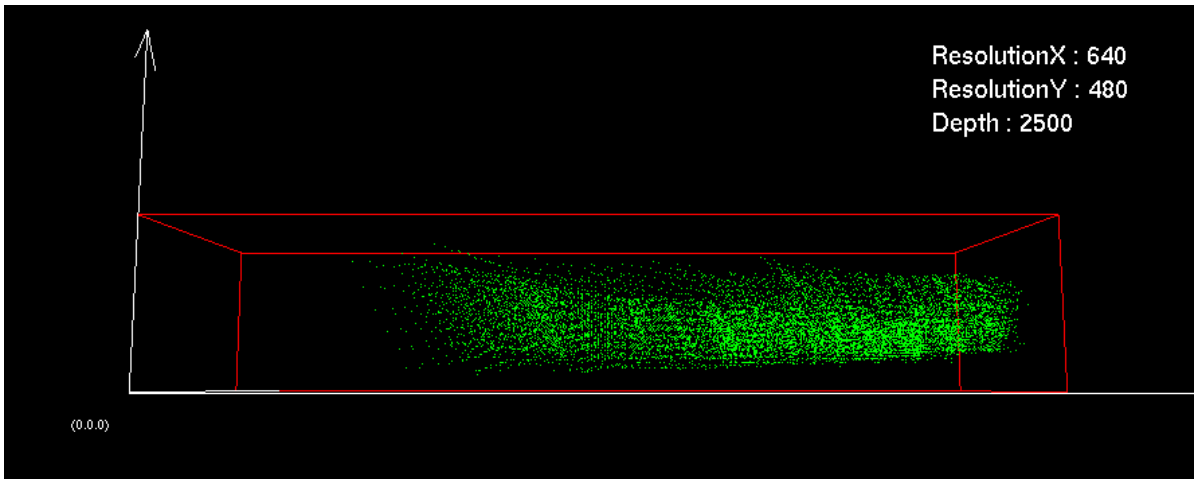


Figure 6: *3-D lookup table visualization.*

table can also be used for storing millimeter values instead of raw values, in the case when camera driver already returns converted values (in our experiments we stored millimeter values).

For the visualization of the table, we render a standard cube (see Fig. 6) and scale it according to the size of the lookup dimensions. The calibrated dataset is visualized inside of the table with a green color. The tool allows to rotate the cube and have a detailed view from all sides.

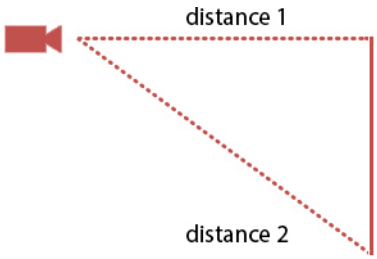


Figure 7: *Varying distance along the surface.*

5. Results and Discussion

The entire procedure of our multiple depth camera calibration can be characterized as a multistage process, where each individual step has a significant influence on the subsequent ones. As already described, the depth correction is done according to our new developed concept that incorporates tracking data for generating 3-D lookup table. This offers superior depth camera calibration w.r.t previous approaches as the following argumentation explains:

As stated before, the existing depth calibration approaches [BKKF13, MBPF12] use laser rangefinders to measure distances. However, the distance is not measured for every pixel of the depth image but instead only for one point, usually the image center. Then, the distance for the rest of the pixels is just assumed to be the same, which introduces significant errors. See Figure 7 for clarification, where according to the described experimental setup, *distance*1 and *distance*2 are assumed to be equal.

To have a more precise depth mapping all possible distances between the camera and target have to be measured, which is accomplished for the first time in our approach.

We carried out experiments to demonstrate the estimated errors for the Kinect sensor. To do this, we positioned the checkerboard target in such a way, so that it is shifted from the camera center. As shown in the Figure 8, the target mostly occupies the left-bottom part of the depth image. This allows us to estimate the change of the depth error depending on the distance from the center.

For the corresponding depth errors see Table 1. The table has a size of $w \times h$ (number of inner corners of checkerboard) where every value in a cell represents the error for the corresponding checkerboard corner expressed in millimeters. At first glance, the relatively high error values become apparent, which increase with a growing distance to the image center. Notice, that the maximum error values are significantly higher then those observed in other papers.

To demonstrate the results of the proposed depth calibration approach, we visualize the dataset extracted from the surface of the checkerboard. In Fig. 9 the green dataset represents camera recorded depth values of checkerboard corners, whereas the yellow one corresponds to corrected depth values.

One advantage of our approach is that it doesn't require hard mechanical setup and equipment for measuring dis-

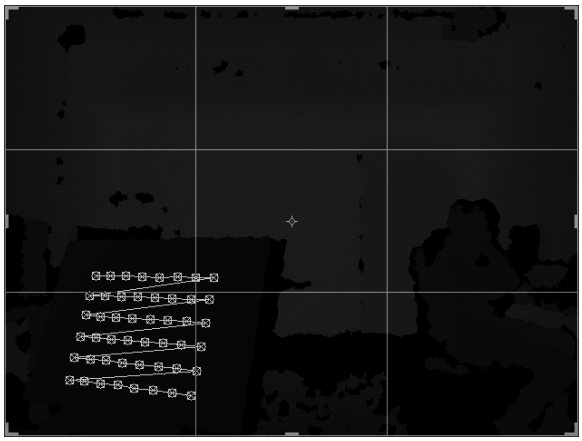


Figure 8: *Detected corners of the checkerboard positioned in about 2 meters from camera (target in the depth image mostly occupies the left-bottom part of the image, thus allowing to consider center shifted distances).*

74	65	56	48	21	13	-3	-9
86	76	76	47	38	29	20	12
100	99	87	76	56	46	37	27
126	114	102	89	77	66	63	53
153	131	118	113	109	96	84	81
183	151	145	132	126	113	115	110

Table 1: *Estimated depth errors measured for a single position of the checkerboard from Fig. 8 (all values are in **millimeters**).*

tances. The usage of a 6DOF marker-based tracking system allows to keep cameras fixed when caring out the camera calibration. In addition to this, our method supports the calibration of more then two cameras simultaneously, which reduces the required time.

In this work we also present a checkerboard-based 2-D calibration for a multi depth-camera system. For the two RGB sensors of the Kinect device we achieved a minimum re-projection error of about 0.3 pixel. This result is better then the original calibration provided by the manufacturer, which is greater than 0.5 pixel.

For an example, the tables presented below show estimated calibration parameters and distortion coefficients for two Kinect cameras $K1$ and $K2$.

Besides the intrinsic parameters, we also estimated the distortion coefficients in this step. In the tables presented below p_1, p_2 are the tangential and k_1, k_2, k_3 are the radial distortion coefficients respectively.

Extrinsic calibration results are presented in the Tables 6 and 7. The translation components are expressed in centimeters and rotation components are in Rodrigues representation.

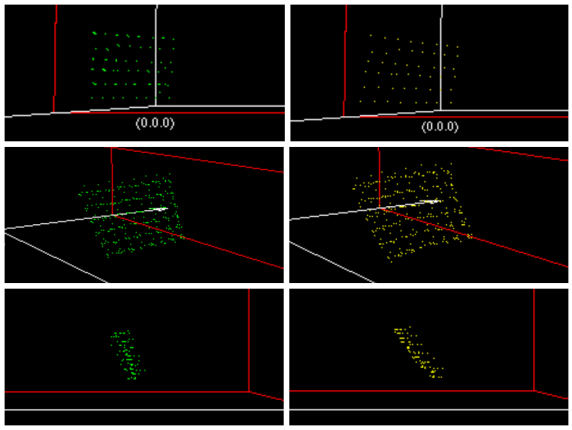


Figure 9: *Visualization of checkerboard corners. Left column: Corners estimated using camera depth values. Right column: Visualization of the same corners using corrected depth values.*

	f_x	f_y	u_0	v_0	*error*
K1	518.00	516.55	298.24	252.06	0.35
K2	523.08	522.35	312.70	248.28	0.31

Table 2: *Intrinsic calibration results of the RGB sensor.*

	f_x	f_y	u_0	v_0	*error*
K1	586.65	586.78	297.56	246.27	0.93
K2	581.53	580.81	292.65	246.36	0.94

Table 3: *Intrinsic calibration results of the IR sensor.*

	k_1	k_2	k_3	p_1	p_2
K1	0.191	-0.455	0.276	-0.001	-0.013
K2	0.247	-0.816	0.927	-0.009	-0.006

Table 4: *Distortion coefficients of the RGB sensor.*

	k_1	k_2	k_3	p_1	p_2
K1	-0.139	0.819	-2.964	-0.005	-0.024
K2	-0.236	1.654	-4.252	-0.007	-0.007

Table 5: *Distortion coefficients of the IR sensor.*

	T_x	T_y	T_z	R_x	R_y	R_z
K1	-18.0	15.5	258.5	-1.8	-2.2	0.4
K2	-58.3	7.2	132.4	1.6	1.4	5.7

Table 6: *Extrinsic calibration results for two Kinect sensors.*

	T_x	T_y	T_z	R_x	R_y	R_z
K1	2.47	-0.06	-0.68	-0.03	-0.03	0.00
K2	2.49	0.10	-0.34	-0.04	-0.04	0.00

Table 7: *Extrinsic calibration results between color and depth cameras of a Kinect sensor.*

6. Conclusion

In this paper we presented a novel depth correction approach incorporation an optical tracking system during the calibration process. Our method uses a 3-D lookup table to give a reliable per-pixel and per-distance mapping at every point in the depth image. In comparison to other methods, our approach support depth calibration of a camera in place using neither mechanical setups nor different types of distance measuring equipment that require the camera to be moved. Moreover, it facilitates the simultaneous depth correction of multiple cameras. To the best of our knowledge, there is no available depth correction algorithm capable to carry out the calibration of more than one camera at once. In addition to the proposed depth calibration approach, we also describe a standard checkerboard-based 2-D calibration. For acquiring the images we use the NITool. Being developed for this work the tool eases the data acquisition by enabling to switch from one camera to another and requesting necessary amount of frames per sensor. Additional support of parallel capturing improves the performance of the extrinsic calibration significantly.

Acknowledgements

This work was supported by the EU FP7 Marie Curie ITN "DIVA" under REA Grant Agreement No. 290227. We wish to thank the anonymous reviewers for their remarks. Special thanks to Christian Rosenke and Martin Luboschik for their valuable comments and suggestions.

References

[AMM12] AUVINET E., MEUNIER J., MULTON F.: Multiple depth cameras calibration and body volume reconstruction for gait analysis. In *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on* (2012), pp. 478–483. 2

[AZD13] ALEXIADIS D., ZARPALAS D., DARAS P.: Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras. *Multimedia, IEEE Transactions on 15*, 2 (2013), 339–358. 2

[BKKF13] BECK S., KUNERT A., KULIK A., FROEHLICH B.: Immersive group-to-group telepresence. *Visualization and Computer Graphics, IEEE Transactions on 19*, 4 (2013), 616–625. 1, 2, 6

[HCKH12] HERRERA C. D., KANNALA J., HEIKKILÃĎ J.: Joint depth and color camera calibration with distortion correction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 34*, 10 (2012), 2058–2064. 2

[Kre13] KREYLOS O.: Kinect camera calibration, 2013. http://www.doc-ok.org/?p=289. 2

[LFZL12] LIU W., FAN Y., ZHONG Z., LEI T.: A new method for calibrating depth and color camera pair based on kinect. In *Audio, Language and Image Processing (ICALIP), 2012 International Conference on* (2012), pp. 212–217. 2

[MBPF12] MAIMONE A., BIDWELL J., PENG K., FUCHS H.: Enhanced personal autostereoscopic telepresence system using commodity depth cameras. *Computers and Graphics, Volume 36 Issue 7, November, 2012, Pages 791-807 36*, 7 (2012), 791–807. 1, 2, 6

[MCAPL13] MACKNOJIA R., CHAVEZ-ARAGON A., PAYEUR P., LAGANIERE R.: Calibration of a network of kinect sensors for robotic inspection over a large workspace. In *Robot Vision (WORV), 2013 IEEE Workshop on* (2013), pp. 184–190. 1, 2, 5

[SSBH11] STÜRMER M., SEILER C., BECKER G., HORNEGGER J.: Alignment of multiple Time-of-Flight 3D Cameras for Reconstruction of walking feet. In *ISB2011 Brussels, Conference book Program & Abstracts* (2011). 1, 2

[Wen12] WENDELIN K.-A.: *Combining Multiple Depth Cameras for Reconstruction*. Master's thesis, Institut für Softwaretechnik und Interaktive Systeme, Technische UniversitaÌĹt Wien, 2012. 2

[Zha00] ZHANG Z.: A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22*, 11 (Nov 2000), 1330–1334. 2

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

S(wi)SS: A flexible and robust sub-surface scattering shader

A. Tsirikoglou[†1], S. Ekeberg[2], J. Vikström[2], J. Kronander[1] and J. Unger[1]

[1]Linköping University, Sweden [2]Swiss International AB, Sweden

Figure 1: *The proposed multi-layered SSS shader enables high quality tunable results. Three separate SSS layers are shown (left), SSS components can be intuitively edited both during rendering and compositing (middle), final composited image (right).*

Abstract

S(wi)SS is a new, flexible artist friendly multi-layered sub-surface scattering shader that simulates accurately sub-surface scattering for a large range of translucent materials. It is a physically motivated multi-layered approach where the sub-surface scattering effect is generated using one to three layers. It enables seamless mixing of the classical dipole, the better dipole and the quantized diffusion reflectance model in the sub-surface scattering layers, and additionally provides the scattering coming of front and back illumination, as well as all the BSDF components, in separate render channels enabling the artist to either use them physically accurately or tweak them independently during compositing to produce the desired result. To demonstrate the usefulness of our approach, we show a set of high quality rendering results from different user scenarios.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

Creating high quality renderings of translucent materials, such as skin, marble, milk, wax etc, is a key aspect in the production of visually interesting images. This, however, presents a significant challenge and requires flexible and highly accurate (physically motivated) subsurface scattering (SSS) modeling. Existing SSS shaders in most current production render engines such as V-Ray, Mental-Ray etc., do not support recently developed highly-accurate SSS models, and provide only a limited support for modeling and tuning materials with multiple subsurface layers (eg. skin consisting of the epidermal, dermal and sub-dermal layers etc.).

In this work, we present *S(wi)SS*, a new artist friendly multi-layered SSS shader that can be used to accurately simulate SSS for a large range of translucent materials. It has been implemented as a shader in V-Ray, and supports the classical dipole [JMLH01, DJ05], the better dipole (BD) [d'E12] and the quantized diffusion reflectance model (QD) [dI11] all in one shader, with artist friendly parameter tuning. Our multi-layered approach enables seamless mixing of the different models in the SSS layers and additionally generates their constituent parts (scattering coming of front and back illumination) in separate render channels enabling the artist to tweak the SSS appearance during both rendering and compositing.

† e-mail: apostolia.tsirikoglou@liu.se

1.1. Related work

Drawing from the classical diffusion theory [FPW92], and assuming semi-infinite homogeneous media, Jensen et al. [JMLH01] presented the dipole approximation model, a fast approximation for subsurface light transport which combines an accurate single scattering computation with a dipole point source diffusion approximation for multiple scattering. This approximation speeds up the computation of multiple scattering compared to Monte Carlo methods. The dipole model can also be applied to curved surfaces and converts the irradiance at a surface point into two point sources, one above and one below the surface. This approach significantly reduces the complexity of the problem and supports approximate angular incident and exitance variation by factoring the full BSSRDF into a product of Fresnel terms and a radially symmetric function. In this framework, Jensen and Buhler [JB02] presented a hierarchical summation technique to accelerate the computation of the outgoing radiance, since the integration of all incident points contributions is required.

Although the method presented by Jensen et al. [JMLH01] generally produces excellent results for optically-thick and highly scattering materials, the underlying assumptions cause inaccuracies for many geometrically complex objects and exhibit artifacts for semi-transparent or optically-thin materials especially in regions of high-curvature. Donner and Jensen [DJ05] extended the dipole model to a multi-pole one to accurately render thin and multi-layered translucent materials, while a little later [DJ07] they overcome the previous limitations by tracing photons inside the medium to capture inter-scattering between surfaces, and use a quad-pole diffusion approximation to improve accuracy. More specifically, Donner and Jensen [DJ07], placed exponentially attenuated point lights along the refracted light path instead of on the vertical line. This produces elliptical profiles, which are usually observed in reality under oblique lighting directions, but it also requires a numerical integral over the incident light path, which is expensive and prone to sampling noise.

Data-driven techniques have also been used to model BSSRDFs. Such a technique was proposed by Donner et al. [DLR*09] where an empirical BSSRDF is produced for semi-infinite homogeneous materials by fitting Monte Carlo simulated data. These methods use the classical diffusion theory approximation, which suffers from significant errors in near-source and high-absorption cases.

D'Eon et al. [dLE07] proposed an efficient multi-layered model by approximating diffusion profiles using sums of Gaussians. Later, Jimenez et al. [JWSG10] based on this work presented an algorithm for real-time realistic skin-rendering that simulates SSS and extends the screen-space based reflectance approach [JMLH01] by adding transmittance. For this, they used a physically based function that relates the light attenuation with the distance traveled inside the medium. More recently, d'Eon and Irving [dI11] proposed a quantized-diffusion based BSSRDF model. Their method is based on the neutron transport theory and led to several improvements of the diffusion theory. They place an infinite number of point light sources along the vertical incident light path. This requires to compute an integral over the path. The integral is evaluated analytically by approximating the diffusion function with quantized Gaussian functions. They promoted the use of an extended source term instead of approximating it as an impulse at a single depth like the dipole model. Since no closed-form solution exists to the extended source integral, and available numerical approaches were expensive, they approximated the resulting diffusion profile as a sum of Gaussians. To avoid fitting the Gaussians [dLE07], d'Eon and Irving further exploited the fact that time-resolved or quantized diffusion results in a Gaussian distribution and used this as a mathematical basis for finding the Gaussian weights.

Lately, Frisvad, Hachisuka and Kjeldsen [FHK13] presented a new fully analytical model for subsurface scattering that requires no precomputation and uses an approximate solution for a light ray in infinite media. This solution takes into consideration the incident light direction by using two directional sources instead of two point sources, like in classical dipole [JMLH01], relaxing that way the assumption of a flat boundary.

2. Background

The layered 2D searchlight problem (Figure 2) is about finding the reflected $R(r)$ and transmitted $T(r)$ energy in a layered scattering medium, as a function of distance r from the point of illumination by a focused beam. The diffuse reflectance profile $R_d(r)$, is used to calculate each shading point color. The implementation of the diffuse reflectance profile is the one that defines the model behavior.

Rendering surfaces requires to compute the light leaving various points on the boundary. For this, Jensen et al. [JMLH01] used Fick's law which states that (for isotropic sources) the vector flux $\vec{E}$ is the gradient of the fluence ϕ (flux integrated over time):

$$\vec{E}(r) = -D\vec{\nabla}\phi(r). \tag{1}$$

In more general terms, this applies Neumann boundary conditions where only derivatives are specified [CC05, HCJ13]. Thus, the diffuse reflectance profile $R_d(r)$ is given by:

$$R_d(r) = C_\phi\phi(r) + C_{\vec{E}}(\vec{E}(r)\cdot\vec{n}), \tag{2}$$

where r is the distance from the illumination point, C_ϕ and $C_{\vec{E}}$ are factors depending on the angular moments and $\vec{n}$ is the normal of the surface where the light arrives.

In classical diffusion theory and due to Fick's law the diffuse reflectance profile $R_d(r)$ depends only on the surface flux (the gradient of the fluence along the surface normal)

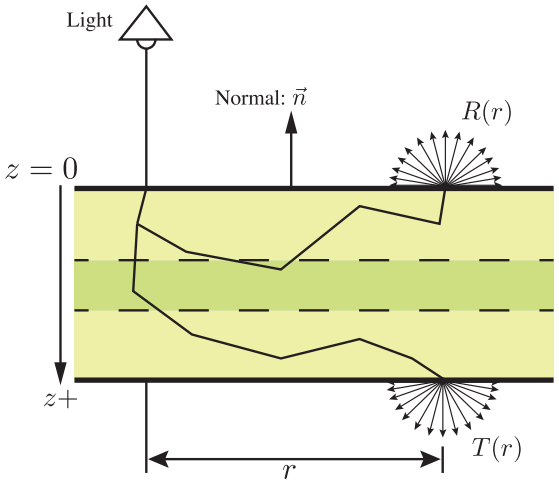


Figure 2: *The layered 2D searchlight problem.*

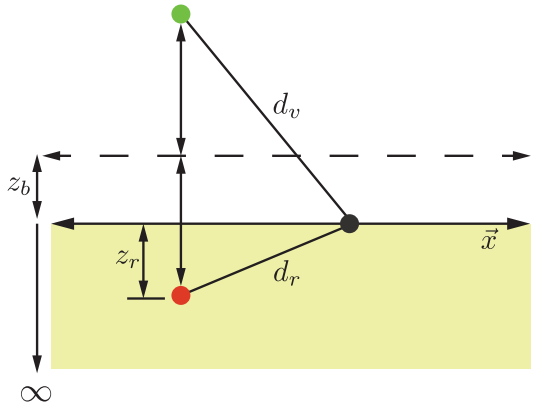


Figure 3: *The dipole model.*

and not on the fluence $\phi(r)$ itself. Thus, with factors $C_\phi = 0$ and $C_{\vec{E}} = 1$ in eq. (2), so only vector flux and not fluence contributes to the diffuse reflectance, it can be shown that the radiant exitance on the boundary is the dot product of the vector flux with the surface normal:

$$R_d(r) = \vec{E}(r) \cdot \vec{n} = -D(\vec{\nabla} \cdot \vec{n})\phi(r). \tag{3}$$

Hence, to compute the diffusion profile $R_d(r)$ due to the classical dipole model (Figure 3, where z_b denotes the linear extrapolated distance), the directional derivative $\vec{\nabla} \cdot \vec{n}$ of the fluence in the direction of the normal is needed to be evaluated. This gives:

$$R_d(r) = \frac{\alpha'}{4\pi}\left[\frac{z_r(1+\sigma_{tr}d_r)e^{-\sigma_{tr}d_r}}{d_r^3} + \frac{z_v(1+\sigma_{tr}d_v)e^{-\sigma_{tr}d_v}}{d_v^3}\right] \tag{4}$$

where α' is the reduced scattering albedo, z_r is the distance of the real light source (placed below the surface) from the surface and z_r the distance of the mirrored virtual light source (placed above the surface) from the surface, d_r is the distance from the ray exit point to the real light source and d_v its distance from the virtual light source, and σ_{tr} is the transport coefficient.

The relative contributions of C_ϕ and $C_{\vec{E}}$ factors change the exitance calculations in eq. (2) for other boundary conditions.

D'Eon [d'E12] introduces the better dipole assuming that the multiple scattering term can be given by a diffusion method of images solution to the 2D searchlight problem. He furthermore simplifies the searchlight problem assuming that the light arrives normal to the surface and computing only the net flux leaving the surface at a distance r from the illumination point. The diffuse reflectance profile $R_d(r)$ for the better dipole model is then given by

$$R_d(r) = \frac{(\alpha')^2}{4\pi}\left[\left(C_{\vec{E}}\frac{z_r(1+\sigma_{tr}d_r)}{d_r^2} + \frac{C_\phi}{D}\right)\frac{e^{-\sigma_{tr}d_r}}{d_r} - \left(C_{\vec{E}}\frac{z_v(1+\sigma_{tr}d_v}{d_v^2} + \frac{C_\phi}{D}\right)\frac{e^{-\sigma_{tr}d_v}}{d_v}\right] \tag{5}$$

where D is the diffusion coefficient and the fluence and flux factors are given now by

$$C_\phi = \frac{1}{4}(1-2C_1) \qquad \text{and} \qquad C_{\vec{E}} = \frac{1}{2}(1-3C_2)$$

respectively instead of classical dipole's $C_\phi = 0$ and $C_{\vec{E}} = 1$, and C_1 and C_2 are angular moments which approximations are given by D'Eon and Irving [dI11].

Finally, D'Eon and Irving [dI11] proposed the quantized diffusion model, where the diffuse reflectance profile is approximated by a sum of Gaussians:

$$R_d(r) \approx \alpha' \sum_{i=0}^{k-1} w_R(i) w_i G_{2D}(\nu_i, r) \tag{6}$$

where $w_R(i)$, w_i are weights and ν_i are the variances of normalized 2D Gaussians G_{2D}. The equations necessary to calculate the weights and variances of the Gaussians are themselves summations of integrals depending on further weights $w_{\phi R}(\nu, i)$ and $w_{\vec{E}R}(\nu, i)$.

3. Multi-layer sub-surface scattering

S(wi)SS creates a layered BSDF based on three components: (1) a bottom SSS component, (2) a middle diffuse component and (3) a top specular component.

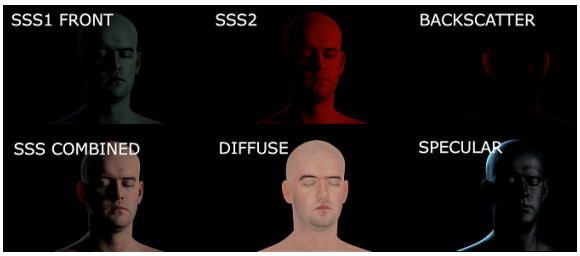


Figure 4: *BSDF's SSS, diffuse and specular components of our multi-layer approach 3.*

In Figure 4 the BSDF components are shown. The bottom SSS component (SSS COMBINED) is produced by the combination of three SSS layers (SSS1 FRONT, SSS2 and BACKSCATTER). The SSS component is then combined with the diffuse component and finally with the specular one.

(1) The bottom SSS component consists of a basic SSS layer providing front, (optionally) back, and single scattering. A second SSS layer provides front scattering, and a final SSS layer for back scattering.
Each of the three SSS layers has its own scatter color and radius and can be weighted separately. In each SSS layer, the user is able to apply the classical dipole, eq. (4), the BD, eq. (5), or the QD, eq. (6), model. Thus, the rendering time can be adjusted by applying, for example, the most-detailed and time consuming QD to the top SSS layer and the faster but less-detailed classical dipole or BD to the remaining SSS layers.
The second and third SSS layers do not significantly contribute to single scattering and thus only support multiple scattering to reduce rendering time.

(2) The middle diffuse component is a classic Lambertian reflection model. Similar to the SSS component, the diffuse component contribution can also be manually adjusted through the user interface. The SSS and diffuse components are combined before layered in the BSDF.

(3) The surface specular component consists of two Phong models for specular reflection. Two specular layers are used to simulate looks such as the upper, oily layer of skin atop the regular specular layer. Each of them provides separate specular color, weight, glossiness, subdivisions and index of refraction settings. Like the rest of the BSDF components, the specular one's contribution can also be weighted through the shader interface.

3.1. Front and back illumination map

We use an illumination map to calculate the color of each shading point. Front and back scattering are separated by classifying the illumination map samples in two categories: the ones on the front side of the object and the ones on the back side with respect to the camera view. During rendering,

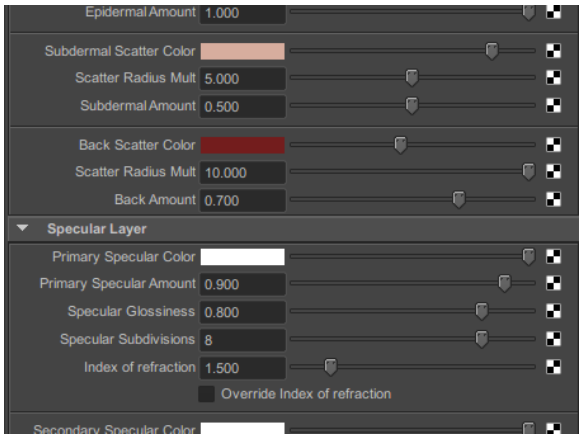


Figure 5: *SSS and specular settings in a representative shader view 4.*

the front and back illumination is weighted according to the applied model (BD or QD).

This way, the user can totally control the SSS component, and its separate front and back SSS layers and therefore to handle them physically accurately or tweak them independently according to the specific needs.

4. Implementation

We implemented our algorithm in V-Ray 2.0 by Chaos Group for Autodesk's Maya but a similar implementation would be possible to use with other render engines as well.

We based the generation of the illumination map for *S(wi)SS* on the existing mechanism in *VRayFastSSS2*, the SSS shader packaged with V-Ray. The front and back scattering layers are generated in separate render channels enabling the artist to tweak them separately, or rely on physical accuracy to determine the final look.

During rendering, the diffuse reflectance profile, Rd, is used to calculate each shading point's color. For the BD model, we improved VRayFastSS2's existing diffuse reflectance profile by implementing the method described by d'Eon [d'E12].

QD is based on the improved diffusion theory with Grojean's approximations and uses a sum of weighted 2D Gaussians to approximate the reflectance profile [dI11]. An extended multipole model for a finite material of finite thickness was set up instead of a dipole. The quantization parameters regarding time, variances and weights are initialized, calculated and saved for later use during rendering when the diffuse reflectance profile is evaluated by the sum of Gaussians. These pre-calculated parameters significantly reduce rendering time.

S(wi)SS provides several settings to modify properties like

the material index of refraction and light map options (eg. the illumination map can be built by an irradiance map or geometry sampling). Moreover, the user can intuitively control all the BSDF components and their layers 3 through the shader interface by setting the desired amounts, colors and the specific options given for each component. In Figure 5 it is given a representative view of the user interface and how the layers of the SSS and specular components can be set. The render elements that can be moreover tuned during compositing phase consist of all three BSDF components (SSS, diffuse and specular), as well as their layers including the scattering coming of front and back illumination. This way, the artist is enabled to tweak (enhance or weaken) any component according to the specific look demands. For example, in skin rendering it is usually desired an enhanced back scattering layer for strong SSS effects and thus highly realistic results.

5. Results

This section presents a set of results rendered using the *S(wi)SS* shader. The rendered images compare and highlight the effect of the artist friendly shader parameters on the final results. We also show figures describing the performance of our implementation. The renderings are mainly focused on the models comparison and different algorithm combinations.

In Figure 6 the final result using the BSDF components shown in Figure 4 is given. The model is illuminated from both the front and the rear. The SSS component is created using all three SSS layers, where the QD model, eq. (6), is applied to the basic SSS layer, and the BD model, eq. (5) is applied to the rest SSS layers. For better SSS algorithm evaluation, we show in Figure 12 the SSS component as it is given by the classical dipole, the BD and QD models. The level of the rendered details is increased from left (classical dipole) to right (BD and rightmost QD). Please note areas like the lips, eyes and ears, where the differences and the SSS effect are clear.

In Figure 8 a 3D model, similar to the one that d'Eon and Irving [dI11] used to present their QD method, has been rendered just with the first basic SSS layer of the SSS component using the classical dipole (*VRayFastSSS2*), eq. (4), BD, eq. (5), and QD, eq. (6), algorithms (from left to right) for easy side-by-side comparisons. In Figure 9 detailed views of the model comparative results in Figure 8 are given. Their rendering times are given in Table 1. All the images were rendered with an Intel Xeon W3680, 12 MB Cache, 3.33 Ghz, Six-Core server processor machine. BD is almost as fast as *VRayFastSSS2*, that implements the classical dipole method, whereas the QD model is slower. This is expected as the implementation of QD requires the sum of 36 approximately weighted Gaussians for every shading point. Moreover, the above given times are for hight quality renderings but it can be reduced by decreasing the resolution of surface

Figure 6: *Final result using the BSDF components shown in Figure 4.*

lighting, computed during the prepass phase. Regarding the visual result, the dominance of QD is obvious. The level of the rendered details is significantly higher than the classical and BD providing at the same time realistic subsurface scattering effects. Moreover, QD gives the smoothest shading result in the rendered surface.

Method	Time per frame
Classical Dipole	∼ 43sec
Better Dipole	∼1min 2sec
Quantized Diffusion	∼6min 2sec

Table 1: *Average rendering times for renderings in Figure 9.*

In Figure 10 the front and back separation can be seen. In the particular rendering, only the first basic SSS layer is used, where the QD model, eq. (6), is applied (rightmost rendering in Figure 8). The SSS layer is then given by the combination of the produced front and back scattering.

In Figure 11 multi-layer SSS rendering results are shown. To the left, the QD model, eq. (6), was used just for the basic SSS layer. In the middle the QD model was used for the basic SSS layer and the BD, eq. (5) for the second SSS layer, whereas to the right the QD model was used for the basic SSS layer, the BD for the second SSS layer and the classical dipole, eq. (4), for the final back SSS layer. It is worth noticing that as we add more layers to the SSS component the result changes depending on the SSS layer's scatter color and radius. Specifically in Figure 11 the added SSS layers scatter mainly the red color, explaining the respective rendered results.

Finally, in Figure 7 renderings for different translucent

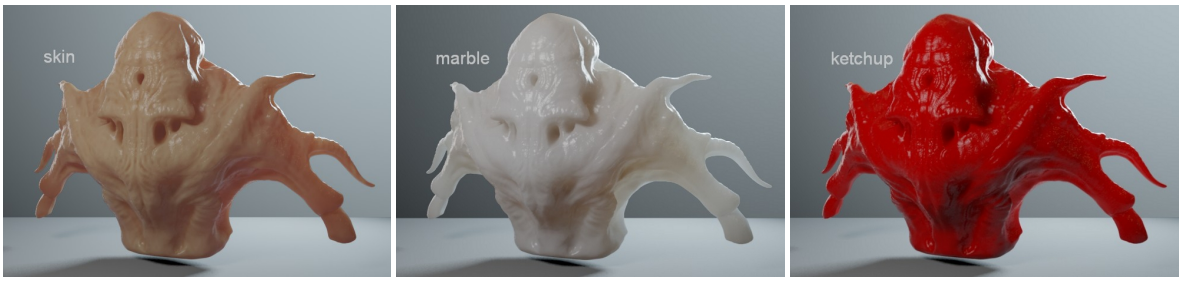


Figure 7: *Skin (left), marble (middle) and ketchup (right) rendered with S(wi)SS shader.*

materials are presented. For skin rendering (left) we used all three layers of the SSS component with the QD model, eq. (6), for the basic SSS layer and the BD one, eq. (5), for the rest two, whereas for the marble (middle) and ketchup (right) renderings only the basic SSS layer was used where the QD model, eq. (6) was applied.

6. Discussion

S(wi)SS is a powerful and flexible tool that compared to the existing commercial shaders uses recently introduced highly-accurate SSS calculations. It is related with the corresponding implementations coming packaged with V-Ray and Mental ray. However, our multilayer approach provides, additionally to the standard options, advanced approximations for detailed SSS for each layer of the SSS component and generates all the BSDF components including scattering coming of front and back illumination in separate render channels all in one rendering.

7. Conclusion

S(wi)SS creates highly detailed SSS using one to three layers. The artist can use the separate render elements either physically accurately or tweak them independently during the compositing phase. Figure 1 represents a typical *S(wi)SS* workflow where the artist adjusts the SSS render elements in compositing software, synthesizing the desired look.

Responding to the demands in visual effects production, *S(wi)SS* is a flexible, accurate, easy-to-use look-development tool that introduces a multi-layered SSS approximation to simulate different material types and provides enhanced user control. The S(wi)SS shader and its surrounding process is currently being used effectively in a visual effects studio.

Acknowledgments

We would like to thank Vladimir Koylazov from Chaos Group for helping us with V-Ray, as well as Eugene d'Eon for his key clarifications on the quantized diffusion model. This project was funded by the Swedish Foundation for Strategic Research (SSF) through grant IIS11-0081, Linköping University Center for Industrial Information Technology (CENIIT), and the Swedish Research Council through the Linnaeus Environment CADICS.

References

[CC05] CHENG A. H.-D., CHENG D. T.: Heritage and early history of the boundary element method. *Engineering Analysis with Boundary Elements 29*, 3 (2005), 268–302. 2

[d'E12] D'EON E.: A better dipole. `http://www.eugenedeon.com/papers/betterdipole.pdf` (2012). 1, 3, 4

[dI11] D'EON E., IRVING G.: A quantized-diffusion model for rendering translucent materials. *ACM TOG 30*, 4 (July 2011), 56:1–56:14. 1, 2, 3, 4, 5

[DJ05] DONNER C., JENSEN H. W.: Light diffusion in multi-layered translucent materials. *ACM TOG 24*, 3 (July 2005), 1032–1039. 1, 2

[DJ07] DONNER C., JENSEN H. W.: Rendering translucent materials using photon diffusion. In *Rendering Techniques* (2007), pp. 243–251. 2

[dLE07] D'EON E., LUEBKE D., ENDERTON E.: Efficient rendering of human skin. In *Proc. of Eurographics Symposium on Rendering* (2007), pp. 147–157. 2

[DLR*09] DONNER C., LAWRENCE J., RAMAMOORTHI R., HACHISUKA T., JENSEN H. W., NAYAR S.: An empirical bssrdf model. *ACM TOG 28*, 3 (July 2009), 30:1–30:10. 2

[FHK13] FRISVAD J. R., HACHISUKA T., KJELDSEN T. K.: Directional dipole for subsurface scattering in translucent materials. *ACM TOG* (Aug. 2013). Manuscript. 2

[FPW92] FARRELL T. J., PATTERSON M. S., WILSON B.: A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties in vivo. *Medical Physics 19*, 4 (1992), 879–888. 2

[HCJ13] HABEL R., CHRISTENSEN P. H., JAROSZ W.: Classic and modified diffusion theory for subsurface scattering. *Tech. rep.* (2013). Disney Research Zürich. 2

[JB02] JENSEN H. W., BUHLER J.: A rapid hierarchical rendering technique for translucent materials. In *Proc. SIGGRAPH 2002* (2002), pp. 576–581. 2

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *Proc. SIGGRAPH 2001* (2001), pp. 511–518. 1, 2

[JWSG10] JIMENEZ J., WHELAN D., SUNDSTEDT V., GUTIERREZ D.: Real time realistic skin translucency. *IEEE CG&A 30*, 4 (2010), 32–41. 2

Figure 8: *S(wi)SS shader renderings using just the first basic SSS layer of the SSS component implementing the classical dipole (left), BD (middle) and QD(right).*

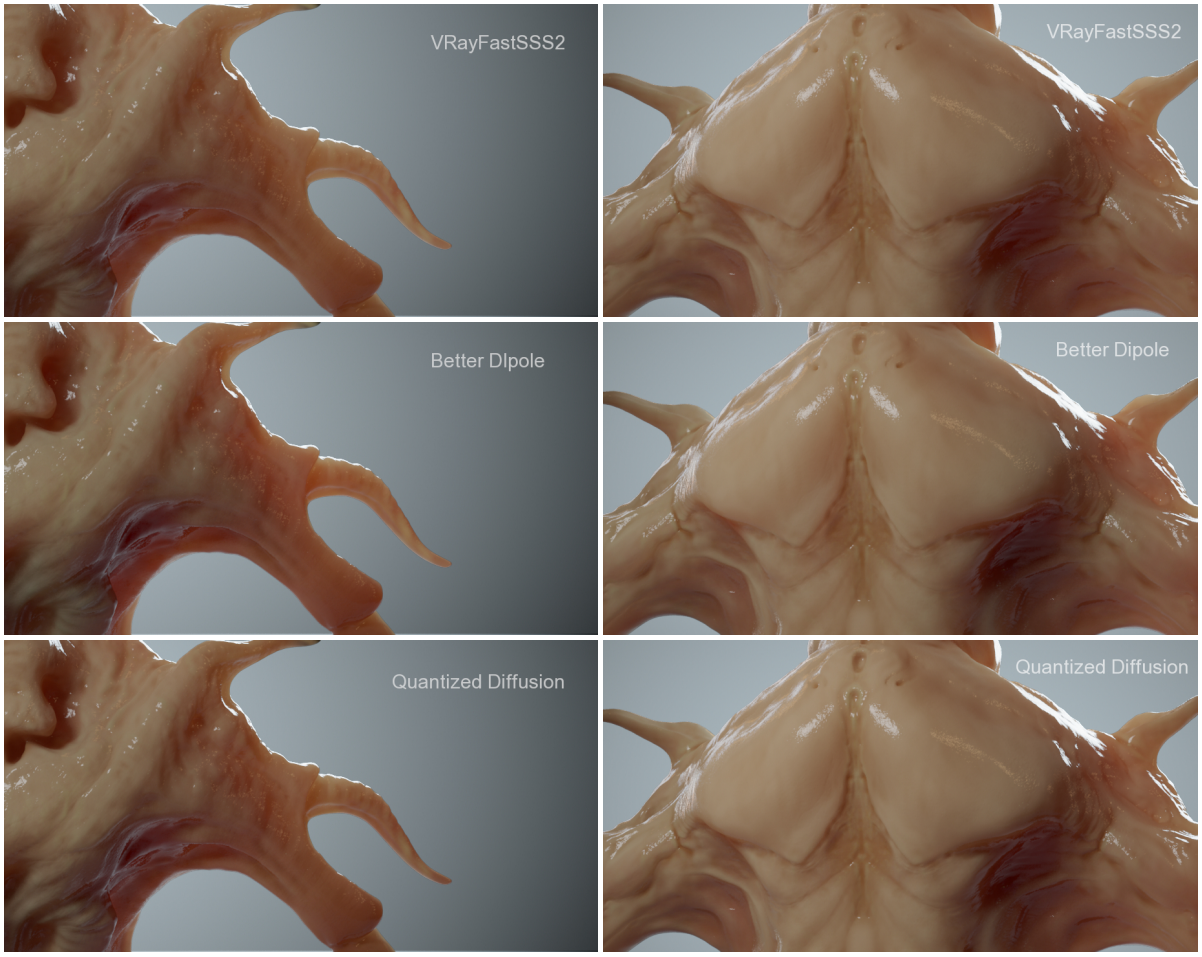


Figure 9: *Detailed views of renderings in Figure 8.*

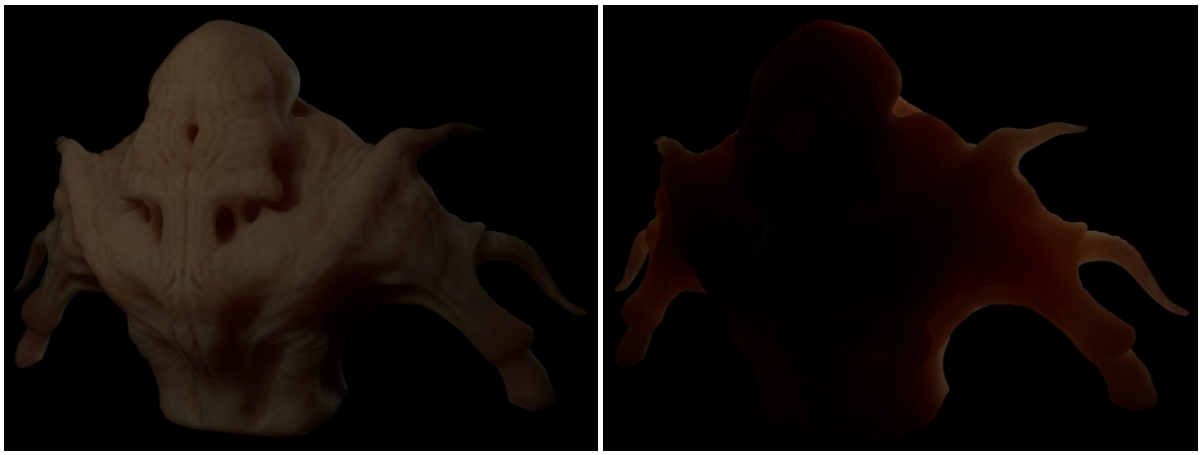


Figure 10: *Front and Back scattering separation in QD model.*

Figure 11: *S(wi)SS renderings with one (left), two (middle) and three (left) SSS layers, using the QD model as the basic SSS layer, the BD as the middle SSS layer and the classical dipole as the last back SSS layer respectively.*

Figure 12: *SSS component rendered with the classical dipole (left), the BD (middle) and QD (right) model.*

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

Accelerated Computation of Minimum Enclosing Balls by GPU Parallelization and Distance Filtering

Linus Källberg & Thomas Larsson

Mälardalen University, Sweden

Abstract

Minimum enclosing balls are used extensively to speed up multidimensional data processing in, e.g., machine learning, spatial databases, and computer graphics. We present a case study of several acceleration techniques that are applicable in enclosing ball algorithms based on repeated farthest-point queries. Parallel GPU solutions using CUDA are developed for both low- and high-dimensional cases. Furthermore, two different distance filtering heuristics are proposed aiming at reducing the cost of the farthest-point queries as much as possible by exploiting lower and upper distance bounds. Empirical tests show encouraging results. Compared to a sequential CPU version of the algorithm, the GPU parallelization runs up to 11 times faster. When applying the distance filtering techniques, further speedups are observed.

Categories and Subject Descriptors (according to ACM CCS): D.1.3 [Programming Techniques]: Concurrent Programming—Parallel programming, I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

The minimum enclosing ball has found widespread use in many different areas of computer science. Depending on the specific context, the dimensionality of the balls varies broadly. Low-dimensional cases are common in computer graphics, computer-aided design, and geographic information systems. Higher-dimensional cases arise frequently in multidimensional index methods and classification based on support vector machines. In these cases, data sets with dimensions in the range 10 to 10000 are common [TKK07].

In the plane, the minimum enclosing circle problem is about fitting a circle to embrace a set of points as tightly as possible. This is a mathematical optimization problem, where the task is to find the unique center such that the maximum distance from the center to any input point is minimized. More generally, in any dimension d, the minimum enclosing ball (MEB) is sought, and the solution is usually denoted $B^* = (c^*, r^*)$, where c^* is the unique center that gives the smallest radius r^*. The MEB problem bears many names, e.g., the smallest bounding sphere or 1-center problem, and the literature is vast [PS85].

Due to the high computational complexity of finding the exact MEB, a number of approximation algorithms have been proposed in the literature [BC03, KMY03, Yıl08]. In a recent publication, Larsson and Källberg [LK13] proposed an efficient algorithm to compute a $(1+\varepsilon)$-approximation of B^* in $O(\frac{dn}{\varepsilon} + \frac{d}{\varepsilon^3})$ time for n input points in dimension d. This algorithm, named FastApxBall, computes a sequence of approximations $B_i = (c_i, r_i)$ such that $r_i \leq r^*$, until a solution is reached whose radius can be enlarged to cover the entire point set without exceeding $(1+\varepsilon)r^*$. Pseudocode is shown in Figure 1. On Lines 1–3, the current approximation is initialized. On Line 4, a subset C of input points, known as a core-set, is initialized. Then in each iteration of the loop beginning on Line 5, the point $q \in P$ located farthest from the current center point is found and inserted into the core-set. The point q as well as $h = \|q - c\|$, where $\|\cdot\|$ denotes the Euclidean norm, are also used on Lines 8 and 9 to update the current solution in each iteration.

The purpose of the subroutine SolveApxBall, invoked on Line 11, is to further refine the current solution by considering only the (at most $2/\varepsilon$) points currently in the core-set (see [LK13] for more details). In practice, this reduces the

FastApxBall(P,ε)
input: $P = \{p_1, p_2, ..., p_n\}$, $\varepsilon > 0$
output: A $(1+\varepsilon)$-approximation B of B^*
1. $q', h' \leftarrow$ FindFarthestPoint(p_1, P)
2. $q, h \leftarrow$ FindFarthestPoint(q', P)
3. $(c, r) \leftarrow ((q'+q)/2, h/2)$
4. $C \leftarrow \{q', q\}$
5. **loop** $2/\varepsilon$ **times** (at most)
6. $q, h \leftarrow$ FindFarthestPoint(c, P)
7. **if** $h \leq r(1+\varepsilon)$ **then exit loop**
8. $r \leftarrow (\frac{r^2}{h} + h)/2$
9. $c \leftarrow q + \frac{r}{h}(c-q)$
10. $C \leftarrow C \cup \{q\}$
11. $(c, r) \leftarrow$ SolveApxBall($(c, r), C, \varepsilon/2$)
12. **return** (c, h)

Figure 1: *The algorithm* FastApxBall.

number of passes considerably, especially in low to moderate dimensions with $n \gg d$. As noted by Larsson and Källberg, it is possible to remove Line 11 from the code to get a modified version of the algorithm, called SimpleApxBall, with time complexity $O(\frac{dn}{\varepsilon})$. Interestingly, it has been shown [KL] that this algorithm is equivalent to Yıldırım's first algorithm [Yıl08].

Since FindFarthestPoint takes $O(dn)$ time in each of the $O(\frac{1}{\varepsilon})$ iterations, the total time spent in this subroutine is $O(\frac{dn}{\varepsilon})$, as captured by the left term in the time complexity of FastApxBall and the remaining term in the time complexity of SimpleApxBall. In practice, these repeated farthest-point queries constitute the main bottleneck in both algorithms. Fortunately, however, due to its high degree of data parallelism, the farthest-point query makes a good candidate for acceleration using parallel computing.

In this paper, GPU parallelization strategies are proposed as well as two algorithmic distance filtering approaches, which are applicable in both serial and parallel settings to reduce the cost of the dominating distance computations. The proposed techniques should be applicable in several other MEB algorithms that perform repeated farthest-point queries, such as [Gär99, BC03, KMY03, Yıl08].

2. Farthest-point queries on the GPU

We let the input points and the center point be represented on the GPU as the following $n \times d$ matrix and d-ary column vector:

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,d} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,d} \end{bmatrix}, \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_d \end{bmatrix},$$

where $p_{j,k}$ and c_k denote the k-th coordinate of p_j and c, respectively. To realize the FindFarthestPoint subroutine efficiently, a straightforward solution is to first compute all of the n squared distances $\|p_j - c\|^2$ in parallel into a vector e, and then find the maximum of e in a separate parallel reduction step. Thus, e is given by

$$e = \begin{bmatrix} (p_{1,1}-c_1)^2 & + & \cdots & + & (p_{1,d}-c_d)^2 \\ (p_{2,1}-c_1)^2 & + & \cdots & + & (p_{2,d}-c_d)^2 \\ \vdots & & \ddots & & \vdots \\ (p_{n,1}-c_1)^2 & + & \cdots & + & (p_{n,d}-c_d)^2 \end{bmatrix}.$$

Note that computing squared as opposed to exact distances is preferable since it avoids taking n square roots. The computation of e can be further simplified by first rewriting each element as

$$e_j = \|p_j - c\|^2 = \|p_j\|^2 + \|c\|^2 - 2\langle p_j, c\rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product. Since the term $\|c\|^2$ is the same in all distances during a pass, it can be disregarded without altering which point is returned as the farthest from the reduction step. Furthermore, the term $\|p_j\|^2$ remains constant throughout all passes of the MEB algorithm; thus, it can be precomputed for each input point into an n-ary vector u, which is then reused in every pass. Using these simplifications, the distance computation step amounts to a GEMV (general matrix-vector multiplication) kernel:

$$e = u - 2Pc.$$

Efficient GPU implementations of GEMV are commonly found in GPU implementations of BLAS (Basic Linear Algebra Subprograms), such as cuBLAS and MAGMA [DDG*]. Similarly, there are efficient GPU implementations of the reduction step available, e.g., from the Thrust template library for CUDA [BH12].

2.1. Tailor-made kernels

The above solution based on GEMV can be expected to work well in many situations. However, to support the distance filtering techniques that will be described in Section 3, we implemented two tailor-made kernels in CUDA that allow for masking out certain rows of the matrix P in the distance computations. To retain some degree of data locality in this case, we let P be stored in row-major order. This way, any point can be loaded from the global memory in a coalesced fashion into the SMs, provided the dimension is large enough and the matrix is allocated with a properly set pitch.

In the first kernel, a tile mesh is superimposed on the matrix such that each tile covers $t_x \times t_y$ elements. Each thread block is mapped to a number of such tiles, determined by the parameters w_x and w_y for the x and y directions, respectively. Thus, in general, a thread block processes

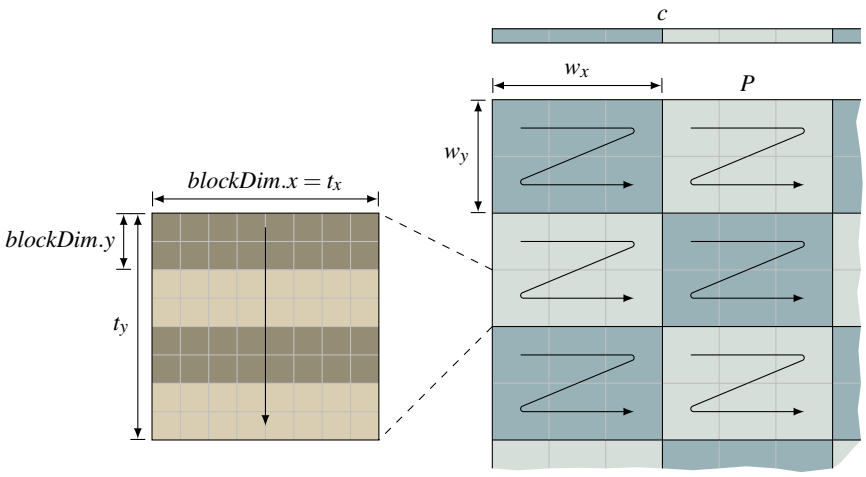


Figure 2: *Illustration of the assignment of matrix tiles to thread blocks and elements within the tiles to threads. In this example, each block processes $w_x = 3$ tiles in the horizontal direction and $w_y = 2$ tiles in the vertical direction. Since $t_x = t_y = 8$ and $blockDim.y = 2$, each thread must iterate four times for a full tile to be processed.*

$(w_x \times w_y) \times (t_x \times t_y)$ elements of the matrix and $w_x \times t_x$ elements of the center point in total in each invocation, although some blocks may do less work depending on the matrix dimensions. Furthermore, depending on the *blockDim* settings, the thread blocks might need to iterate within each tile to process all of its elements. To keep the parameter space manageable, we let the tiles be squares with side t_d, i.e., $t_x = t_y = t_d$, and we let $blockDim.x$ be fixed to t_d. This leaves the parameters t_d, $blockDim.y$, w_x, and w_y that can be varied to tune the performance of the kernel. More details on the used tuning process are given in Section 4.

As illustrated in Figure 2, the thread blocks process their assigned tiles in a left-to-right, top-to-bottom order. Within each tile, the elements are processed from top to bottom. With these traversal orders, each thread can maintain $t_y/blockDim.y$ partial sums of its computed squared differences by storing them in registers. After the last tile in the current row has been processed, all threads in the block store their partial sums into a matrix of size $t_y \times t_x$ in shared memory. Each row of this matrix is then summed up, which reduces it to a column vector of t_y partial sums. Then, before moving on to the next row of tiles, these values are added to the output vector e, which was initialized to 0 before invoking the kernel. Note that in cases when there are more than one thread block working on the same rows of P, i.e., when $w_x \times t_x < d$, the latter addition to e must be done using an atomic add operation, since another thread block might attempt to update the same elements of e simultaneously.

Otherwise, when there is only one block in the horizontal dimension of the grid of thread blocks, a regular add operation can be used instead. However, a more significant optimization is possible in this case: since each distance is computed by a single thread, the reduction needed to find the largest distance can be integrated with the distance computation kernel. This fuses two kernel calls (first computing the distances, then finding the maximum) into one. Moreover, it reduces the amount of data that needs to be written out to global memory, since it becomes unnecessary to store all the distances to the vector e. Thus, we made the kernel detect this special case, so that a local reduction is performed by each thread block to find the maximum of its computed distances. Then, before returning, an inter-block reduction is performed in a scalar in global memory using atomic operations.

Whenever the matrix dimensions are not multiples of the tile dimensions, the last row and/or column in the grid of thread blocks will have tiles to process that are not completely filled. This is handled as a special case in the kernel, so that the overhead from the necessary conditional statements is isolated to the affected thread blocks. Since there are not enough elements in these partially filled tiles to occupy all threads in these blocks, parts of the thread blocks become inactive. As long as both n and d are large enough to give mostly filled tiles overall, this reduction in thread utilization is amortized sufficiently over the processing of the filled tiles. However, whenever either n or d is very small, measures must be taken to avoid poor performance due to low utilization. Since we focus mainly on cases where $d \ll n$ here, we consider only cases where d is small, i.e., when P has a tall shape. Our second kernel is specifically optimized for such matrices. When executing this kernel, we relax the requirement on square tiles and fix only $t_x = d$. As before, we set $blockDim.x = t_x$, which in this case automatically implies $w_x = 1$. Thus, the remaining tunable launch parameters for this kernel are t_y, $blockDim.y$, and w_y. Note that there

may be more than one warp working on each row of the matrix in this case, when d is neither a multiple nor a divisor of the warp size 32. For example, in $d = 3$, each warp spans $\lfloor \frac{32}{3} \rfloor = 10$ full rows as well as parts of one or two other rows. This keeps all threads of the block fully occupied, and due to the row-major organization of P, it also increases the locality of the data accessed by each warp.

3. Distance filtering

Evaluating the distance metric for pairs of points is an expensive operation, particularly in high dimensions. Therefore, attempting to reduce the number of exact distance computations by utilizing approximate distance measures may speed up the processing substantially. By exploiting knowledge from already computed similar distances together with the triangle inequality, safe bounds can be derived to give simple conditions for when an exact distance computation can be skipped.

Of course, using the triangle inequality to filter out distance computations is not a new idea. Such approaches are well-known in data mining where similarity queries are common (see, e.g., [BK73, BS98, BEKS01]). Hjaltason and Samet describe several general rules which can be applied to prune the search space under varying circumstances depending on what distances are known and what distance calculations are attempted to be avoided [HS03].

The filtering techniques we describe below are designed for the specific case of MEB computation. Intuitively, this is a situation where this type of filtering can be expected to pay off very well. The sequence of generated center points sweeps out a path in the neighborhood of the optimal center c^*, and overall the distance between each pair of consecutive center points tends to decrease with each pass. Thus, searching for the farthest point from c_{i+1} is expected to be very similar to searching for the farthest point from c_i. To study the benefit of reducing the number of full distance computations between points during the farthest-point queries, we describe three different filtering schemes for caching and reusing computed distances between the passes of the MEB algorithm. To make the discussion more general, we will use the notation $D(\cdot,\cdot)$ to represent any distance measure that satisfies the triangle inequality.

3.1. Triangle inequality filtering

During the first farthest-point query of the MEB algorithm, the distances from the first center point to all input points are computed and stored in a simple auxiliary array of size $O(n)$, hereafter called the distance array. Then in all the subsequent passes, the distances cached in this array are used to derive an upper bound on the distance to each point in P from the current center point. Before a pass is started, say pass $i+m$, the distances from c_{i+m} to all previous center points are computed. Also, a lower bound f on the actual

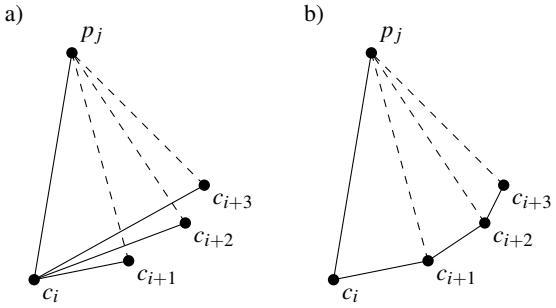


Figure 3: *Upper bounds on the unknown distances, shown as dashed lines, are derived from the known distances, shown as solid lines. a) The exact distance from c_i to the current center point is recomputed in each pass. b) The accumulated movement from c_i is used instead.*

farthest distance $\max(D(c_{i+m}, p \in P))$ is needed. We simply let $f = r_{i+m}$, where r_{i+m} is the current radius. Assuming the distance from an earlier center point c_i to a point p_j is currently stored in position j of the distance array, the following condition can be tested before evaluating the exact distance $D(c_{i+m}, p_j)$:

$$D(c_i, p_j) + D(c_i, c_{i+m}) < f.$$

When this is fulfilled, it is impossible that $D(c_{i+m}, p_j) \geq f$ holds, and the computation of the exact distance can be skipped. This is an immediate consequence of the triangle inequality, which states that

$$D(c_i, p_j) + D(c_i, c_{i+m}) \geq D(c_{i+m}, p_j).$$

This type of filtering is illustrated in Figure 3a., where the distance to a point p_j is computed in pass i, and then the triangle inequality gives approximate distance measures to p_j in the next three passes. Given that the filtering condition succeeds, the unknown distances shown with dashed lines are never computed.

Whenever an actual distance has to be computed for p_j, it is also cached in the distance array simply by overwriting the previously cached value. Note that each stored distance is associated with a certain center point c_i, which must be identified during filtering. Therefore, the value of i is cached as well together with each computed distance. Furthermore, if a computed distance is the largest encountered so far, the point is stored as the current farthest point. When all points have been processed this leaves us with the true farthest point, as well as a distance array that has been updated for all points where the filtering failed.

Two possible drawbacks of this technique are that the distances between all possible pairs of center points are computed, and that the complete sequence of computed center points must be stored. In the algorithms considered here, this filtering method thus requires $O(d/\varepsilon^2)$ time and $O(d/\varepsilon)$

storage. Depending on which MEB algorithm is used, this may degrade the theoretical time complexity of the algorithm. Nevertheless, for sufficiently large point sets and reasonable ε values, this overhead is expected to have little effect on the run time in practice, even in cases when the filtering effectiveness is low.

3.2. Filtering with accumulated distances

An alternative filtering method can be designed, which avoids the above-mentioned potential problems of the previous approach by using a less tight upper bound. In this approach, the distance array stores upper bounds on the distances to earlier center points as opposed to the exact distances. As before, the distance array is initialized with the exact distances from the first query point to all input points. Then in each pass, all these cached distances are incremented with the movement of the center point from the previous pass. Thus, assuming c_i is the last center point from which the distance to a point p_j was computed, the following filtering condition is tested in pass $i+m$:

$$D(c_i,p_j)+D(c_i,c_{i+1})+D(c_{i+1},c_{i+2})+\ldots \\ \ldots+D(c_{i+m-1},c_{i+m})<f,$$

where the sum in the left-hand side represents the value stored for point p_j. When the condition fails, the exact distance is computed and the accumulated bound in the distance array is reset. In Figure 3b, this type of filtering is illustrated. The distance from c_i to p_j is computed in pass i, and then during the following three passes, the consecutive movements of the center points are accumulated in the distance array and used to find approximate distance measures to p_j. Again, given that the filtering condition is fulfilled, the unknown distances to p_j in the next three passes are never computed.

Clearly, the accumulation tends to make the approximate distances less tight compared to the those achieved by the first approach described in the previous section. But similarly to the previous approach, they should improve gradually as the movement of the center point decreases.

Of course, the main advantage of this approach is that it avoids the $O(d/\varepsilon^2)$ computation cost and $O(d/\varepsilon)$ storage cost introduced by the first method above. Thus, incorporating this acceleration technique in farthest point-based MEB algorithms will not affect their theoretical asymptotic time complexity. Of course, storing the auxiliary distance array still introduces an $O(n)$ storage cost.

3.3. Cauchy-Schwarz inequality filtering

Nielsen and Nock [NN04] proposed a filtering criterion based on an upper bound on the Euclidean distance, derived as

$$\begin{aligned}\|p_j-c_i\| &= \sqrt{\|p_j\|^2+\|c_i\|^2-2\langle p_j,c_i\rangle} \\ &\leq \sqrt{\|p_j\|^2+\|c_i\|^2+2\|p_j\|\|c_i\|}.\end{aligned} \tag{1}$$

The bound follows from the Cauchy-Schwarz inequality, which states that for two vectors u and v, it holds that $|\langle u,v\rangle|\leq\|u\|\|v\|$. By computing $\|p\|$ for each $p\in P$ at the beginning of the algorithm, and recomputing $\|c_i\|$ before each farthest-point query, the upper bound in Equation 1 can be computed in constant time per point. As before, if the upper bound is smaller than the lower bound on the searched farthest distance, there is no need to compute the exact distance.

In general, this method can be expected to be less effective in pruning distance computations than the methods of Sections 3.1 and 3.2. To see why, consider that the bound in Equation 1 too expresses the triangle inequality:

$$\begin{aligned}\|p_j-c_i\| &\leq \sqrt{\|p_j\|^2+\|c_i\|^2+2\|p_j\|\|c_i\|} \\ &= \sqrt{(\|p_j\|+\|c_i\|)^2} \\ &= \|p_j\|+\|c_i\|.\end{aligned}$$

In this interpretation, instead of utilizing center points computed in earlier passes of the algorithm, this filtering method repeatedly applies the triangle inequality to the origin o, in addition to p_j and c_i. Thus, expressed with a general distance measure D, the bound becomes

$$D(c_i,p_j)\leq D(o,p_j)+D(o,c_i).$$

Consequently, the distance $D(c_i,p_j)$ is approximated well only when at least one of $D(o,p_j)$ and $D(o,c_i)$ are sufficiently small. Thus, in order to get effective filtering, either most of the input points must be clustered close to the origin, or the sequence of generated center points must be near the origin. In other cases, the distances tend to be largely overestimated by this bound, leading to poor filtering.

3.4. Distance filtering on the GPU

All of the above distance filtering methods require an additional filtering step before invoking the distance computation kernel in each pass. In this step, it is determined in parallel which of the points in P need to have their distances computed exactly. Although the details of how this is determined differs between the methods, the result is a vector b of n binary values, where $b_j=1$ if the distance to point p_j needs to be computed, and $b_j=0$ otherwise. The vector b then goes through a parallel compaction step, which turns it into a vector x containing all indices j such that $b_j=1$. For example, given a vector

$$b=(0,1,0,1,1,0,0,1),$$

the compaction yields

$$x = (2,4,5,8),$$

assuming 1-based indexing. The vector x is then provided as an argument to the distance computation kernel, which processes the rows of the matrix P indirectly through x. In the case of the filtering methods of Sections 3.1 and 3.2, the vector x is then used once again to store the computed distances back into the right positions of the distance array.

4. Experiments

The discussed techniques were evaluated in practice on a laptop equipped with an Intel i7-3820QM processor (2.7 GHz) as well as an Nvidia Quadro K4000M GPU, whose specifications are listed in Figure 4. Sequential CPU implementations, written in C++ and compiled with Visual Studio 2012, as well as parallel GPU implementations, based on CUDA version 6.0, were tested. In addition, all three of the distance filtering methods discussed in Section 3 were evaluated in both the CPU and the GPU versions. Finally, two implementations following the approach outlined in Section 2 were included, using the column-major and row-major (transposed) versions of SGEMV (single-precision GEMV) in cuBLAS. Throughout all experiments, single-precision floating-point was used, and the approximation quality of the balls was set to $\varepsilon = 10^{-3}$. No robustness issues were observed in the experiments.

To tune the performance of our hand-written kernels, we executed an automatic process reminiscent of auto-tuning (cf. [Sør12, DO12]) to find good choices for the kernel launch parameters, as well as to select between our two distance computation kernels. For each pair of d and n included in the experiments, we executed both kernels using different sets of parameters and kept track of the most efficient configuration. The first kernel was run using all combinations of the following parameters:

$$t_d \in \{16,32,64\},$$
$$blockDim.y \in \{\frac{t_d}{1}, \frac{t_d}{2}, \frac{t_d}{4}, \ldots, 1\},$$
$$w_x, w_y \in \{1,2,4,\ldots,512\}.$$

The kernel designed for tall matrices was evaluated with the following parameters:

$$t_y \in \{16,32,64,\ldots,512\},$$
$$blockDim.y \in \{\frac{t_y}{1}, \frac{t_y}{2}, \frac{t_y}{4}, \ldots, 1\},$$
$$w_y \in \{1,2,4,\ldots,512\}.$$

Of course, many combinations of the parameters above are not valid, either due to hardware limitations or because they are not applicable to the particular problem size, so these were skipped. The parameters selected for each problem size were then used regardless of whether filtering was enabled or not.

Architecture	Kepler
SMs	5
CUDA cores	960
Clock rate	600 MHz
Memory clock rate	1.4 GHz
Memory bus width	256 bits

Figure 4: *Specifications for Nvidia's Quadro K4000M GPU.*

4.1. Moderate to high dimensions

The results from SIMPLEAPXBALL and FASTAPXBALL for moderate- to high-dimensional cases are shown in the left and right columns of Table 1, respectively. Nielsen and Nock's distance filtering method is denoted by NN, and our own filtering methods from Sections 3.1 and 3.2 are denoted by TI and TI2, respectively. For each of the selected pairs of d and n, the algorithms were executed ten times on different input sets randomly generated with a uniform distribution in $[-1,1]^d$. The table shows average figures from all ten runs: the number of passes k, the total number of full distance computations (d.c.) performed during the farthest-point queries as a fraction of $k \times n$ (which is the number of such computations in the non-filtering version), the execution time in seconds, and the speedup factor s. The latter parameter gives the average of the speedup factors computed relative to the sequential CPU version that does not use distance filtering.

The non-filtering GPU adaptations of the algorithms show speedups of up to 11×. The cuBLAS implementations exhibit good performance as long as the dimension is quite high, especially the column-major version. Judging from the case $d = 10$, however, the GEMV kernels in cuBLAS seem less optimized for tall matrices. The GPU implementations based on our custom kernels are significantly faster than both of the cuBLAS implementations in this case.

In $d = 10$, the tall matrix kernel was selected in the tuning process, with the parameters t_y, $blockDim.y$, and w_y chosen as shown in parentheses for that case in Table 1. In the remaining cases, the more general kernel proved to be more efficient. Similarly, the parameters t_d, $blockDim.y$, w_x, and w_y selected for this kernel for each problem size are shown in parentheses in the table. In general, the former kernel seems to be the most efficient in dimensions up to $d \approx 20$, as indicated by additional test runs. Notice that the launch parameters for the more general kernel were consistently selected so as to give only one thread block in the horizontal direction, i.e., such that $t_x \times w_x \geq d$. It seems natural that this is an efficient division of labor among the thread blocks, as it allows each thread to sum up its computed squared differences along an entire tile row using registers before the reduction in shared memory takes place. Thus, with only one thread block in the horizontal direction, the number of such

SIMPLEAPXBALL			
$d = 10, n = 10^6, k = 728.1$			
algorithm	**d.c.**	**time**	s
CPU	1.000	6.650	1.0
CPU, NN	0.000	0.724	9.2
CPU, TI	0.004	1.384	4.8
CPU, TI2	0.021	1.764	3.8
cuBLAS, r.m.	1.000	5.752	1.2
cuBLAS, c.m.	1.000	2.446	2.7
GPU $(128, 16, 16)$	1.000	0.877	7.6
GPU, NN	0.003	0.467	14.2
GPU, TI	0.005	0.594	11.2
GPU, TI2	0.022	0.632	10.6
$d = 100, n = 10^5, k = 769.7$			
algorithm	**d.c.**	**time**	s
CPU	1.000	5.888	1.0
CPU, NN	0.166	1.685	3.7
CPU, TI	0.014	0.272	21.7
CPU, TI2	0.047	0.605	9.7
cuBLAS, r.m.	1.000	1.219	4.8
cuBLAS, c.m.	1.000	0.855	6.9
GPU $(32, 8, 4, 32)$	1.000	0.962	6.1
GPU, NN	0.195	0.568	10.5
GPU, TI	0.020	0.473	12.5
GPU, TI2	0.055	0.520	11.4
$d = 500, n = 10^5, k = 618.5$			
algorithm	**d.c.**	**time**	s
CPU	1.000	25.106	1.0
CPU, NN	0.885	22.811	1.1
CPU, TI	0.040	1.301	19.3
CPU, TI2	0.094	2.837	8.8
cuBLAS, r.m.	1.000	2.954	8.5
cuBLAS, c.m.	1.000	2.314	10.9
GPU $(32, 8, 16, 32)$	1.000	2.376	10.6
GPU, NN	0.906	3.090	8.1
GPU, TI	0.060	0.836	30.2
GPU, TI2	0.114	0.941	26.7
$d = 1000, n = 10^4, k = 407.9$			
algorithm	**d.c.**	**time**	s
CPU	1.000	3.346	1.0
CPU, NN	1.000	3.362	1.0
CPU, TI	0.120	0.493	6.8
CPU, TI2	0.199	0.710	4.7
cuBLAS, r.m.	1.000	0.478	7.0
cuBLAS, c.m.	1.000	0.422	7.9
GPU $(32, 8, 32, 16)$	1.000	0.395	8.5
GPU, NN	1.000	0.547	6.1
GPU, TI	0.174	0.590	5.7
GPU, TI2	0.249	0.484	6.9
$d = 5000, n = 10^4, k = 348.3$			
algorithm	**d.c.**	**time**	s
CPU	1.000	14.354	1.0
CPU, NN	1.000	14.404	1.0
CPU, TI	0.302	4.655	3.1
CPU, TI2	0.402	5.844	2.5
cuBLAS, r.m.	1.000	1.609	8.9
cuBLAS, c.m.	1.000	1.346	10.7
GPU $(32, 8, 256, 16)$	1.000	1.333	10.8
GPU, NN	1.000	1.668	8.6
GPU, TI	0.434	1.990	7.2
GPU, TI2	0.516	1.489	9.6

FASTAPXBALL			
$d = 10, n = 10^6, k = 14.9$			
algorithm	**d.c.**	**time**	s
CPU	1.000	0.137	1.0
CPU, NN	0.002	0.025	5.5
CPU, TI	0.222	0.063	2.2
CPU, TI2	0.344	0.105	1.3
cuBLAS, r.m.	1.000	0.126	1.1
cuBLAS, c.m.	1.000	0.059	2.3
GPU $(128, 16, 16)$	1.000	0.019	7.4
GPU, NN	0.150	0.026	5.1
GPU, TI	0.260	0.023	6.0
GPU, TI2	0.352	0.024	5.6
$d = 100, n = 10^5, k = 56.2$			
algorithm	**d.c.**	**time**	s
CPU	1.000	0.440	1.0
CPU, NN	0.361	0.227	2.0
CPU, TI	0.191	0.129	3.4
CPU, TI2	0.280	0.183	2.4
cuBLAS, r.m.	1.000	0.107	4.1
cuBLAS, c.m.	1.000	0.082	5.4
GPU $(32, 8, 4, 32)$	1.000	0.080	5.5
GPU, NN	0.519	0.086	5.1
GPU, TI	0.270	0.065	6.8
GPU, TI2	0.342	0.069	6.4
$d = 500, n = 10^5, k = 127$			
algorithm	**d.c.**	**time**	s
CPU	1.000	5.237	1.0
CPU, NN	0.953	5.093	1.0
CPU, TI	0.187	1.167	4.5
CPU, TI2	0.274	1.674	3.1
cuBLAS, r.m.	1.000	0.703	7.5
cuBLAS, c.m.	1.000	0.576	9.1
GPU $(32, 8, 16, 32)$	1.000	0.560	9.4
GPU, NN	0.968	0.749	7.0
GPU, TI	0.274	0.370	14.2
GPU, TI2	0.349	0.393	13.3
$d = 1000, n = 10^4, k = 149.1$			
algorithm	**d.c.**	**time**	s
CPU	1.000	1.348	1.0
CPU, NN	1.000	1.359	1.0
CPU, TI	0.277	0.489	2.8
CPU, TI2	0.369	0.599	2.3
cuBLAS, r.m.	1.000	0.307	4.4
cuBLAS, c.m.	1.000	0.288	4.7
GPU $(32, 8, 32, 16)$	1.000	0.269	5.0
GPU, NN	1.000	0.328	4.1
GPU, TI	0.394	0.339	4.0
GPU, TI2	0.469	0.312	4.3
$d = 5000, n = 10^4, k = 269.3$			
algorithm	**d.c.**	**time**	s
CPU	1.000	11.690	1.0
CPU, NN	1.000	11.736	1.0
CPU, TI	0.364	4.843	2.4
CPU, TI2	0.462	5.780	2.0
cuBLAS, r.m.	1.000	1.853	6.3
cuBLAS, c.m.	1.000	1.647	7.1
GPU $(32, 8, 256, 16)$	1.000	1.618	7.2
GPU, NN	1.000	1.873	6.2
GPU, TI	0.516	2.055	5.7
GPU, TI2	0.588	1.770	6.6

Table 1: *Experimental results for uniformly distributed input. Timings are given in seconds.*

SIMPLEAPXBALL					
model	n	k	CPU	cuBLAS	GPU
Lucy	14.0M	15	0.595	0.256	0.061
Thai Statue	5.0M	584	8.215	2.969	0.929
Vase	4.6M	721	9.435	3.424	1.116
Asian Dragon	3.6M	730	7.416	2.727	0.885
Goblet	1.0M	830	2.354	1.047	0.386

FASTAPXBALL					
model	n	k	CPU	cuBLAS	GPU
Lucy	14.0M	5	0.198	0.120	0.020
Thai Statue	5.0M	8	0.113	0.060	0.013
Vase	4.6M	7	0.088	0.051	0.011
Asian Dragon	3.6M	6	0.061	0.037	0.007
Goblet	1.0M	10	0.028	0.017	0.005

Table 2: *Experimental results for polygon meshes in 3D. The Lucy, Thai Statue, and Asian Dragon models are provided by the Stanford Computer Graphics Laboratory. Timings are given in seconds.*

reductions per row is minimized. Furthermore, it enables the kernel-fusion optimization described in Section 2.1.

All the distance filtering techniques show successful reductions in distance computations under certain circumstances, leading to quite impressive speedups. In $d = 10$, Nielsen and Nock's method is the most effective, but loses its effectiveness rapidly as the dimension grows. Given the uniform distribution of these point sets, this is an expected result, as no clusters tend to be formed around the origin. Furthermore, since it becomes increasingly improbable that c^* occurs in the vicinity of the origin in the higher dimensions, it also becomes less likely that c_i does. The triangle inequality-based methods, on the other hand, show less pruning power in $d = 10$, but give better results in higher dimensions.

In the GPU case, it is clear that the overhead of the additional kernel invocations needed to realize the filtering procedure has a limiting effect on the achieved performance. In fact, the introduction of distance filtering can be seen to decrease performance in several cases, particularly in the higher dimensions, where fewer distance computations are skipped. In this regard, the CPU algorithms have the benefit that testing the filtering condition and updating the cached distances can be done in an integrated fashion during a single traversal of the point set. Nevertheless, several successful cases can be observed as well, indicating a potential of the presented approach.

Noteworthy here is also that the sequential implementations possess an additional opportunity to skip more distance computations compared to the parallel implementations: during the sequential scan for the farthest point, the lower bound on the largest distance is updated continuously as new candidate farthest points are encountered. Thus, the upper distance bounds are compared against a gradually increasing lower bound, whereas in the parallel distance filtering, the same lower bound is used throughout the whole pass. The effects of this optimization can be seen in column d.c. in Table 1 by comparing the figures of the CPU and GPU versions that use the same filtering method.

4.2. Polygon meshes

To evaluate the presented techniques also in low dimensions, and to include more realistic data sets in the experiments, we executed the algorithms with a selection of 3-dimensional polygon meshes as input. The results from this experiment are shown in Table 2. Listed for each case are the number of vertices n in the model, the total number of passes k, and the run time in seconds of three of the evaluated implementations. Included here are the non-filtering CPU versions, the GPU versions based on the column-major kernel in cuBLAS, and the non-filtering GPU versions using the kernel optimized for tall matrices. The parameters used for the latter were $t_y = 512$, $blockDim.y = 64$, and $w_y = 64$ in all runs.

The results from the GPU versions based on the tailor-made distance computation kernel are encouraging, with speedups in the range 5.8–9.8× on both algorithms. The cuBLAS-based versions, on the other hand, give somewhat disappointing speedups of at most 2.8× on SIMPLEAPXBALL and 1.9× on FASTAPXBALL. Again, this indicates a lack of support for tall matrices in the cuBLAS GEMV kernels.

Note that in the 3-dimensional case, less performance

benefits can be expected from using the discussed distance filtering techniques, in the GPU case as well as the CPU case. As it takes only 8 arithmetic operations to compute a squared distance, the relative savings from filtering such a computation is limited. On the above input examples, distance filtering gave occasional performance improvements of up to 2.1× in the CPU version of SIMPLEAPXBALL, and minor slowdowns to modest speedups in the CPU version of FASTAPXBALL.

5. Conclusions

Given the extensive applicability of minimum enclosing ball algorithms in both low and high dimensions, we expect performance studies and speed-up techniques, such as the ones presented in this paper, to be beneficial in several research communities. Clearly, the offloading of the repeated farthest point queries to massively parallel GPUs pays off with speedups up to 11×. Also, the presented algorithmic techniques for distance filtering give additional opportunities for savings in execution time both in the sequential and the parallel implementations.

The proposed distance filtering approaches can be harnessed also in applications using other distance measures than Euclidean distance, as long as the used distance function obeys the triangle inequality. Furthermore, despite that only points sets were considered as input here, it is straightforward to generalize the presented techniques to deal also with ball sets.

In the future, it would also be exciting to compare other possible acceleration techniques with the strategies presented here, such as considering alternative ways of parallelization, different pruning methods [KL], usage of k-farthest points queries in each algorithm pass, and hierarchical searching using multidimensional tree structures (see, e.g., [CPZ97]). This also includes evaluating combinations of the approaches, such as using pruning techniques to eliminate points permanently combined with distance filtering on the remaining points, as well as more aggressive and diverse algorithm parallelization on heterogeneous computing platforms with support for different compute targets (CPU, GPU, and FPGA). Hopefully, such studies could provide a basis for the design of even faster ball computation algorithms.

Acknowledgements

Both authors are supported by a research grant from the Swedish Foundation for Strategic Research (No. IIS11-0060).

References

[BC03] BÂDOIU M., CLARKSON K. L.: Smaller core-sets for balls. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* (2003), pp. 801–802. 1, 2

[BEKS01] BRAUNMULLER B., ESTER M., KRIEGEL H.-P., SANDER J.: Multiple similarity queries: a basic DBMS operation for mining in metric databases. *IEEE Transactions on Knowledge and Data Engineering 13*, 1 (Jan 2001), 79–95. 4

[BH12] BELL N., HOBEROCK J.: Thrust: A productivity-oriented library for CUDA. In *GPU Computing Gems Jade Edition*, Hwu W.-m. W., (Ed.). Morgan Kaufmann Publishers Inc., 2012, pp. 359–371. 2

[BK73] BURKHARD W. A., KELLER R. M.: Some approaches to best-match file searching. *Communications of the ACM 16*, 4 (Apr. 1973), 230–236. 4

[BS98] BERMAN A., SHAPIRO L.: Selecting good keys for triangle-inequality-based pruning algorithms. In *IEEE International Workshop on Content-Based Access of Image and Video Database* (Jan 1998), pp. 12–19. 4

[CPZ97] CIACCIA P., PATELLA M., ZEZULA P.: M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases* (1997), Morgan Kaufmann Publishers Inc., pp. 426–435. 9

[DDG*] DONGARRA J., DONG T., GATES M., HAIDAR A., TOMOV S., YAMAZAKI I.: MAGMA: a new generation of linear algebra library for GPU and multicore architectures. 2

[DO12] DAVIDSON A., OWENS J.: Toward techniques for auto-tuning GPU algorithms. In *Applied Parallel and Scientific Computing*, vol. 7134 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 110–119. 6

[Gär99] GÄRTNER B.: Fast and robust smallest enclosing balls. In *Proceedings of the 7th Annual European Symposium on Algorithms* (1999), Springer-Verlag, pp. 325–338. 2

[HS03] HJALTASON G. R., SAMET H.: Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems 28*, 4 (Dec. 2003), 517–580. 4

[KL] KÄLLBERG L., LARSSON T.: Improved pruning of large data sets for the minimum enclosing ball problem. *Graphical Models (to appear)*. 2, 9

[KMY03] KUMAR P., MITCHELL J. S. B., YILDIRIM E. A.: Approximate minimum enclosing balls in high dimensions using core-sets. *Journal of Experimental Algorithmics 8* (2003). 1, 2

[LK13] LARSSON T., KÄLLBERG L.: Fast and robust approximation of smallest enclosing balls in arbitrary dimensions. *Computer Graphics Forum 32*, 5 (2013), 93–101. 1

[NN04] NIELSEN F., NOCK R.: Approximating smallest enclosing balls. In *Proceedings of International Conference on Computational Science and Its Applications (ICCSA)* (2004), vol. 3045 of *Lecture Notes in Computer Science*, Springer. 5

[PS85] PREPARATA F. P., SHAMOS M. I.: *Computational Geometry: An Introduction*. Springer-Verlag New York, Inc., 1985. 1

[Sør12] SØRENSEN H. H. B.: High-performance matrix-vector multiplication on the GPU. In *Euro-Par 2011: Parallel Processing Workshops*, vol. 7155 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 377–386. 6

[TKK07] TSANG I. W., KOCSOR A., KWOK J. T.: Simpler core vector machines with enclosing balls. In *Proceedings of the 24th International Conference on Machine Learning* (2007), ACM, pp. 911–918. 1

[Yıl08] YILDIRIM E. A.: Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization 19*, 3 (Nov. 2008), 1368–1391. 1, 2

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

GPU-based ray-casting of non-rigid deformations: a comparison between direct and indirect approaches

F. M. M. Marreiros[1,2] and Ö. Smedby[1,2,3]

[1]Center for Medical Image Science and Visualization (CMIV), Linköping University, Sweden
[2]Department of Science and Technology (ITN) - Media and Information Technology (MIT) , Linköping University, Sweden
[3]Department of Radiology (IMH), Linköping University, Sweden

Abstract

For ray-casting of non-rigid deformations, the direct approach (as opposed to the traditional indirect approach) does not require the computation of an intermediate volume to be used for the rendering step. The aim of this study was to compare the two approaches in terms of performance (speed) and accuracy (image quality).
The direct and the indirect approach were carefully implemented to benefit of the massive GPU parallel power, using CUDA. They were then tested with Computed Tomography (CT) datasets of varying sizes and with a synthetic image, the Marschner-Lobb function.
The results show that the direct approach is dependent on the ray sampling steps, number of landmarks and image resolution. The indirect approach is mainly affected by the number of landmarks, if the volume is large enough. These results exclude extreme cases, i.e. if the sampling steps are much smaller than the voxel size and if the image resolution is much higher than the ones used here. For a volume of size 512×512×512, using 100 landmarks and image resolution of 1280×960, the direct method performs better if the ray sampling steps are approximately above 1 voxel. Regarding accuracy, the direct method provides better results for multiple frequencies using the Marschner-Lobb function.
The conclusion is that the indirect method is superior in terms of performance, if the sampling along the rays is high, in comparison to the voxel grid, while the direct is superior otherwise. The accuracy analysis seems to point out that the direct method is superior, in particular when the implicit function is used.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling ; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism ;

1. Introduction

Since the introduction of the Graphics Processing Units (GPUs) with programming functionalities, we have seen an explosion of algorithms being ported to these devices. The NVIDIA's Compute Unified Device Architecture (CUDA) enables the usage of these devices. The necessary software programming interface to access the hardware is available in "C for CUDA" [Cud10] (C with NVIDIA extensions and certain restrictions), but many language bindings are also available.

One algorithm that largely benefited from programmable GPU architectures is volume ray-casting. Volume ray-casting is a well known Direct Volume Rendering (DVR) technique used mainly to render regular grid volume data, but it also can be used to render implicit surfaces [Sig06]. Examples of such data are medical image modalities like Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). Ray-casting operates by casting one ray per pixel, and each ray is computationally independent, thus benefiting greatly from highly parallel programmable GPU architectures, e.g. shaders or CUDA. In this paper, we will use mainly isosurface ray-casting, a special case of volume ray-casting also known as *first-hit ray-casting*, with regular grid and implicit surface data. The same conclusions can be generalized for other DVR types.

Ray-casting is mostly used to generate images of static objects; if the volume is deformed over time, the traditional approach (indirect) is to compute an entire new volume for each time step and render it. The intensities at the corresponding positions in the original volume have to be calculated for each voxel. In the literature there are several voxel-based non-rigid deformation methods, which can also be used for specific tasks like volume morphing or volume registration. In this paper we focus on the Radial Basis Functions (RBF) method, one of the most widely used non-rigid deformation methods in the medical field that also has fast GPU implementations.

We compare the direct and indirect methods exploiting in both approaches the benefits of the GPU hardware architecture. The comparisons are provided in terms of performance and accuracy.

The motivation is to study if it is possible to enable deformation interaction with a frame rate that allows visual feedback (at least 1 frame-per-second). There are two main application scenarios considered: first tracking of points in real-time to guide the deformation; second for artistic purposes where the artist/animator is controlling the position of the points to guide the deformation. In both cases visual feedback is critical, thus a proper frame rate is essential.

2. Related work

2.1. GPU-based ray-casting

Several GPU-based ray-casting techniques have been proposed in the literature. An overview of the major developments in this area, including acceleration techniques like early-ray termination or empty space skipping, can be found in [HLSR09]. From a historical perspective, some of the initial implementations of GPU-based ray-casting with acceleration techniques are from Krüger and Westermann [KW03] and Röttger et al. [RGW*03]. Both implementations use shaders and early-ray termination, but their empty space skipping implementations are different. Krüger and Westermann [KW03] use an octree for empty-space skipping. Röttger et al. [RGW*03] calculate the ray's intersections with the volume bounding box and use them as the starting and end points of the rays. Li et al. [LMK03] compared several empty space skipping and occlusion clipping techniques for texture-based volume rendering.

More recently, a combination of CUDA and OpenGL can be used to compute the images (CUDA side) and pass them directly to be rendered (OpenGL), bypassing the traditional rendering pipeline. This uses the OpenGL Pixel Buffer Object (PBO) to store the texture ensuring that the generated images reside in the GPU as described in the chapter OpenGL Interoperability of the CUDA Programming Guide and the example code [Cud10]. There are already advanced CUDA ray-casting implementations [MRH10]. To be noticed, in the CUDA SDK a simple ray-casting example can also be found. Although CUDA was used here the implementation could also be made using OpenCL or shaders, with similar results expected.

2.2. Non-rigid deformations

A considerable number of non-rigid deformations methods have been proposed. A good survey of physically based deformable models can be found in [NMK*05]. Most of these approaches are dependent on a mesh representation, but a subset of mesh-free methods exist. Besides the physically based methods, many others exist that in general try to optimize the transformation minimizing some constrains. In this work we will use a non physically based mesh-free method.

A distinguishing factor between them is the data they use to drive the deformation. They can thus be divided into point-based, surface-based, intensity-based, etc. In this paper, we will focus on the point-based approach, in particular the Radial Basis Function (RBF) method, and one specific RBF: the Thin Plate Spline (TPS). A comparison of some of these methods can be found in Fang et al. [FSRR00] who also describe an implementation that does not require the intermediate volume. Further methods that also do not require the intermediate volume and are tied to the rendering stage, include ray deflectors [KY97], free-form [CHM03], free-form and texture mapping [WRS01], spatial transfer functions [CSW*03], constrained illustrative [CSC10], and 3D chainmail algorithms [Gib97]. From the last set of methods the ones that can use homologous points to control the deformation in the same way as with the TPS are the free-form methods [WRS01], [CHM03]. These share many similarities with our work. In [WRS01] no intermediate volume is calculated but instead a shape model and an appearance model are used. The shape model is a tessellation of the surface enclosing the object and the appearance a 3D texture of the volume. The second free-form method [CHM03] computes a deformed ray in the original object space and approximates the ray path by polylines.

Recent, RBFs have been implemented in the GPU, for warping and non-linear registration. Levin et al. [LDS04], Rowland [Row07] and Lapeer et al. [LSR10] all use shaders or a shader/CPU combination for comparisons. One should notice that these implementations could be tuned to trade speed for accuracy, a feature essential in some application scenarios. All these approaches require the computation of a new volume (indirect), which can possibly be rendered in a subsequent step. The voxel positions in the new volume (Target) have to be evaluated, and the intensity of the corresponding position in the original volume (Source) fetched and assigned to the new voxel. This process is known as *backward mapping*.

We also implement the indirect method for comparison purposes using CUDA. For the direct approach, we evaluate and perform backward mapping at the ray-casting sampling positions. This has the drawback of being useful only

for rendering purposes, but if rendering is the primary goal, then our approach may prove valuable. In addition, we can control image quality and frame rate using the traditional ray-casting parameters.

3. Materials and Methods

3.1. RBF

RBFs are well known for their smooth interpolation properties. They are able to smoothly interpolate point positions in 3D to generate a surface, as seen, e.g., in [CBC*01]. We use the RBF as a smooth mapping function, to determine corresponding positions in two 3D volume spaces. Each space has a set of points, also referenced in the literature as landmarks or centers, necessary to drive the mathematics of the method. Both spaces need to have the same number of landmarks and each landmark in one space (Template or Source) has a corresponding homologous landmark (e.g. the same anatomical location) in the mapped space (Target). An initial use of RBFs, in particular the thin-plate spline (TPS) for modeling of biological shape change, was proposed by Bookstein [Boo89]. Bookstein formulated the TPS algebra in 2D; later extensions by Lapeer and Prager [LP00] enabled the use in 3D. We can also find in [LP00] the concepts of forward and backward transformation.

In algebraic terms, the backward mapping can be described by the following equation:

$$(x_s, y_s, z_s) = f(x_t, y_t, z_t) \tag{1}$$

where: (x_t, y_t, z_t) are target point coordinates in Cartesian space;

(x_s, y_s, z_s) are source point coordinates in Cartesian space;

$f(x, y, z)$ is a function mapping points in the target space to the source space. This function needs to be decomposed, solved and evaluated separately per coordinate, in the 3D case:

$$f(x_t, y_t, z_t) = [f'_X(x_t, y_t, z_t), f'_Y(x_t, y_t, z_t), f'_Z(x_t, y_t, z_t)] \tag{2}$$

where: $f'_X(x_t, y_t, z_t) = x_s; f'_Y(x_t, y_t, z_t) = y_s; f'_Z(x_t, y_t, z_t) = z_s$. Per coordinate the function f' has the following equation:

$$f'_*(x, y, z) = a_{1*} + a_{2*}x + a_{3*}y + a_{4*}z + \sum_{i=1}^{n} \lambda_{i*}\phi(|P_i - (x, y, z)|) \tag{3}$$

where:

$*$ is an index for the individual coordinates, it should be replaced by X, Y or Z;

$a_{1*}, a_{2*}, a_{3*}, a_{4*}$ are the coefficients of an affine transformation;

n the number of landmarks;

λ_{i*} are the weights;

P_i a landmark point - in backward mapping the target landmarks, in forward mapping the source landmarks;

ϕ the radial basis function.

The TPS function in 3D is given by the following equation.

$$\phi(r) = r \tag{4}$$

Using the landmark values in equation (2), we obtain a linear system of equations that can be directly solved by LU decomposition. In this way, the unknown affine coefficients and weights can be obtained for each coordinate ([Boo89], [LP00]). To solve the linear system of equations we used a C++ linear algebra library named Armadillo [San10] in CPU, as the time needed to solve the system is very short. The greatest amount of time is spent in the evaluation of the new point positions performed in GPU.

Knowing the coefficients of the affine transformation and the weights, we can evaluate the point coordinates directly using equation (2). The purpose of this mapping is to obtain the corresponding image intensity values of the target points. Using the indirect method, we want to know the intensity values of the voxels positioned on a regular grid (target volume image). Since the target voxel positions are known, we can use the mapping function to obtain the corresponding position in the source volume (equation (2)).

$$I(x_{t_i}, y_{t_j}, z_{t_k}) = I(x'_s, y'_s, z'_s); i = 1...n_x, j = 1...n_y, k = 1...n_z$$
$$(x'_s, y'_s, z'_s) = f(x_{t_i}, y_{t_j}, z_{t_k}) \tag{5}$$

where: n_x,n_y,n_z are the dimensions of each volume coordinate.

The mapped point in the source image may lie between voxel positions; if this is the case, then trilinear interpolation is required to obtain the intensity value. The trilinear interpolation is performed by the GPU in hardware, considerably reducing computation time. Still, evaluating all voxels in the volume requires considerable computational power, since the RBF depends on the number of landmarks.

3.2. Ray-casting of non-rigid deformations

The traditional (indirect) way of using ray-casting and non-rigid deformations is to first calculate an entire new volume by backward mapping all the voxels and then render it. Our direct approach is different: we evaluate (backward mapping and check the intensity in source image) at the sampling points locations along the ray. The evaluation of the gradients, used for illumination, is also calculated in this way. In our case, a 3D Sobel filter is used to estimate the gradient. This filter requires 26 neighbor points, which are backward mapped to obtain the appropriate intensity value in the source image. The indirect method also uses the Sobel filter, but in the intermediate volume space. Furthermore early-ray termination is used to stop the ray when the maximum opacity level is reached and adaptive sampling to increase the precision of the isosurface position by increasing the sampling

rate by eight times in positions where the intensity values approximate the isosurface threshold.

3.3. Volume bounding boxes

The position and size of the target bounding box is defined by the user. This is done to allow the user to select the appropriate dimensions. In cases where the source and target bounding boxes have the same size, an expansion of the target volume would possibly clip it. If this is the case, the user manually increases the size of the bounding box and the problem is solved.

The bounding boxes serve also to determine where the rays initiate and possibly terminate, if the surface was not hit. This is performed by computing where the ray intercepts the faces of the bounding boxes to determine the points closer and further from the center of projection.

3.4. Accelerating the RBF evaluation

As mentioned earlier, with the direct approach the RBF is evaluated at the points along the sampling ray and with the indirect method at the point locations of the voxels in the intermediate volume. The evaluation requires the computation of a sum that depends on the number of landmarks. In order for these operations to be performed as fast as possible, we make use of CUDA texture memory. We then need to store in texture memory the coefficients of the affine part, the target landmarks and the weights. Note that these variables are required for each ray/voxel, because they all are needed for the deformation computation of each sample point. This CUDA memory usage greatly improves performance since texture memory is cacheable; the alternative would be to get these variables from global memory every time we need to use them. This would introduce a great penalty due to memory latency and bandwidth.

4. Results

To compare consistently the indirect and direct methods in terms of performance, independently of the volume data we have used empty volumes with source and target landmarks with the same values (no deformation) and three volume sizes:

- 512×512×512
- 512×512×256
- 512×512×128

with isotropic voxels of 0.5 mm per coordinate. The performance tests were dependent on:

- ray sampling step
- number of landmarks
- image resolution (number of rays)

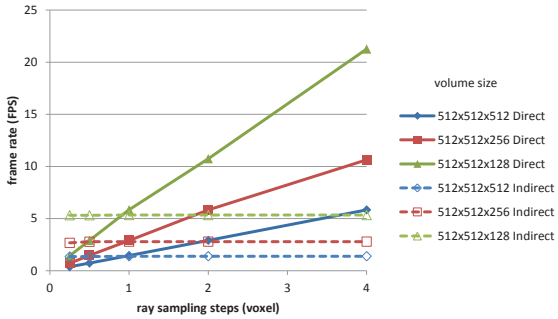


Figure 1: *Performance test: variation of ray sampling steps. The image resolution used was 1280×960 and the number of landmarks 100. The direct and indirect methods are compared.*

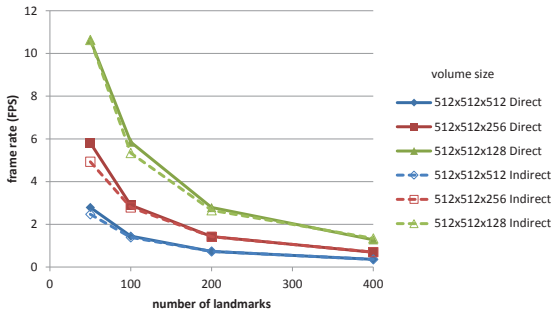


Figure 2: *Performance test: variation of number of landmarks. The image resolution used was 1280×960 and the ray sampling step 1.0 voxel. The direct and indirect methods are compared.*

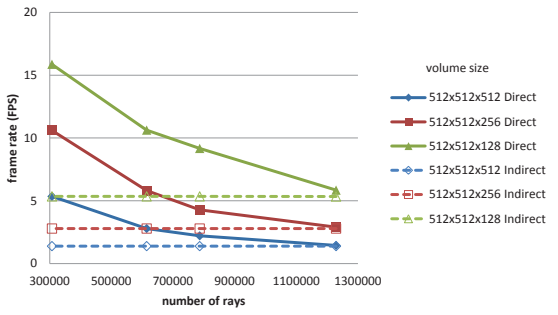


Figure 3: *Performance test: variation of number of rays. The resolutions used were: 1280×960 (1,228,800 rays), 1024×768 (786,432 rays), 960×640 (614,400 rays), 640×480 (307,200 rays). The ray sampling step used was 1.0 voxel and the number of landmarks 100. The direct and indirect methods are compared.*

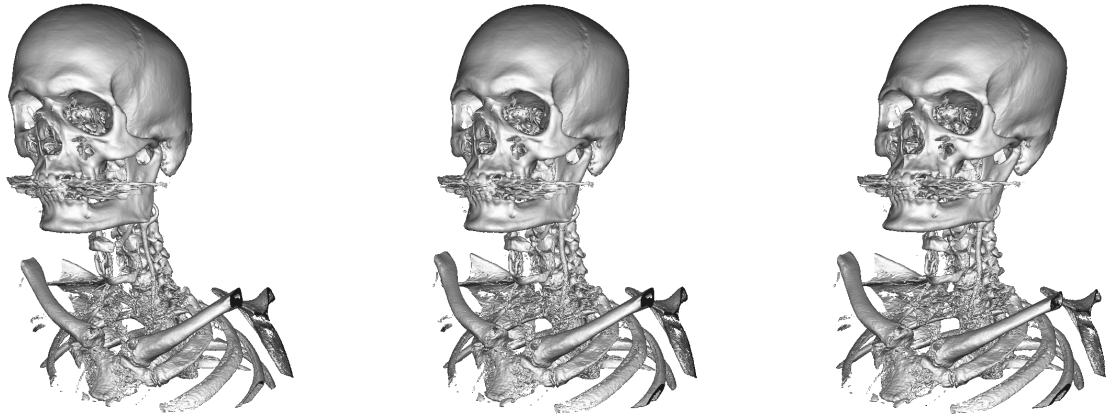


Figure 4: *Examples of a TPS non-rigid deformation, using a CT of a head and partially chest (512×512×512) dataset. The images on the left and right are deformed and the center one is not deformed.*

Care was taken to place the volumes projections (by controlling the camera position) fully within the image space. The graphics card used was an NVIDIA Geforce GTX 680 with 1536 CUDA cores, 2GB of dedicated video memory and CUDA architecture 2.0. Figures 1, 2, 3 present the results found for the several tests. To analyze the performance we look at the frame rate in frames per second (FPS). Figure 1 shows that the direct method is linearly dependent on the ray sampling steps: the frame rate varies linearly for all datasets, but with different slopes. For the indirect method, there are no significant differences if the volume is large enough. These results exclude extreme cases, i.e. if the sampling steps are much smaller than the voxel size. In these cases it is expected that the time to compute the deformation is neglectable in comparison to the ray traversal producing very low framerates. Figure 2 shows that both methods are dependent on the number of landmarks: the frame rate decays exponentially with the increase of landmarks. Finally, in Figure 3 we can see that the frame rate using the direct method is exponentially decaying, while for the indirect method, if the volume is large enough, the frame rate is constant. Extreme cases where the image resolution is much higher than the ones used here are not considered. The same consideration as in the ray sampling steps apply here.

For the direct method, we perform an extra test. The performance with and without early-ray termination was compared, using a CT dataset of the head and part of the chest (512×512×512), presented in Figure 4, where the central image corresponds to the volume without deformation and the remaining with a deformation applied. The results are presented in Figure 5.

In Figure 5 we can see that the increase in performance with early termination is small. In fact, it may even take more time to produce the rendering with the early-ray termination than to traverse the entire empty volume. This is due to adaptive sampling: the sampling rate increases in regions with values near the isosurface threshold, but if the ray does not hit the surface, then the performance penalty is high.

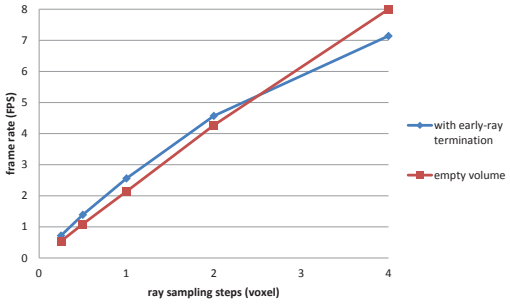


Figure 5: *Performance test: early-ray termination versus an empty volume of the same size, using the direct method and multiple ray sampling steps. The image resolution used was 1280×960 and the number of landmarks 100.*

Besides the performance tests we also tested image quality. A synthetic volume was created using the Marschner-Lobb function [ML94], with a resolution of 512×512×256. To generate the ground truth approximation, we used the direct method and in the source space tested the voxel values with the analytical function with a translation of 1.25 voxel in each coordinate, rendering the values larger or equal to 0.5. To compute the gradient, we used also the Sobel filter, but again testing the values with the analytical function. The offsets of the sampling positions were in all three coordinates 0.5 voxel, and the ray sampling steps 0.05 voxel. The comparison was made using two frequencies: $f_M = 10$ and $f_M = 20$ and $\alpha = 0.25$. The two rendering approaches had as input an implicit function (Marschner-Lobb) or a sampled volume regular grid. The values of the voxels of the

Table 1: *PSNR of the direct and indirect method (with implicit or volume regular grid as input) for the Marschner-Lobb functions with frequencies ($f_M = 10$, $f_M = 20$) and $\alpha = 0.25$.*

PSNR	direct implicit	indirect implicit
$f_M = 10$	+57.4130 dB	+25.3260 dB
$f_M = 20$	+57.1295 dB	+23.5643 dB
	direct regular grid	indirect regular grid
$f_M = 10$	+32.8225 dB	+25.5789 dB
$f_M = 20$	+24.5623 dB	+23.3019 dB

Table 2: *SSIM of the direct and indirect method (with implicit or volume regular grid as input) for the Marschner-Lobb functions with frequencies ($f_M = 10$, $f_M = 20$) and $\alpha = 0.25$.*

SSIM	direct implicit	indirect implicit
$f_M = 10$	0.998835	0.834781
$f_M = 20$	0.998947	0.682028
	direct regular grid	indirect regular grid
$f_M = 10$	0.957126	0.847693
$f_M = 20$	0.718059	0.663997

regular grid were obtained using the analytical function. The translation applied to the ground truth was necessary in order to test the impact of the TPS transformation (the source landmarks and target landmarks have different positions). To produce equivalent results, the target landmarks need also to be translated 1.25 voxel in each coordinate. The results are presented in Figure 6.

To compare the images, we use the image quality metrics peak signal-to-noise ratio (PSNR) and the structural similarity measure (SSIM) [WBSS04]. SSIM takes into account human eye perception. Table 1 and 2 present the results obtained. The images used for these tests were cropped versions (boundaries were cropped) of the images in Figure 6.

5. Discussion

The major contribution of this paper is the comparison of direct and indirect TPS non-rigid deformations and making use of the graphics hardware (CUDA). Regarding direct methods, [FSRR00], [WRS01] and [CHM03] also have implementations that not require the intermediate volume and can use points to control the deformation. Although these seem conceptually similar the main differences are that they use intermediate data structures for acceleration purposes instead we rely on brute force GPU acceleration. Also, we expect a better deformed image quality due to the fact that the intermediate data structures are used for approximation purposes and can trade image quality for speed. The most similar method to ours is the free-form method [CHM03], but a key difference is that the free-form method deforms the rays in the original volume space and then traverse them. In our approach the rays are straight and traversed in the deformed volume space; the sampling positions are backward mapped to obtain the intensity values in the original volume space.

Comparing the two methods in terms of performance and accuracy (image quality), we note that the indirect method is less sensitive to changes in the ray-sampling steps and to image resolution, while both seem to be similarly affected by the number of landmarks. For larger volumes and greater sampling steps, the performance of the indirect method is worse than the direct method. There are also other drawbacks of the indirect method, for instance the inability to trade image quality for frame rate, because the main parameter used for this purpose is the ray sampling steps and for large volumes the indirect method presents almost no change (constant). In contrast, the direct method is very sensitive to the ray-sampling steps, thus allowing easy control of the frame rate. Furthermore, the indirect method requires that two volumes are loaded simultaneously to texture memory, which for larger volumes can present a severe limitation.

In this work, we did not explore faster deformation techniques, as we are interested in preserving image quality and keeping the method as general as possible. Using RBFs fast alternatives exist, that trade image quality for speed, as pointed out in the non-rigid deformations section [Row07], [LSR10]. We could also use the locally bounded Hardy method like in [FSRR00], and RBF with compact support [FRS01]. The choice of the method to be used should depend on the data and the purpose. Our implementation can easily be changed to use some of these methods, if needed.

The second set of tests performed in our study related to the accuracy of both methods. In Figure 6 we can see that differences exist in both methods, although, in some cases, they are rather hard to perceive with the naked eye. Using the image quality metrics PSNR and SSIM (the higher the better), the direct method gives superior results, in particular when the implicit function is used. The worse results of the indirect method can be explained by the double trilinear interpolation: first at the evaluation of each voxel value (backward mapping - generation of intermediate volume) and second in the reconstruction (at each sampling position). It also depends on the resolution with which the intermediate volume is sampled. Also to observe, the direct implicit method is less sensitive to frequency, because no volume grid sampling is performed to obtain the voxel intensities. In this case, the sources of errors reside in the sampling positions along the ray and in numerical rounding errors in the TPS calculation. However, as can be seen, these are rather small errors.

6. Conclusion

A comparison of direct and indirect ray-casting TPS non-rigid deformations was performed. Both methods were im-

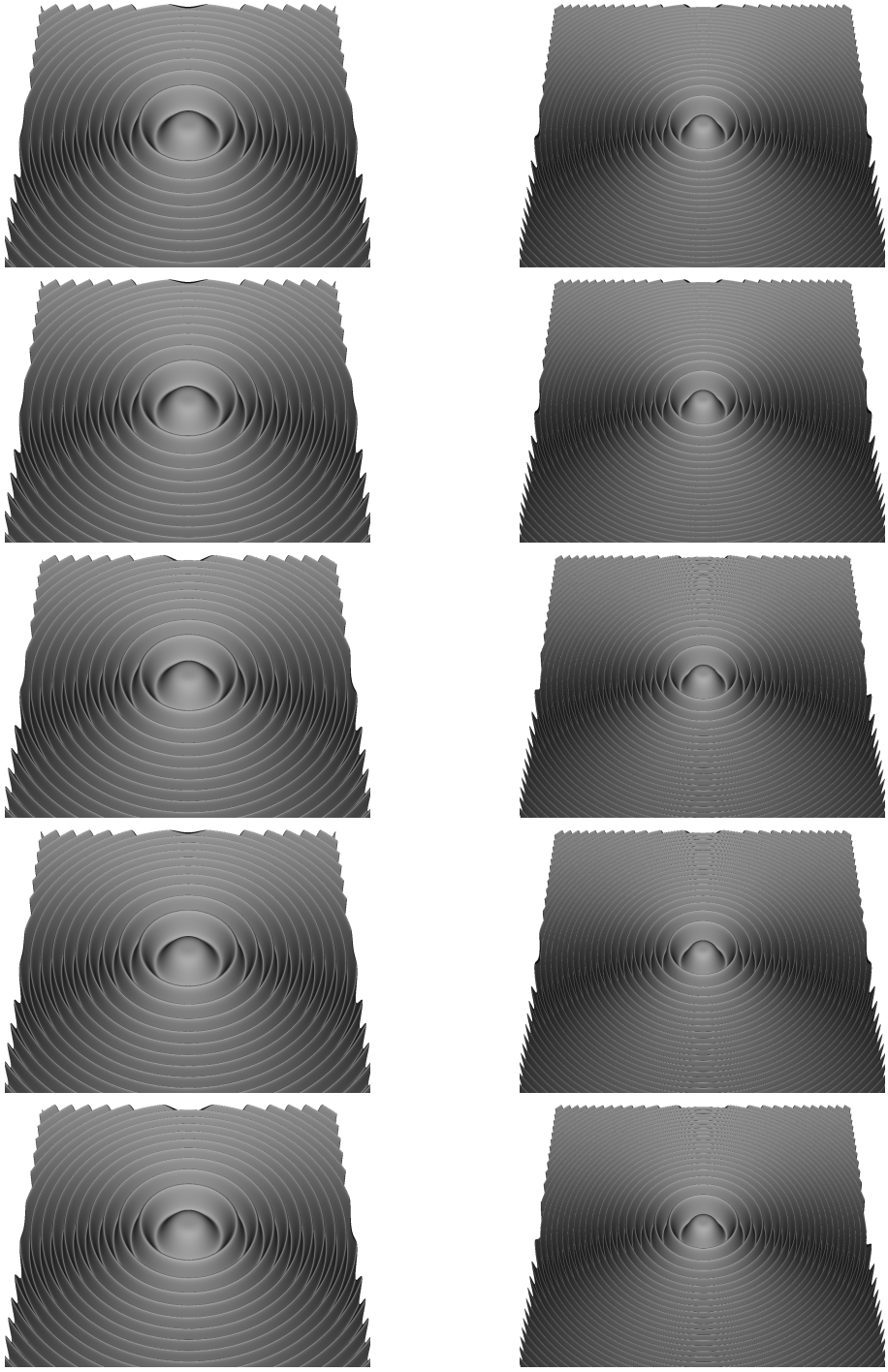


Figure 6: *Marschner-Lobb function comparison, ground truth (top row), direct method with implicit function (second row), direct method with regular grid (third row), indirect method with implicit function (fourth row), indirect method with regular grid (fifth row). The ground truth approximation was generated using the direct method and evaluated in source space with the analytical function with a 1.25 voxel translation, threshold 0.5 and using a Sobel filter for the gradient with offsets of the sampling positions in all tree coordinates 0.5 voxel. The ray sampling steps 0.05 voxel. Two frequencies were used:* $f_M = 10$ *(Left) and* $f_M = 20$ *(Right) and* $\alpha = 0.25$.

plemented in the GPU using the NVIDIA CUDA architecture for acceleration purposes. The comparisons of both methods were made in terms of performance and accuracy. Regarding performance, the indirect method is superior if the sampling along the rays is high, in comparison to the voxel grid, while the direct is superior otherwise. The accuracy analysis seems to point out that the direct method is superior, in particular when the implicit function is used.

Acknowledgment

This work was funded by the Visualization programme of the Swedish Foundation for Strategic Research, the KK Foundation, Vinnova, Invest in Sweden Agency, Vårdalstiftelsen (grant 2009/0079) and the Swedish Childhood Cancer Foundation (grant no. MT2013-0036). The authors are indebted to Petter Dyverfeldt for his help in the medical images acquisition and selection.

References

[Boo89] BOOKSTEIN F.: Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11* (1989), 567–585. 3

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 67–76. 3

[CHM03] CHEN H., HESSER J., MANNER R.: Raycasting free-form deformed-volume objects. *The Journal of Visualization and Computer Animation 14* (2003), 61–72. 2, 6

[CSC10] CORREA C. D., SILVER D., CHEN M.: Constrained illustrative volume deformation. *Computer & Graphics 34* (2010), 370–377. 2

[CSW*03] CHEN M., SILVER D., WINTER A. S., SINGH V., CORNEA N.: Spatial transfer functions - a unified approach to specifying deformation in volume modeling and animation. Proceedings of volumegraphics'03, pp. 35–44. 2

[Cud10] Nvidia cuda c programming guide, version 3.1, 2010. 1, 2

[FRS01] FORNEFETT M., ROHR K., STIEHL H.: Radial basis functions with compact support for elastic registration of medical images. *Image and Vision Computing 19* (2001), 87–96. 6

[FSRR00] FANG S., SRINIVASAN R., RAGHAVAN R., RICHTSMEIER J. T.: Volume morphing and rendering - an integrated approach. *Comput. Aided Geom. Des. 17* (January 2000), 59–81. 2, 6

[Gib97] GIBSON S. F.: 3d chainmail:a fast algorithm for deforming volumetric objects. Proceedings of the 1997 symposium on interactive 3D graphics. 2

[HLSR09] HADWIGER M., LJUNG P., SALAMA C. R., ROPINSKI T.: Eurographics 2009 course notes: Gpu-based volume raycasting with advanced illumination. In *Eurographics 2009 course* (2009), Eurographics 2009. 2

[KW03] KRÜGER J., WESTERMANN R.: Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings IEEE Visualization 2003* (2003). 2

[KY97] KURZION Y., YAGEL R.: Interactive space deformation with hardware-assisted rendering. *IEEE Computer Graphics and Applications 17*, 5 (1997), 66–77. 2

[LDS04] LEVIN D., DEY D., SLOMKA P. J.: Acceleration of 3d, nonlinear warping using standard video graphics hardware: implementation and initial validation. *Computerized Medical Imaging and Graphics 28*, 8 (2004), 471–83. 2

[LMK03] LI W., MUELLER K., KAUFMAN A.: Empty space skipping and occlusion clipping for texture-based volume rendering. In *Proc. IEEE Visualization 2003* (2003), pp. 317–324. 2

[LP00] LAPEER R., PRAGER R.: 3d shape recovery of a newborn skull using thin-plate splines. *Computerized Medical Imaging and Graphics 24* (2000), 193–204. 3

[LSR10] LAPEER R., SHAH S., R.S. R.: An optimised radial basis function algorithm for fast non-rigid registration of medical images. *Computers in Biology and Medicine 40*, 1 (2010), 1–7. 2, 6

[ML94] MARSCHNER S. R., LOBB R. J.: An evaluation of reconstruction filters for volume rendering. IEEE Visualization'94. 5

[MRH10] MENSMANN J., ROPINSKI T., HINRICHS K. H.: An advanced volume raycasting technique using gpu stream processing. In *GRAPP: International Conference on Computer Graphics Theory and Applications* (Angers, 2010), INSTICC Press, pp. 190–198. 2

[NMK*05] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics, 2005. 2

[RGW*03] RÖTTGER S., GUTHE S., WEISKOPF D., ERTL T., STRASSER W.: Smart hardware-accelerated volume rendering. In *VisSym* (2003). 2

[Row07] ROWLAND R. S.: *Fast Registration of Medical Imaging Data Using Optimised Radial Basis Functions :PhD Dissertation*. PhD thesis, University of East Anglia, 2007. 2, 6

[San10] SANDERSON C.: Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments, 2010. 3

[Sig06] SIGG C.: Representation and rendering of implicit surfaces, 2006. 1

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13* (2004), 600–612. 6

[WRS01] WESTERMANN R., REZK-SALAMA C.: Real-time volume deformations. *Computer Graphics Forum 20*, 3 (2001). 2, 6

SIGRAD 2014
M. Obaid, D. Sjölie and M. Fjeld (Editors)

Mirror Stereoscopic Display for Direct Volume Rendering

F. M. M. Marreiros[1,2] and Ö. Smedby[1,2,3]

[1]Center for Medical Image Science and Visualization (CMIV), Linköping University, Sweden
[2]Department of Science and Technology (ITN) - Media and Information Technology (MIT) , Linköping University, Sweden
[3]Department of Radiology (IMH), Linköping University, Sweden

Abstract

A new mirror stereoscopic display for Direct Volume Rendering (DVR) is presented. The stereoscopic display system is composed of one monitor and one acrylic first surface mirror. The mirror reflects one image for one of the eyes. The geometrical transformations to compute correctly the stereo pair is presented and is the core of this paper. System considerations such as mirror placement and implications are also discussed.
In contrast to other similar solutions, we do not use two monitors, but just one. Consequently one of the images needs to be skewed. Advantages of the system include absence of ghosting and of flickering.
We also developed the rendering engine for DVR of volumetric datasets mostly for medical imaging visualization. The skewing process in this case is integrated into the ray casting of DVR. Using geometrical transformations, we can compute precisely the directions of the rays, producing accurate stereo pairs.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture—Three-dimensional displays

1. Introduction

Stereopsis or binocular disparity is one of the most important binocular depth cues for close range visualization (personal space or action space) [CV95]. Using two images (one for each eye) at slightly different angles, it is possible to triangulate the distance to an object. To do so, our brain has to find corresponding points in both images (stereo matching) and then triangulate. In some cases, regions might be occluded in one view; for these regions other depth cues come into play.

We have developed a stereoscopic system that is, in principle, similar to other stereoscopic systems, i.e., we need to generate the stereo pair and display each image to the correct eye. The difference between stereoscopic technologies lies in how the images are presented to the viewer.

Here we will only consider systems with relatively large screens like typical monitors. Thus we exclude head-mounted displays and similar systems that make use of small displays.

The most common active stereo display technique is shutter glasses, where the display is synchronized with the opening and closing of the shutters of the glasses, so that the viewer views just one image at a time [Nvi12], [Bar03]. The drawback is the synchronization process that might not work perfectly, producing ghost artefacts (crosstalk), and the risk of prolonged usage triggering epileptic seizures due to flickering. Other passive systems include:

- anaglyphs, that use glasses with colour filters to filter the images, limiting the usage of colour. Ghosting can occur as the colour filters are not perfect;
- polarized, where special screens are used to display images with different light polarizations and special glasses are required. They suffer from ghosting due to light polarization crosstalk;
- wavelength multiplex imaging [JF03], where light is separated into three spectral ranges and special glasses are used with filters for specific spectral ranges. It also suffers from ghosting due to spectral range overlapping.

There are also autostereoscopic displays from several brands in the market. Two examples of this are parallax barrier and lenticular autostereoscopic displays [Ber96], [Ive02]. In fact, the principles of this technology are rather old, but were made available by the new LCD screen technology. For a more complete survey of stereo display systems, see [Hol05], [KH07].

1.1. Related work

Dual monitor systems [KH07], [FRM*05] can also be used for stereoscopic viewing. The Planar StereoMirror 3D Monitor is an example that uses a semi-transparent polarized mirror [Pla]. Most similar to our system is a dual-monitor single-mirror system originally proposed by Hart [WC10], [Boh] where the viewer has to place his eyes at a certain angle between the edge of the mirror. The major benefit of this system is that there are no ghosting artefacts. Our system is based on the same concept, but a major difference is that we only use one monitor. To note that one single monitor stereoscopic system is also presented in: http://stereo.jpn.org/eng/stphmkr/mirror/mirrorview.htm. Although this system may seems equal to ours, in fact it is not because it does not compute the correct geometrical stereo pairs. Instead, it uses an approximation that can produce a reasonable stereo sensation because our visual system can compensate to a large extent positional errors.

1.2. Motivation

The motivation was to study alternative configurations to the dual monitor system (fixed configuration - same angles between components of the system) and still maintain its benefits. In particular we were interested in the possibility to simplify the system (by using only one monitor) and to understand the general implications of such new configurations on the system parameters in order to produce accurate stereo-pairs.

In the next sections, we will present the steps needed for the construction of the system and system considerations.

2. Stereo system design

The stereoscopic display system we have developed is composed of one monitor and one acrylic first surface mirror (First Surface Mirror, Toledo, Ohio, USA, http://www.firstsurfacemirror.com). This special mirror needs to be used because traditional mirrors have two reflections, one in the front layer and another in the back layer. If a conventional mirror is used, ghosting effects will be visible due to the double reflection. According to manufacturer specifications, the mirror has a reflectance level of 94-96%. This affects slightly the brightness of the colours, but fortunately the changes are small enough that our visual system can easily compensate for these variations.

As in most stereoscopic systems, two images are presented to the user in the same viewing window (display), one for each eye. For one of the eyes - in this case the left eye - this is trivial, but for the right eye, the aid of the mirror is required. Before placing the mirror, a choice must be made regarding the center of the viewing region and the viewing direction. In this system, the view direction is perpendicular to the view window and the center at the physical center

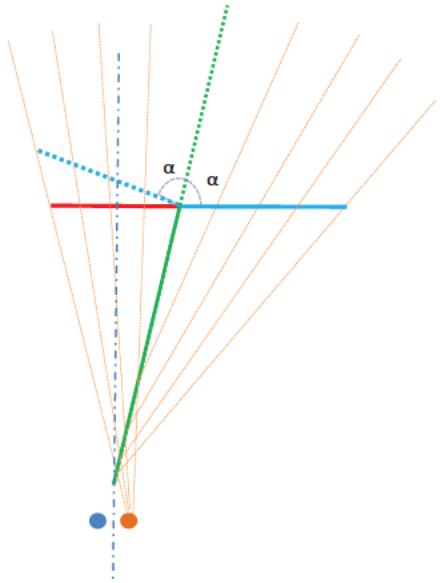


Figure 1: *Top view schematic presenting the left (orange) and right (blue) eye positions; the left (red), right (cyan) and reflected right (dashed cyan) viewing windows (the center of the left view window is presented in dotted-dashed blue); the reflected rays and their straight equivalent (dotted orange); the mirror position (green) and the angle of reflection.*

of the window. The configuration is different than in [Boh], where the view direction is angled. This configuration constrains the position of the mirror: The front edge (closer to display) is placed on the right edge of the view window and the back edge placed parallel to the vertical line at the center of the view window, Fig. 1 and 2. Note also the position of the eyes relative to the mirror: they have to be centered (around the back edge of the mirror) and at a small distance from it (80 mm). This small distance allows the face, in particular the nose, not to be in direct contact with the mirror, thus avoiding an uncomfortable position for the viewer. In this work, a fixed interpupillary distance of 65 mm was used. This value is slightly above the mean adult male of interpupillar distance 64.7 mm reported in [Dod04], [GCC*89].

The next step is to determine the directions of the reflected rays, using the law of specular reflection [Hea81]. The law states that the direction of incoming light (the incident ray), and the direction of outgoing light reflected (the reflected ray) make the same angle with respect to the surface normal and that the incident, normal, and reflected directions are coplanar.

To exemplify, we cast some rays from the right eye position and follow their path, as presented in Fig. 1. Here we can clearly see that we require an extra viewing window for

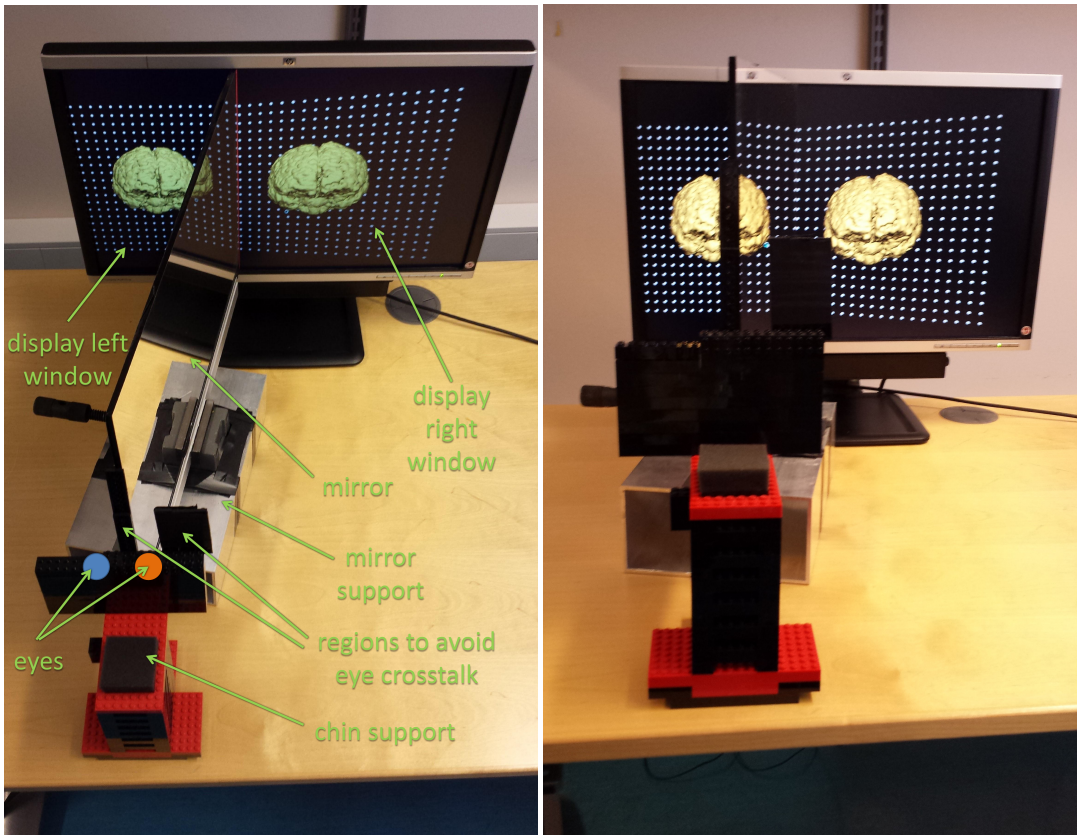


Figure 2: *Photographs of the stereo system prototype. On the left are included annotations of the system components and the eyes positions.*

the right eye. In practice, we have two viewing windows occupying two portions of the display (Fig. 2). The right window needs to have a larger length, due to the image skewing caused by the angle of the mirror. The reflection of the right viewing window produces a reflected window. To calculate it, the four vertices of the right window are reflected to obtain their 3D location. To do so, we use a reflection matrix introduced by Bimber et al. [BES00], [BFSE01]. If the mirror's plane is: $f(x,y,z) = ax+by+cz+d = 0$. The reflected points can be calculated by multiplying the reflection matrix with the original point: $\vec{p}' = M \cdot \vec{p}$.

$$M = \begin{bmatrix} 1-2a^2 & -2ab & -2ac & -2ad \\ -2ab & 1-2b^2 & -2bc & -2bd \\ -2ac & -2bc & 1-2c^2 & -2cd \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Using the reflected right viewing window, we avoid calculating reflected rays (the equivalent straight rays are used) and construct two different scenes with the same virtual objects.

Once we know the position of both viewing windows (left and reflected right) and the eye positions, we can place objects in the scene and correctly render them. For the rendering, we use a ray-casting algorithm that casts rays from the eye position passing by the center of each individual pixel into the scene we want to render.

Before performing the rendering, the stereo region must be checked. For optimal results, the virtual objects should be confined to the stereo region. This region is defined by the intersection of the left and right view frustums (Fig. 3).

Regarding the viewing transformation, there is only one last transformation to perform: a horizontal image flip due to the mirror reflection. This inversion can be seen in Fig. 1 by looking at e.g. the leftmost ray that is not reflected which corresponds to the rightmost ray reflected.

3. System stereo pair rendering

This system was designed to render Direct Volume Rendering (DVR) [Lev88], [DCH88] stereo pairs. To generate DVR images, there are several possible algorithms. As mentioned previously, a DVR ray-casting algorithm was chosen because it follows precisely the geometry designed for our

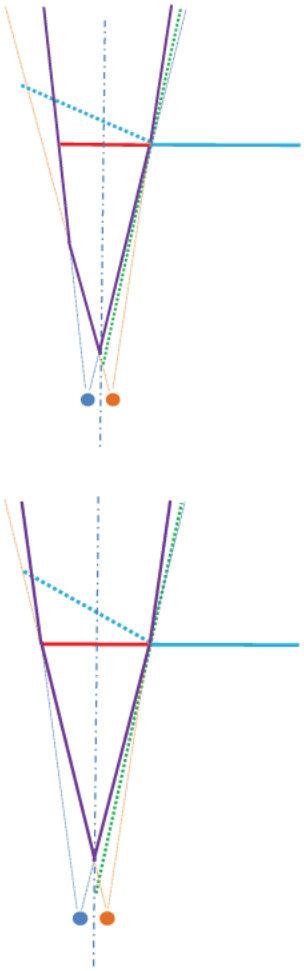


Figure 3: *Top view schematic presenting a non-symmetric (Top) and symmetric (Bottom) stereo region (inside the violet lines). The horizontal display size and mirror horizontal size are the same in both cases.*

system. Ray-casting works, as the name, indicates by casting rays from the observer position towards the scene. The direction of the ray is provided by the viewer position and the center of each pixel on the viewing window. The data used are regular grids with scalar values (voxels). These are typically acquired by medical image modalities such as Computed Tomography (CT) or Magnetic Resonance Imaging(MRI). The rays are sampled at fixed intervals and interpolated to obtain a scalar value (intensity). A transfer function relates intensity to opacity and color, thus calculating their contribution at each sample. Each ray, accumulates color and opacity along it until the opacity level reaches the maximum (opaque). In the case of surface rendering, the opacity for the voxels in the range of visible intensity values is always at the maximum, i.e. the rays will terminate when the object is first hit. Different illumination models and rendering styles can also be applied. Once the ray is terminated, the color values of the pixels that belong to the each ray are saved to produce the final images.

Illustrations of the casted rays for the right eye are presented in Fig. 1. The rays pass by the center of the pixels of the virtual window (reflected window). To obtain these pixel locations, the 3D positions of the edges of the virtual window are computed with the reflection matrix and the pixel positions sampled along this window. As previously mentioned, an image flip is then performed to render the image appropriately in the right region of the display.

4. System considerations

The construction of the system imposes certain limitations on the viewing angles. We limit these angles by introducing rectangular regions in front of both eyes through which the user will look. This is introduced for two reasons: first, we want to avoid crosstalk, since the eyes are placed at a small distance from the mirror, and second, with the right eye we only want to see the reflected image and not the image presented in the display.

According to [Hen93], the human horizontal field of view is approximately 200 degrees, but the stereo region is approximately 120 degrees. We should point out that the effective stereo region can be smaller, since the nasal angle is smaller than the temporal angle, i.e. some rays are obscured by the nose region.

In our case, this is not a problem, as we limit the field of views to the viewing windows, i.e. the rectangular regions limiting the field of view only need to allow the passage of the rays that pass by the center of each pixel (both viewing windows).

The stereo regions using the design presented in the previous section might be non-symmetric as presented in Fig. 3. To enforce the symmetry of the stereo region, constraints must be introduced. For the stereo region to be symmetric, the left-most ray of the right eye must cross simultaneously the left edges of the left and reflected right viewing windows. For fixed mirror size, display size, interpupillary distance and nose-to-mirror distance, the only parameter that can be optimized is the left viewing window size (by consequence the position and size of the other windows are also changed). The possible values of the left viewing window size are in the interval from 0 to half the horizontal size of the display. In order to sample pixel-by-pixel, the horizontal resolution must be provided. To solve this problem, a Matlab script was developed (available upon request) that computes the optimal horizontal size of the left viewing window. This value is obtained when the distance from the left edge of the reflected window to the left-most ray of the right eye (that passes by the left edge of the left window) is minimized.

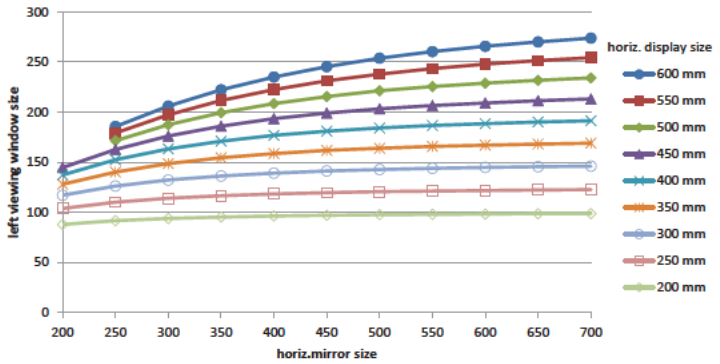


Figure 4: *Left viewing window size values by changing the mirror horizontal size and the horizontal display size.*

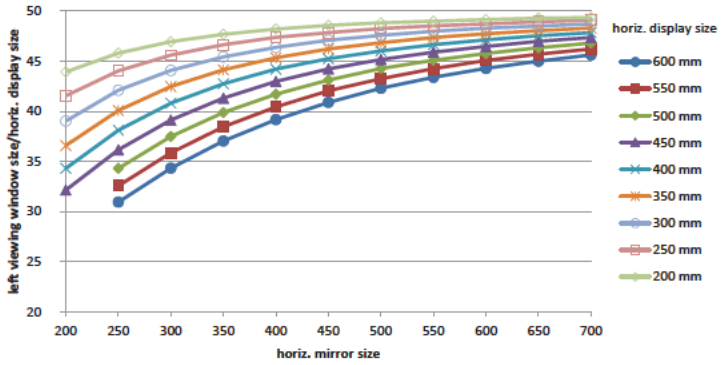


Figure 5: *Ratios of left viewing window size to horizontal display size by changing the mirror horizontal size and the horizontal display size.*

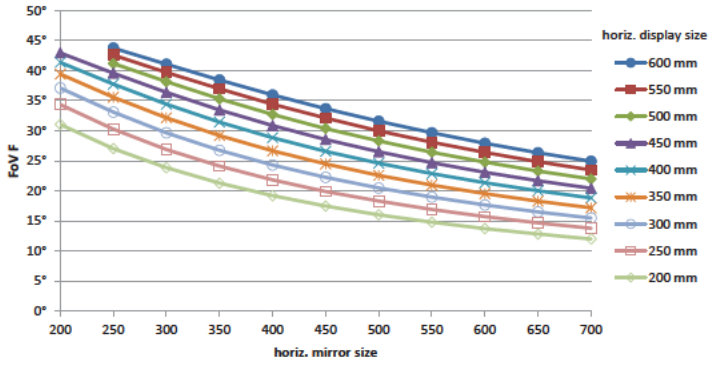


Figure 6: *FoV F (Field of View of the region in front of the viewing window) values by changing the mirror horizontal size and the horizontal display size.*

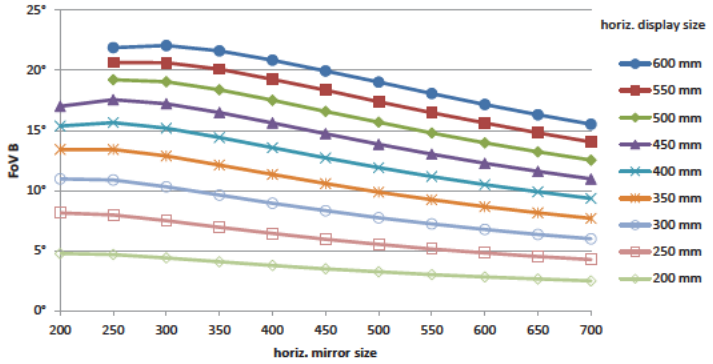


Figure 7: *FoV B (Field of View of the region behind the viewing window) values by changing the mirror horizontal size and the horizontal display size.*

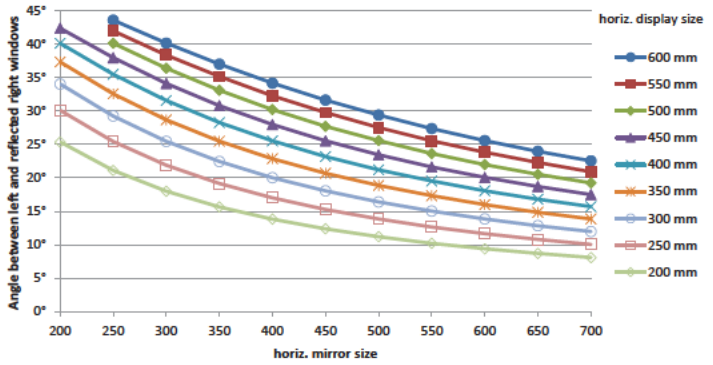


Figure 8: *Angles between left and reflected right viewing windows by changing the mirror horizontal size and the horizontal display size.*

To further study the system (with symmetry correction), the mirror size and display size were changed to explore how these affect the remaining system parameters. The results are presented in the graphs and examined in the next section. For the range of values studied, there are configurations that are considered invalid. These are the ones where the left-most ray of the right eye does not intercept the mirror or the right-most ray of the left eye also does not intercept the mirror. In the graphs these cases will make some lines start at higher values.

5. Results

The results show that the horizontal left viewing window size increases with the horizontal display and mirror horizontal sizes (Fig. 4). The ratio of the horizontal left viewing window size to horizontal display size is presented in Fig. 5. It shows that for smaller displays, the ratio is higher i.e. a more effective usage of the display is achieved. For larger mirrors, the horizontal left viewing area will approximate half the horizontal display size. In Fig. 6 and Fig. 7 are presented the angles of the two horizontal Fields of View (FoV) of the system (in front - FoV F and behind - FoV B the viewing window, as presented for the symmetric case in Fig. 3). For FoV F, the angles increase with horizontal display size and decrease with horizontal mirror size. For FoV B, the relation is more complex: the angles increase with horizontal display size, but there seems to be a maximum of the horizontal mirror size per horizontal display size. Finally, the angles between the horizontal left and reflected right windows increase with horizontal display size and decrease with horizontal mirror size (Fig. 8).

6. Discussion

The results show that most parameters are optimized by using larger mirrors with the exception of the FoV. Although the FoV might be smaller, the stereo region may in fact be larger, because a more effective use of the viewing windows is achieved. Furthermore, the angles between left and re-

flected right windows are minimized for larger mirrors. For these reasons we recommend the use of mirrors with at least the same horizontal size as the display. In practice, the user will have to balance the system parameters in order for the stereo area to enclose all the virtual objects. With the script we have developed, the user can test different mirror and display configurations to select the best option. The angles between left and reflected right windows might have an impact on other depth cues such as convergence and accommodation. To minimize this problem, configurations with lower angles should be used. These depth cues have a rather low strength in comparison to binocular disparity [CV95]. In fact, just by using smaller display sizes with the same mirror to change the angles we can still have a very strong depth sensation with large angles.

A limitation of our system is that it is only developed for DVR. If rectangular stereo images are available, they cannot be directly used because one of the images needs to be skewed (Fig. 2). For our main purpose, visualization of stereo medical volumes, our system fulfils most of the relevant criteria and can compete with traditional systems.

In the medical area, some evaluations of stereoscopic systems were performed. Some even go back as far as 1990 [OO90], and a more recent performed by radiologists comparing different 3D systems can be found in [TSO*12]. Several works in the visualization literature suggest that by using stereopsis, we increase greatly the understanding of depth [HWSB99], [HHM98], in particular for medical datasets [KSTE06]. It should, however, be pointed out that the usage of stereopsis is very task-dependent [WT05].

If collaboration is required two scenarios are possible, the first is to duplicate the system and render the same stereo pairs (simply by duplicating the screens) and second by allowing one user at a time to view the scene. The last scenario is similar to microscope visualization, traditionally found in biology, where different users view at a time an image using a microscope. An alternative scenario is also possible, where one user views the stereoscopic view and the remaing a monoscopic view.

Stereoscopic display systems play an important role in the study of depth perception. In such perception studies it is important to reduce visual fatigue due to flickering effects and produce sharp stereo pairs (with no significant color changes and no ghosting). By eliminating these artifacts, possible confounding factors are discarded. To solve this problem, we selected one system that does not suffer from these artifacts (dual monitor system) and simplified it. By doing so the costs of the system are reduced and a smaller physical space is required in the room. Also in the dual-monitor systems the user has to place his eyes at a certain angle, although it is possible to compute this angle in practice it is rather hard to place the user in the right position. In our system the user looks straight at the display avoiding this problem.

This system has been used to study depth perception of enclosed objects [MS13]. In this case, it is important that the objects in the scene are perceived in their correct spatial position. The results of our previous study demonstrate that with appropriate DVR rendering methods, the users are rather accurate in this task, enabling the system to be used for medical purposes, e.g. to view the location of a tumour inside the brain. Small errors in eye position (due to head movements or different interpupillary distance between subjects), mirror or display positions might occur, but within a small range, the visual system is rather good to compensate for them. In fact, by simply changing the angle of the mirror we can still have a strong stereo sensation with large incorrect angles, but this will affect the correct spatial position of the objects and will make it more difficult to perceive stereo until the effect completely breaks down. The results of this previous study were consistent across participants with high scores of accuracy in perceiving objects spatial position; thus the system is well designed for depth peception even in the presence of small eye position errors. If this would not be the case, our previous study would not have produced consistent results.

The previously reported single monitor system: http://stereo.jpn.org/eng/stphmkr/mirror/mirrorview.htm was not designed for DVR and only provides an approximation of the correct geometry; thus it cannot be used in cases where the spatial position of objects is critical.

We use our stereo system mainly with wide screen monitors, but smaller screens like popular pad systems can be considered. Here, the limitation is the human focal length, approximately 20−24 mm [Gra68], [Sac04].

We have designed the stereoscopic system mainly for stereoscopic visualization of DVR images, but it is possible to make it general purpose if we consider rendering of polygons using off-axis projection that requires a non symmetric camera frustum, by OpenGL in a similar fashion as presented in [Bou99], [MM05].

In Fig. 2 we present the real prototype of the stereo system. In the display we can see the stereo pair in this case of a segmented brain from a MRI image of the head with a plane of dots in the back. We would also like to mention that the system is easy to assemble and that the additional cost is low (mainly the price of the mirror).

Prolonged usage of the system is problematic due to the fact that the head position is fixed. To reduce this problem, a base to place the chin was constructed, but this is by no means a recommendable ergonomic position. For this reason, the system should be only used for a short period of time.

7. Conclusion

We have presented a new stereoscopic system for visualization of DVR images. The main difference in comparison to

the most similar system [Boh] is the use of only one monitor. We also developed the rendering engine for DVR of volumetric datasets mostly for medical imaging visualization. In spite of certain geometrical constraints, the system may prove useful for medical imaging applications.

Acknowledgment

This work was funded by the Visualization programme of the Swedish Foundation for Strategic Research, the KK Foundation, Vinnova, Invest in Sweden Agency, Vårdalstiftelsen (grant 2009/0079) and the Swedish Childhood Cancer Foundation (grant no. MT2013-0036). The authors are indebted to Torbjörn Gustafsson and Per Carleberg for their contribution in the display construction.

References

[Bar03] BARCO: Stereoscopic projection: 3d projection technology. http://www.barco.com/projection_systems/downloads/barco_stereoscopic_proj.pdf, 2003. 1

[Ber96] BERTHIER A.: Images stéréoscopiques de grand format (in French). *Cosmos 34*, 590 and 591 (May 1896), 205–210 and 227–233. 1

[BES00] BIMBER O., ENCARNAÇÃO L. M., SCHMALSTIEG D.: Augmented reality with back-projection systems using transflective surfaces. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2000) 19*, 3 (2000), 161–168. 3

[BFSE01] BIMBER O., FRÖHLICH B., SCHMALSTIEG D., ENCARNAÇÃO L. M.: The virtual showcase. *IEEE Computer Graphics and Applications 21*, 6 (2001), 48–55. 3

[Boh] BOHÁČ M.: Dual monitor set−up for stereoscopic viewing. http://klub.stereofotograf.eu/dual_monitor.php. 2, 8

[Bou99] BOURKE P.: Calculating stereo pairs. http://paulbourke.net/miscellaneous/stereographics/stereorender/, 1999. 7

[CV95] CUTTING J. E., VISHTON P. M.: Perceiving layout and knowing distances: The integration, relative potency, and contextual use of different information about depth. "W. Epstein and S. Rogers (eds.), Handbook of perception and cognition", 1995. 1, 7

[DCH88] DREBIN R. A., CARPENTER L., HANRAHAN P.: Volume rendering. *SIGGRAPH Computer Graphics 22*, 4 (1988), 65–74. 3

[Dod04] DODGSON N. A.: Variation and extrema of human interpupillary distance. In *Proc. of SPIE: Stereoscopic Displays and Virtual Reality Systems XI* (2004), vol. 5291, pp. 36–46. 2

[FRM*05] FERGASON J., ROBINSON S., MCLAUGHLIN C., BROWN B., ABILEAH A., BAKER T., GREEN P.: An innovative beamsplitter-based stereoscopic/3d display design. *SPIE Stereoscopic Displays and Virtual Reality Systems 5664* (May 2005), 488–494. 2

[GCC*89] GORDON C. C., CLAUSER B. B. C. E., CHURCHILL T., MCCONVILLE J. T., TEBBETTS I., WALKER R. A.: 1987−1988 anthropometric survey of u.s. army personnel: Methods and summary statistics. tr−89−044. natick ma: U.s. army natick research, development and engineering center., 1989. 2

[Gra68] GRAND Y. L.: *Light, Color and Vision*, second ed. London: Chapman and Hall, 1968. 7

[Hea81] HEATH S. T. L.: *A history of Greek mathematics. Volume II:From Aristarchus to Diophantus*. Oxford: At The Clarendon Press., 1981. 2

[Hen93] HENSON D.: *Visual Fields*. Oxford: Oxford University Press, 1993. 4

[HHM98] HUBBOLD R., HANCOCK D., MOORE C.: Stereoscopic volume rendering. *Proc. Visualization in Scientific Computing '98 6*, 3 (1998), 105–115. 7

[Hol05] HOLLIMAN N.: 3d display systems. http://www.dur.ac.uk/n.s.holliman/Presentations/3dv3-0.pdf, 2005. 1

[HWSB99] HUBONA G., WHEELER P., SHIRAH G., BRANDT M.: The relative contributions of stereo, lighting and background scenes in promoting 3d depth visualization. *ACM Transaction on Computer-Human Interaction 6*, 3 (1999), 214–242. 7

[Ive02] IVES F. E.: A novel stereogram. *Journal of the Franklin Institute 153* (1902), 51–52. 1

[JF03] JORKE H., FRITZ M.: Infitec−A new stereoscopic visualization tool by wavelength multiplexing imaging. In *Proc. Electronic Displays* (2003). 1

[KH07] KONRAD J., HALLE M.: 3-d displays and signal processing. *IEEE Signal Processing Mag. 24*, 7 (May 2007), 97–111. 1, 2

[KSTE06] KERSTEN M., STEWART J., TROJE N., ELLIS R.: Enhancing depth perception in translucent volumes. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1117–1124. 7

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 3 (May 1988), 29–37. 3

[MM05] MARREIROS F. M. M., MARCOS A.: Calculating the stereo pairs of a mirror-based augmented reality system. *13º Encontro Português de Computação Gráfica* (2005). 7

[MS13] MARREIROS F. M. M., SMEDBY Ö.: Stereoscopic static depth perception of enclosed 3d objects. In *SAP '13 Proceedings of the ACM Symposium on Applied Perception* (2013), pp. 15–22. 7

[Nvi12] NVIDIA: 3D Vision. http://www.nvidia.com/object/3d-vision-main.html, 2012. 1

[OO90] OWCZARCZYK J., OWCZARCZYK B.: Evaluation of true 3d display systems for visualizing medical volume data. *The Visual Computer 6*, 4 (1990), 219–226. 7

[Pla] PLANAR3D: 3D Technologies. http://www.planar3d.com/3d-technology/3d-technologies/. 2

[Sac04] SACKETT C.: Survey of optical systems: Phys 531, lecture 11, university of virginia. http://galileo.phys.virginia.edu/classes/531.cas8m.fall04/l11.pdf, 2004. 7

[TSO*12] TOURANCHEAU S., SJÖSTRÖM M., OLSSON R., PERSSON A., RUDLING J., ERICSON T., NORÉN B.: Subjective evaluation of user experience in interactive 3d visualization in a medical context. *Proc. SPIE 8318*, 831814 (2012), 219–226. 7

[WC10] WU H.-H. P., CHANG S.-H.: Design of stereoscopic viewing system based on a compact mirror and dual monitor. *SPIE Optical Engineering 027401 49*, 2 (2010), 1–6. 2

[WT05] WICKENS C., THOMAS L.: Effects of CDTI display dimensionality and conflict geometry on conflict resolution performance. *In Proceedings of the 13th International Symposium on Aviation Psychology* (2005). 7

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

PREDICTION OF PHYSICS SIMULATIONS FOR GRAPHICS AND ANIMATION

R. Dupre, V. Argyriou, D. Greenhill

Kingston University

Abstract

In this paper a novel approach for physics simulation prediction is proposed with applications in graphics rendering and multimedia. A prediction mechanism is introduced based on regression that aims to reduce the computational cost of simulations in a given scene by negating the need to perform physics calculations every frame. Novel features based on the energy of a scene over time are suggested for the training stage. Experiments were performed to evaluate the performance of the proposed prediction system, indicating that in cases where precision is not essential, regression tools can be utilized providing visually similar kinematics.

Physics, Performance, Correlation and regression analysis, Graphics rendering

Categories and Subject Descriptors (according to ACM CCS): - [Real-time rendering]: Visual Analytics—-

1. Introduction

Development of physics engines in computer games has had major advances in recent years. Also, the increased use of physics engines in other industries such as robotics and engineering, medicine, and mathematics [LLS09, HQZ12, SLM06] has led to the development of faster and more precise engines. In tandem the increase in computer performance has allowed and facilitated the development of such engines. Physics engines tend to focus on two main areas when it comes to simulating the physical world. The first is concerned with high precision, and aims to replicate the physical world as accurately as possible. The second is focused on the speed in which it can calculate these physical representations. An evaluation of free publicly available physics engines has been done [BB07], which highlighted that of the engines tested none were better at all experiments conducted and nearly all were better in one test than another.

Due to increasing physical complexity of scenes and the need for real-time rendering in combination with limitations of computer hardware it is important to find a balance between accurate representation and calculation time based on a set of parameters, such as distance and type of simulation.

In this paper we will demonstrate a novel dynamic approach in which the computation required to mimic physical aspects of a 3D environment can be reduced through the use of prediction techniques based on machine learning and Support Vector Regression. The proposed framework could learn the physic simulations during a game and dynamically switch to a prediction mode for a certain amount of time to reduce the overall complexity and computational workload. An analysis of the area will be followed by an overview of previous work. An outline of the basic concepts related to physics simulations and then the proposed methodology and results will be presented, leading to the final conclusions of this work.

2. Previous Work

Physics simulation plays an important role in many fields such as graphics rendering, multimedia, engineering, science, and education, allowing us to understand the laws of motion, matter, space and time. The ability to simulate physical behavior is critical in order to understand the complex physical world. This aids in improving design and realism in various industries such as multimedia and game applications, special effects and real-time rendering. Examples of industries that utilise visual simulation techniques where speed is a consideration include game production, and real time simulation environments such as flight simulators. In [Gou06] the fundamentals and basic methodologies for physics simulation can be found. A dynamic simulation approach for rigid bodies was proposed by Baraff in [Bar93] and in [Ega03] implementation techniques for real time rigid body simulation were suggested. Physics modeling for com-

puter games development and other multimedia applications was also analysed in [BM11, DMI11, SM12, RSH*13].

2.1. Regression and Support Vector Machines

Regression is the statistical process of analysing data sets to discover a relationship amongst its variables. Often used in the areas of forecasting and data analysis [WHP10], it provides us the ability to define and explore relationships between dependent and independent variables. Support Vector machines, originally designed as a classification algorithm, and the later devised Support Vector Machines for regression allows the presentation of a solution based on a small subset of training data.

3. Proposed Methodology for Physics Simulation Prediction

In this section, the problem of predicting physics mechanics in a given scene is re-formulated as a regression problem. Considering the advantages of Support Vector Regression (SVR), it was adopted as a regression tool in this work. Regression tries to estimate the relationship between a dependent variable (location or acceleration) and independent variables (a selected initial energy or force under a selected direction). This analysis allows the prediction of the physics dynamics in near future given the initial conditions for a short amount of time.

3.1. Regression problem formulation

SVR as a regression method was chosen for the proposed approach, because it features good generalization performance [SL06] that is essential for regression applications in combination with a polynomial kernel. SVR does not have a local minimum problem compared to other regression techniques such as neural networks. Additionally, SVR is not limited to the input space by using a linear kernel, but instead is operated in a arbitrary large feature space, since it is a kernel-based regression technique.

Let us assume that the training data is given as $\{(x_1,y_1),\ldots,(x_k,y_k)\} \subset X\times$, where k is the number of samples and X denotes the space of the input patterns (e.g. $X = R^d$), which might be a vector indicating the direction and magnitude of an applied force or equivalently an obtained acceleration and position). In the case of linear functions the regression equation $f(x)$ is defined as:

$$f(x) =< w,x > +b \tag{1}$$

where $w \in X, b \in R$, and $< \cdot,\cdot >$ denotes the dot product in X. The goal is to estimate a function $f(x)$ that satisfies the error from the difference between observed target y_i and the predicted value $f(x_i)$, with $i = 1\ldots k$, is disregarded as long as it is less than e. While at the same time it is as flat as possible by minimizing the norm $\|w\|^2 < w,w >$. Also, in order to avoid non feasible convex optimization problems some error is allowed analogously to the 'soft margin' loss function. Furthermore, slack variables ξ, ξ^* are introduced to cope with data points that lie outside the absolute e regions, and the following formulation is obtained.

$$minimise\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{k}(\xi_i + \xi_i^*) \tag{2}$$

$$subject\ to \begin{cases} y_i - < w,x_i > -b \le e + \xi_i \\ < w,x_i > +b - y_i \le e + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \tag{3}$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than e are tolerated.

A Lagrange function is constructed from the objective function and the corresponding constraints, by introducing a dual set of variables. In order to solve more easily this optimization task the obtained dual optimization problem is defined as:

$$maximise\ to \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{k}(a_i - a_i^*)(a_j - a_j^*) < x_i,x_j > \\ -e\sum_{i=1}^{k}(a_i - a_i^*) + \sum_{i=1}^{k} y_i(a_i - a_i^*) \end{cases} \tag{4}$$

$$subject\ to \sum_{i=1}^{k}(a_i - a_i^*) = 0, a_i, a_i^* \in [0,C] \tag{5}$$

where $a_i, a_i^* \ge 0$ are the Lagrange multipliers. So, the objective function is obtained from the Support Vector expansion and can be written as follows:

$$f(x) = \sum_{i=1}^{k}(a_i - a_i^*) < x_i,x > +b \tag{6}$$

where the w can be described as a linear combination of the training pattern x as:

$$w = \sum_{i=1}^{k}(a_i - a_i^*)x_i \tag{7}$$

In order to apply a non-linear kernel to SVR in equations 5 and 6, the dot products $< x_i,x_j >$ for the linear kernel is replaced with a polynomial kernel defined as:

$$k^{pol}(x_i,x_j) = \beta_0 + \beta_1(x_i - x_j) + \ldots + \beta_p(x_i - x_j)^p + \varepsilon_i \tag{8}$$

The order of the polynomial was selected experimentally.

3.2. Feature selection and simulation

Based on the proposed methodology, a signal representation of a physical concept, such as acceleration, velocity or position over time, could be predicted using a few initial measurements and a set of training data. For example, instead of performing the complex physics simulations for an event's entire duration, only a few initial calculations over a very

short amount of time could be applied and the remainder of the simulation predicted using regression. This presents a huge computational saving at the cost of a minor decrease in visual accuracy. An example in a game environment; if the object or event that is affected is far away from the camera or is defined as being unimportant to accurately represent (i.e. not effecting the game play), prediction could provide a more efficient solution whilst maintaining the visualisation.

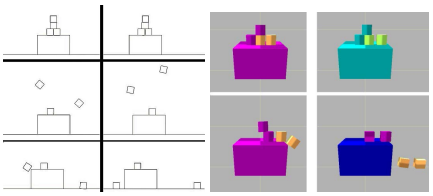


Figure 1: *An example of (left) 2D and (right) 3D simulated and predicted movements.*

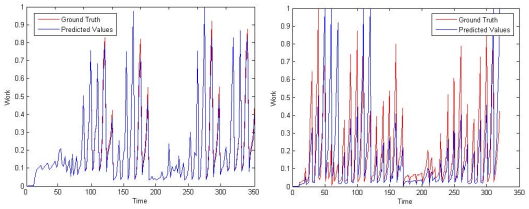


Figure 2: *Work over time for a (left) 2D and (right) 3D scene.*

Regarding the training dataset; regression is used to fully define a selected feature, in this case the Verlet Integrator acceleration, of an object in a scene over its duration. In order to obtain the position, we regard the velocity as the derivative of the position and the acceleration is the derivative of the velocity. This is repeated for a number of scenes each with forces of a variety of directions and magnitudes producing a model. For the prediction, a initial number of accelerations are simulated for each object in a scene. These initial values are then matched to one in the model using SVR and the polynomial kernel.

4. Results

In order to evaluate the performance of the proposed approach, experiments were performed using three physics engines. Box2D which performs constrained rigid body simulations, the Bullet open source physics engine featuring 3D collision detection, soft body dynamics, and rigid body dynamics; and the final, the NVIDIA PhysX engine which supports a number of game related features such as rigid and soft body dynamics as well as volumetric fluid simulation.

Both for the Box2D and Bullet engines, six scenes were designed and on each object an impulse force was applied. In the Box2D simulations, 36 force directions (sampling a circle every 10 degrees) and 10 different magnitudes were used to obtain the model. The same approach was used in the Bullet simulation, here the sampling was performed around a sphere. During the prediction stage the first 10 samples (locations of the scene objects) were calculated (less than 10%) and the remaining predicted. In figure 1 examples scenes are shown. In figure 2 the predicted and the calculated energy are plotted over time for two scenes demonstrating that predicted values closely follow those of the simulated. All scene results are shown in table 1 with the error varying from 1% to 15% based on the scene.

	Case1	Case2	Case3	Case4	Case5	Case6
2D	0.009	0.005	0.101	0.033	0.143	0.005
3D	0.017	0.009	0.102	0.045	0.153	0.009

Table 1: *Average percentage error based on measured difference between predicted and simulated values for 2D and 3D scenes for all scenes.*

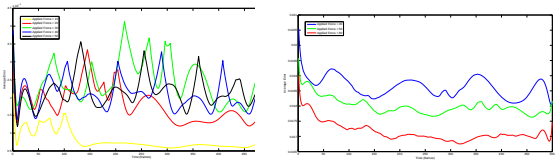


Figure 3: *The Sequential and Explosion Model's average error per frame between training data and the regression output (per Force).*

To further test this concept, two experiments were set up using the NVIDIA PhysX engine to visualise familiar scenarios in a multimedia environment. The first a simulated explosion, the second a waterfall. For each scenario, training data was collected and used as the basis for the prediction to reduce the computational complexity of these effects.

For the explosion scene, two models were created. In the first; samples of different forces were taken from around a sphere ('sequential' model). The second created training scenes in which 100 cubes (particles) have a force applied in various strengths and directions with each cube being recorded individually ('explosion' model). The accuracy of a scene is measured by the Euclidean distance between the predicted track of each particle and the original simulated track, normalised across all the particle data in the scenes.

Error	Sequential	Explosion	Both	Water
Force 1	0.013	0.011	0.010	0.023
Force 2	0.017	0.015	0.013	0.029
Force 3	0.022	0.024	0.017	0.036

Table 2: *Sequential, Explosion, Both and Water Models average error per force during testing.*

Figure 3 shows the average error that each model has over time against its original training track due to the regression. For testing; five explosions were simulated each with 3 forces for a 10 second period using 100 cubes (particles). The models were then used to predict each cubes track in 3D

space, using the initial 10% of simulated values to match a predefined trajectory. Matching was done by calculating the function curve that matched closest with the initial simulation samples during the training. Visualisations of the results are shown in figures 4 and 5. Table 2 demonstrates the accuracy of each model in relation to a simulated track, as well as the robustness of the method across differing scenes.

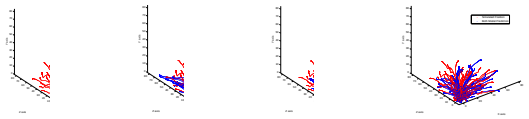


Figure 4: *Visualisations of a symmetric explosion (a) ground truth trajectories and the predicted ones for each model, (b) 'explosion', (c) 'sequential' and (d) both, (red is the ground truth and blue the estimated trajectories).*

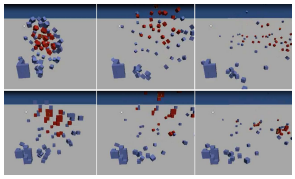


Figure 5: *Visualisations of an asymmetric explosion (top) ground truth trajectories and (bottom) the predicted ones for the 'sequential' model, (the red cubes indicate the applied forces).*

The same experiments were performed in the waterfall scene. A single model was created during the training comprising of five various waterfall scenes each with three different forces. Tests were carried out on 4 test waterfall scenes each with 3 forces. Examples of the outputted prediction verses the original simulation are scene in figure 6. The visualisations demonstrate that we can get a very close approximation of the original tracks though the use of the proposed methodology, with the overall average error shown in table 2. In our test environment we predict the movement of the particles in a scene for 90% of the time, this represents a considerable computational saving.

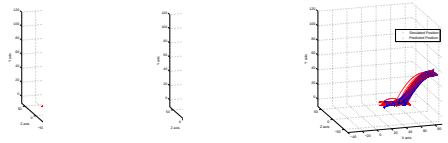


Figure 6: *Visualisations of real (red) and predicted (blue) trajectories for a test scene, three different forces.*

5. Conclusion

In this paper, a novel approach to predict simulated physics was proposed based on regression. Also, a proposed framework can operate dynamically and learn the prediction models during the game play. From the obtained results it can be observed that for cases that accuracy is not essential such as distant events or special effects that humans cannot distinguish the differences significant reduction in the overall required computational cost can be achieved.

References

[Bar93] BARAFF D.: Non-penetrating rigid body simulation. *State of the art reports*, May (1993). 1

[BB07] BOEING A., BRÄUNL T.: Evaluation of real-time physics simulation systems. *Proceedings of the 5th international conference on . . . 1*, 212 (2007), 281–288. 1

[BM11] BOGDAN P., MARCULESCU R.: A fractional calculus approach to modeling fractal dynamic games. *IEEE Conference on Decision and Control and European Control Conference* (Dec. 2011), 255–260. 2

[DMI11] DELGADO-MATA C., IBÂŤNEZ J.: Adaptive Physics for Game-Balancing in Video-Games for Social Interaction. *2011 International Conference on Technologies and Applications of Artificial Intelligence* (Nov. 2011), 254–259. 2

[Ega03] EGAN K.: Techniques for Real-Time Rigid Body Simulation. 1

[Gou06] GOULD, HARVEY; TOBOCHNIK, JAN; CHRISTIAN W.: *Introduction to Computer Simulation Methods: Application to Physical Systems*. Addison-Wesley, 2006. 1

[HQZ12] HU W., QU Z., ZHANG X.: A New Approach of Mechanics Simulation Based on Game Engine. *Computational Sciences and . . .* (2012). 1

[LLS09] LUO F., LIU C., SUN Z.: Intelligent Vehicle Simulation and Debugging Environment Based on Physics Engine. *2009 International Asia Conference on Informatics in Control, Automation and Robotics* (Feb. 2009), 329–333. 1

[RSH*13] ROENNAU A., SUTTER F., HEPPNER G., OBERLAENDER J., DILLMANN R.: Evaluation of physics engines for robotic simulations with a special focus on the dynamics of walking robots. *2013 16th International Conference on Advanced Robotics (ICAR)* (Nov. 2013), 1–7. 2

[SL06] SAKHANENKO N., LUGER G.: Shock physics data reconstruction using support vector regression. *. . . of Modern Physics C 17*, 9 (2006), 1313–1325. 2

[SLM06] SERVIN M., LACOURSIERE C., MELIN N.: Interactive simulation of elastic deformable materials. *Proceedings of SIGRAD Conference* (2006), 22–32. 1

[SM12] SILVA D. F., MACIEL A.: A comparative study of physics engines for modeling soft tissue deformation. *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)* (Oct. 2012), 1–7. 2

[WHP10] WU J., HUANG L., PAN X.: A novel bayesian additive regression trees ensemble model based on linear regression and nonlinear regression for torrential rain forecasting. *2010 Third International Joint Conference on Computational Science and Optimization* (2010), 466–470. 2

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

TopoLayout-DG: A Topological Feature-Based Framework for Visualizing Inside Behavior of Large Directed Graphs

Ragaad AlTarawneh, Max Langbein, Shah Rukh Humayoun, Hans Hagen

Computer Graphics and HCI Group
University of Kaiserslautern, Germany
{tarawneh, m_langbe, humayoun, hagen}@cs.uni-kl.de

Abstract

Directed graphs are a useful model of many computational systems including software, hardware, fault trees, and of course the Internet. We present the TopoLayout-DG framework, an extension to the original TopoLayout algorithm, for visualizing the inside behaviour of large directed graphs. The proposed framework consists of: a feature-based multi-level algorithm, called ToF2DG*, that detects topological features in large directed graphs in a hierarchical fashion; and visualization methods for the resulting levels of details of the graph's topological structure. In this work-in-progress paper, we highlight the main steps of the proposed ToF2DG algorithm. Moreover, we show some preliminary visual representations of artificial directed graphs. These preliminary representations indicate that the framework promises to solve some scalability issues in the visualisation of large directed graphs.*

Categories and Subject Descriptors (according to ACM CCS): G.2.2 [Mathematics of Computing]: Graph Theory—Graph algorithms I.3.8 [Computing Methodologies]: Computer Graphics—Applications

1. Introduction

Producing meaningful abstract representations are crucial in the case of large data and limited display sizes. One of the common means to express it is through a multi-level representation of the original graph. This assumes providing a hierarchical structure of the original graph. This hierarchical structure can be modelled using the containment relationships between the graph components, which helps in keeping the adjacency relations between the graph nodes. In this work, we present a multi-level framework, called **TopoLayout-DG** (**TopoLayout** for **D**irected **G**raphs), for visualising large directed graphs and to try overcoming the scalability and visualisation quality issues in these graphs. The proposed TopoLayout-DG framework aims at helping in analysing the topological structures of directed graphs based on topological features of their sub-graphs.

Our solution is based on extending the TopoLayout algorithm, initially introduced by Archambault et al. in [AMA07], for visualising large undirected graphs with less edge-crossings. In this extension, we design the detection phase to take the direction of the arc into account. The main difference between the original TopoLayout algorithm and our extension is the graph type. Such that, the original TopoLayout framework was dedicated to visualise general undirected graphs, while in our case we extend it to take care of the edge direction between the nodes. In order to detect different topological features inside the graph we then create another level of the abstraction of the original graph. We concern ourselves in finding interesting topological features like strongly connected components (e.g., cycles or complete graphs), trees, and DAGs (Directed Acyclic Graphs). Moreover, we use the graph-drawing algorithms already proposed in the literature (e.g., the tree layout algorithms [Ead92], the force directed layout algorithm [Ead84]) to visualise the detected features. Our approach is based on subdividing the underlying large directed graph into a set of sub-graphs with regard to the topological features of the local sub-graphs. Then we visualise each sub-graph using one of the algorithms that is tuned to its topological feature. Once the main graph has been decomposed into the basic features that form its original structure, we compose them into a set of MetaNodes. This composition is done in a fashion that each set of nodes which are connected with each others through a specific feature, e.g. a tree-structure, are collapsed into one node called the "MetaNode". We do this for all the interesting and required topological features in the underlying main graph. The resulting representation provides an abstract hierarchi-

cal multi-level view of the whole graph in a 2D representation, which helps in overcoming the size and visual quality limitation. It also offers a new strategy to cluster large directed graphs according to their inside topological features.

The remainder of the paper is structured as follows: First we highlight briefly the related work (Sec. 2). Then we provide the framework pipeline (Sec. 3), explain our ToF2DG algorithm (Sec. 4), and describe the visualisation of the final representation (Sec. 5). Finally, we conclude (Sec. 6).

2. Related Work

A directed graph G can be defined as a set of nodes V, where these vertices are connected with each other via a set of edges $A \subset V \times V$. The presented work focuses on providing a 2D visual presentation of the directed graph G. Until now, only a few algorithms have been proposed to visualise directed graphs. The Sugiyama algorithm is one of the first algorithms that draws general directed graphs [STT81]. It works in two phases: First, it layers the graph nodes, which means assigning a layer for each node and then placing all nodes to the corresponding layers; Secondly, it reduces edges crossings and nodes overlapping. Many of the suggested Sugiyama algorithm's steps are proved as NP-hard [GJ83] while some of these steps were proved as NP-Complete [EW94].

In [KT13], authors presented DAGView framework that aimed at visualising directed acyclic graphs more clearly. They take into account the users' preferences during the layout phase such as users can control the size of the underlying grid and the crossings that appear in the final layout. H3Viewer is one of the tools that visualises large directed graphs (semi trees) in 3D spaces [Mun97]. Technically, this tool can be used for visualizing large graphs. However, visual cluttering and extra occlusion makes it difficult to read or extract information from the generated visualisation. To tackle this problem, many clustering algorithms have been proposed in order to provide a multi-level visualisation of large graphs [RS97]. Additionally, edge-clustering and edge-bundling techniques have also been proposed to reduce the cluttering issue in the final representation [Hol06]. Our work is the first try towards providing a feature-based multi-level visualisation technique for large directed graphs.

3. The Approach

The TopoLayout-DG pipeline consists of four main phases (see Fig. 1), based on the original TopoLayout algorithm [AMA07]:

1. *The Detection Phase:* This is similar to the coarsening operation in multi-level techniques. In this phase, the feature hierarchy is recursively created through our algorithm by identifying the feature type of each sub-graph (see Sec. 4).
2. *The Visualisation Phase:* The detected topological feature for each sub-graph is drawn using a layout algorithm tuned to its type.
3. *The Crossing Reduction Phase:* It aims at reducing the crossings between edges.
4. *The Overlapping Elimination Phase:* The phase reduces the overlapping between nodes in the final graph.

The TopoLayout-DG differs in working from the original TopoLayout in the first two phases. In the forthcoming sections, we focus only on these first two phases, as phases three and four are out of the scope of this work. TopoLayout-DG uses the Libgraph package [Hei11] and the OpenGL library for the realisation of detection and visualisation phases respectively. To handle the third and fourth phases of the pipeline, we use the approximate solutions provided by [AMA07].

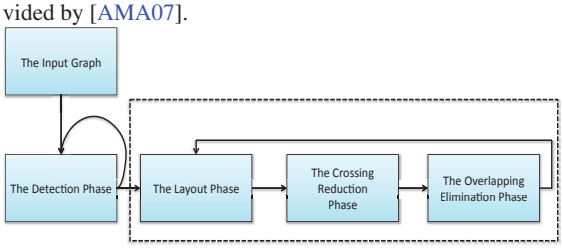


Figure 1: *The four main phases in TopoLayout-DG pipeline*

4. The ToF2DG Algorithm

We provide an algorithm, called **ToF2DG** (**To**pological **F**eatures **D**etector for **D**irected **G**raphs), for performing the topological feature-based detection during the *detection* phase of TopoLayout-DG pipeline. ToF2DG detects the set of topological features in the input graph, where each topological feature is then collapsed into one node called the MetaNode. This process is applied recursively over the graph to build the required hierarchy of the graph. Fig. 2 provides ToF2DG's pseudo code. In the following, the term *node* means the actual node in the graph while the term *MetaNode* means a group of nodes (either actual graph nodes or another set of MetaNodes), which together belong to one of the considered topological features. For keeping the original graph connectivity, TopoLayout-DG creates a *MetaEdge* to represent the initial set of edges between those nodes that belong to different MetaNodes. Below we provide the details of the algorithm.

First of all, ToF2DG finds the number of connected components in the graph G using the *depth-first-search* algorithm. This information is stored in a list C. The step of detecting the connected components is performed only once at the beginning. Each connected component is represented as a MetaNode in the final representation. For each connected component in C, ToF2DG performs the following steps: First, it measures the sub-graph size. If it is 1 then this sub-graph is detected as an isolated node and the ISOLATED

```
Procedure ToF2DG
Input: Direct Graph G
Output: Hierarchy container to describe the highest level of details
C = List of unconnected sub-graphs in G
for each sub-graph S in C do
    if sizeOf(S) == 1 then
        create y=MetaNode(S); mark y as ISOLATED;
    else
        for x in all SCC's in S do
            create y=MetaNode(x);
            if (x.edges () == x.vertices())
            mark y as CYCLE;
            else if (x.edges() > 2*x.vertices())
            mark y as COMPLETE;
            else mark y as SCC;
            create MetaEdges e⃗ to replace the connecttions of y to the other
            nodes;
            if y is SCC and |y| > limit then
                Bisect y into Metanodes u and v;create a Metaedge
                between them containing the cut edges;
                TOF2DG(u);Hierarchy.push(u);
                TOF2DG(v);Hierarchy.push(v);
            end
            Hierarchy.push(y);
            S.edges.push(e⃗);
            S.edges.remove(edges contained in e⃗);
        end
        while x = Extract.Trees(S) do
            create y = MetaNode(x);Mark y as TREE;
            Hierarchy.Push(y);
            S.Remove(x.nodes except x.root);
            Assign y to x.root;
        end
        while x = Extract.DAGs(S) do
            create y = MetaNode(x);Mark y as DAG;
            Hierarchy.Push(y);
            S.remove(x.nodes except x.root);
            Assign y to x.root;
        end
        ToF2DG(S)
    end
end
```

Figure 2: *The ToF2DG pseudo code.*

feature is assigned to it. ToF2DG creates a MetaNode to represent it in the final view (see Section 5). If the size is greater than 1, ToF2DG finds the Strongly Connected Components (SCCs) using the Tarjan algorithm [Tar72]. For every SCC, ToF2DG checks if it is a normal cycle or a complete directed graph based on the number of edges in this SCC, using an approximate solution similar to the proposed one by [AMA07]. In both cases, it creates a MetaNode to represent the nodes in the underlying SCC and assigns the CYCLE or COMPLETE label accordingly. This step is performed iteratively until all SCC features are detected and collapsed into MetaNodes.

For SCCs which are neither CYCLE nor COMPLETE, the following bisection is proposed, which is a compromise between a minimum number of cut edges and an even bisection: a node is chosen at random and the other nodes are classified according to the distance (i.e., the number of edges) from that node. We denote these node classes as D_e with e = distance. We define $A_j := D_0 \cup \ldots \cup D_j$, $B_j := D_{j+1} \cup \ldots \cup D_n$, $\alpha(e) :=$ the number of edges between nodes in A to nodes in B, $n :=$ the maximum distance. Then the Metanode is bisected into A_j and B_j with $j = \max_i\{\Big(\min\{|A_j|, |B_j|\}/\alpha(i),\ i\Big)\}$ (see Fig. 3).

Then the MetaNodes are created for them and a MetaEdge to connect them is also created. After it, ToF2DG is applied on these MetaNodes again. All MetaNodes are then pushed in a special data structure, called *Hierarchy*, to represent the next level of detail of the input graph. This *Hierarchy* container stores all the required information about the original nodes and their corresponding MetaNodes. After storing the required information in this container, ToF2DG removes all the original nodes, except the roots, from the original graph, as now a MetaNode is there to represent them in the graph next view. This helps in reducing the time complexity during the detection phase.

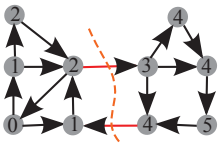


Figure 3: *Bisection of an SCC into an SCC (left) and a DAG (right). The numbers are the distance from node 0.*

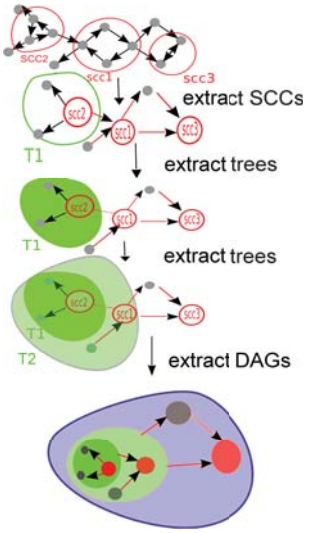


Figure 4: *An example of topology extraction. Colours indicate the feature type while the nested nodes show the hierarchy of the graph*

After the above step, ToF2DG starts detecting the acyclic features in each particular connected component. ToF2DG finds the tree structures by searching all leaves in the subgraph (with in-degree = 0 or out-degree = 0). For this, it starts from one leaf and then expands upward to the leaf parents until the root of the tree is found. Then it creates a MetaNode to represent this set of nodes and assigns a TREE feature to this MetaNode. ToF2DG detects all Tree-structures in two directions: from top to bottom (called *DownTree*) in which all edges are directed away from the root to leaves, and in bottom-up style (called *UpTree*) in which directions of the edges go from leaves to the root. ToF2DG performs this iteratively until there are no more trees in the sub-graph.

For detecting a DAG structure, ToF2DG starts from those leave nodes that have two parents. A DAG can have more than one parent compared to only one in a tree. ToF2DG performs the DAG detection steps in the same manner as the

tree detection steps. It distinguishes in this structure between the downward direction and the upward direction. Here the root nodes are not removed from the graph because they keep the graph connectivity.

The result of ToF2DG is that each node in the original graph is associated with a topological feature attached to its MetaNode. The set of MetaNodes represents the graph hierarchy. It is used to produce the higher level of detail for the same graph resulting in a *MetaTree*. As in the original TopoLayout algorithm, the graph hierarchy represents the different levels of detail such that level i is a parent of level $i+1$, where each level is a different abstraction of the graph. In order to create a multi-level representation of the graph, ToF2DG performs the same mentioned steps over the list of MetaNodes to find how these nodes are topologically connected with each other. In this step, a MetaGraph is created to represent the next higher level of details. This step is repeated until the targeted graph is abstracted into one single node, which represents the highest and the most abstracted level of details of this sub-graph. The process of feature extraction is illustrated in Figure 4.

5. The Layout Phase

Due to the application of ToF2DG, each vertex in the original graph now refers to a topological feature and belongs to a MetaNode which is passed to the visualisation phase. First the topological feature of each MetaNode is checked and then it is passed to the corresponding layout algorithm tuned to the MetaNode's topological feature. Then it is visualised according to one of the following cases:

- *The ISOLATED Structure:* In this case, a point is displayed on the screen to represent the MetaNode.
- *The TREE Structure:* If the topological feature is a tree then the radial layout algorithm is called. In this, internal nodes of the underlying MetaNode are displayed as a red colour radial tree. Because we deal with relatively large graphs, we selected the radial layout [Ead92] to utilize screen space efficiently.
- *The DAG Structure:* If internal nodes of the MetaNode are connected as a DAG then the force directed layout [Ead84] is used to visualise it. However, we are planning to use the Sugiyama algorithm [STT81] in future as it is more convenient for DAGs or drawing layered graphs.
- *The COMPLETE/CYCLIC Structure:* In the case of complete or cyclic topological feature, the circular layout algorithm is used to visualise the nodes.

TopoLayout-DG framework creates multilevel views of the original graph in a manner that different views of the graph are related somehow to each other in a multi-level representation. The edge direction between any two nodes is shown using the colour interpolation from red (source) to green (target). Through this, each cell represents a MetaNode while the included graph provides the lower level of detail. In future, we aim at relating the multilevel of the structure without losing the global context.

6. Conclusion

In this work-in-progress paper, we presented the TopoLayout-DG framework to handle directed graphs. The framework offers ToF2DG algorithm for detecting the topological features in large directed graphs. In the future, we intend to provide practical results using large directed graphs. The proposed framework can improve the visualisation of compound graphs naturally, as it provides a better understanding of the structural relations between the graph nodes using the multi-level layout technique. This solves some scalability issues in the final layout of large graphs. For example, visualising large software systems according to the topological features between the different system elements in a multilevel fashion can help software architects in understanding the behavioural pattern of the system. However, one important aspect to be investigated is the time complexity analysis of the framework. We also intend to analyse the time complexity using different theoretical and practical methods while testing TopoLayout-DG over large directed graphs in real situations.

References

[AMA07] ARCHAMBAULT D., MUNZNER T., AUBER D.: Topolayout: Multilevel graph layout by topological features. *IEEE TVCG 13/2* (2007), 305–317. 1, 2, 3

[Ead84] EADES P.: A heuristic for graph drawing. *Congressus Numerantium* (1984), 149–160. 1, 4

[Ead92] EADES P.: Drawing free trees. *Bulletin of the Institute for Combinatorial and its Applications 5(2)* (1992), 10–36. 1, 4

[EW94] EADES P., WHITESIDES S.: Drawing graphs in two layers. *Theoretical Computer Science 131* (1994), 361–374. 2

[GJ83] GAREY M. R., JOHNSON D. S.: Crossing number is np-complete. *SIAM Journal on Algebraic and Discrete Methods* (1983). 2

[Hei11] HEINE C.: Libgraph, November 2011. URL: http://www.informatik.uni-leipzig.de/~hg/libgraph/. 2

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE TVCG.* (2006). 2

[KT13] KORNAROPOULOS E. M., TOLLIS I. G.: DAGView: An approach for visualizing large graphs. In *Proceedings of the 20th International Conference on Graph Drawing* (Berlin, Heidelberg, 2013), GD'12, Springer-Verlag, pp. 499–510. 2

[Mun97] MUNZNER T.: H3: laying out large directed graphs in 3d hyperbolic space. *Proceedings of the IEEE Symposium on Information Visualization* (1997). 2

[RS97] ROXBOROUGH T., SEN A.: Graph clustering using multiway ratio cut. *Proceedings of Graph Drawing, Lecture Notes on Computer Science 1353* (1997). 2

[STT81] SUGIYAMA K., TAGAWA S., TODA M.: Methods for visual understanding of hierarchical system structures. *IEEE Transactions On Systems Man And Cybernetics* (1981). 2, 4

[Tar72] TARJAN R. E.: Depth-first search and linear graph algorithms. *SIAM J. Comput. 1*, 2 (1972), 146–160. 3

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

Multi-Scale Trend Visualization of Long-Term Temperature Data Sets

Andreas Kerren[1], Ilir Jusufi[2], and Jiayi Liu[1]

[1]Linnaeus University, Department of Computer Science, ISOVIS Group, Växjö, Sweden
[2]University of California, Department of Computer Science, Davis, CA, USA

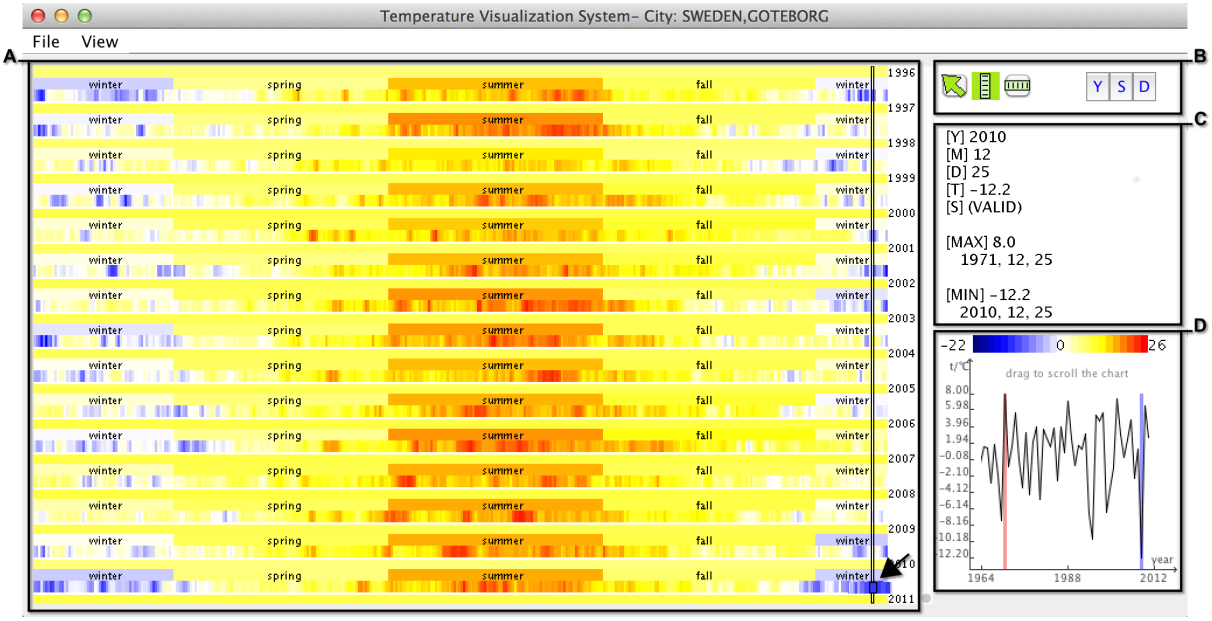


Figure 1: *An overview screenshot of our visualization tool. It shows temperature data for Gothenburg, Sweden, over a time span of 52 years in total. Region A indicates the* main view *of the tool. A* toolbox *is located in the top-right corner (B) and provides quick access to several selection options and view modifications. Information about selected data items is shown in textual format by the* details view *marked with C. The* selection view *(D) shows different selection ranges in the form of a standard timeline plot. A (vertical) range for December 25 over all available years was selected by the user (black arrow).*

Abstract

The analysis and presentation of climate observations is a traditional application of various visualization approaches. The available data sets are usually huge and were typically collected over a long period of time. In this paper, we focus on the visualization of a specific aspect of climate data: our visualization tool was primarily developed for providing an overview of temperature measurements for one location over decades or even centuries. In order to support an efficient overview and visual representation of the data, it is based on a region-oriented metaphor that includes various granularity levels and aggregation features.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces (GUI) I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques J.2 [Computer Applications]: Physical Sciences and Engineering—Earth and atmospheric sciences

1. Introduction

Together with the daily discussions on the climate change in climate research and media in general, the need to analyze the permanently increasing amount of climate data is becoming overwhelming. Here, interactive visualizations—possibly in combination with traditional data mining methods—are the key to get meaningful results out of these typically huge time-dependent data sets.

Many advanced and powerful tools exist for the visualization of general time-series (cf. the survey [AMM*08], for instance), but most of them are not really used in climate research and unknown to many researchers in this domain. Our aim in this work was the development of an easy to use visualization tool that supports overviews of long-term climate data and facilitates interactive trend analyses. In order to solve such tasks, we had to decide on the visual representation and functionality of our tool: we concentrate on one user-defined location (i.e., city) and one scalar measurement (i.e., temperature), as well as the implementation of a region-oriented visualization metaphor that supports time visualization at multiple scale levels (granularity).

The rest of this paper is structured as follows. Section 2 discusses the most important related work and highlights main differences to our visualization tool. Next, the properties of the input data are given (Section 3). Section 4 presents our visualization approach, discusses interaction features, and provides some implementation aspects. Finally, we conclude with Section 5.

2. Related Work

A large variety of visualization systems have been developed for the interactive analysis of climate data over the past years. Many of them are map-based, i.e., geo-spatial climate data is represented as a map with a gradient color coding. Since map-based tools only present data at one time point, developers either use animation (e.g., trigged by a time slider) to display the whole history of usually one climate attribute, such as done in [VW14]; or they use glyphs for representing (multivariate) climate data over time, like the tool presented in [TSWS05]. Other more advanced visualization tools are composed of several coordinated views (e.g., line plots, maps, or scatterplots) and provide a rich set of interaction and visual analysis techniques, for instance SimVis [LSL*09] or WeatherSpark [Wea14]. Because of the page limit, we do not discuss general temperature and climate data visualizations and refer to the overview paper of Nocke et al. [NSBW08]. In the following paragraphs, we concentrate on our restricted focus area, i.e., on trend visualization of long-term temperature data sets as discussed in the introduction.

According to the previously mentioned overview paper, "other temporal visualization techniques such as pixel-oriented visualizations are rarely used [in climate research]" [NSBW08]. This still appears to be the case also nowadays, but there are some approaches that are clearly related to our design considerations and implementation. As we want to provide an efficient overview of long-term temperature trends, pixel-oriented techniques are particularly well-suited to display this kind of data. Recursive patterns [KKA95] are an early realization of a pixel-based approach for large multivariate data sets. They also support a way to show the hierarchical structure of time-dependent data, i.e., colored pixels may be arranged as groups that represent data for days, for instance, and these groups can be again arranged for other levels of granularity like weeks. However, recursive patterns do not inherently support advanced interaction possibilities. The pixel-based GROOVE tool [LAB*09] supports more interaction and advanced features like the combination of aggregated values with detailed information by using several types of overlays. In contrast to our approach, it is not possible to visualize a trend explicitly by a timeline, e.g., to show the temperature trend of the winter seasons over several years.

A related technique are so-called tile maps [MFSW97] which were designed to represent air quality trends. The attribute values (such as ozone measurements) are arranged on a grid following a calendar division. This approach provides a good overview and also the possibility to compare values of a specific week day (comparison of row elements of the grid) or weeks (comparison of grid columns). But, it is neither interactive, nor does it provide any aggregation features. Shimabukuro et al. [SFdOL04] propose a visualization technique for the analysis of trend patterns at several granular levels. This approach is pretty similar to ours, but they arrange the different levels (days, months, years) into three big separated blocks or regions, i.e., there are two cells for a specific month—one shows the details for each day of this month (by using a pixel-based approach), and the other one shows aggregated data (by using a colored tile). In contrast, our approach hierarchically arranges data of different granular levels side by side similar to an icicle plot.

3. Input Data

We obtain our data sets from the European Climate Assessment & Dataset project (`http://eca.knmi.nl`). Their website offers free data sets collected from different meteorological stations throughout Europe and the Mediterranean. The data can be downloaded via interactive queries as comma separated values (.csv) text documents. Available query parameters are location (country, city), period, elevation, series type (blended or non-blended data of nearby stations), and attribute (average/maximal/minimal temperature, amount of precipitation/sunshine, etc.).

Here, we focus on daily average temperature data. The data comprises temperature measurements that may span more than a hundred years back. Possible faulty readings and errors are marked in the downloaded data files. We tackle this problem of missing/faulty data by using green color in our visualization tool to denote such values.

Figure 2: *Temperature visualization of a single year (2013 in this case). The daily temperatures are drawn from left to right as small color blocks (or nodes) at the bottom of the visual representation. Blueish colors show temperatures below zero degree Celsius, and yellow to orange colors depict temperatures above the freezing point. Color saturation represents the temperature value. Green color indicates missing/faulty data. Seasons are placed on top of the days and are color-coded based on the average temperature of the season. The top-most horizontal strip represents the yearly average temperature.*

4. Visualization and Interaction Approach

In the following, we discuss our visualization and interaction approaches for the visual analysis of temperature time-series together with a brief discussion on implementation aspects.

4.1. Views

We want to show as much historical temperature data as possible in a single view, while also allowing the user to get insight on selected data ranges. To achieve this, our tool is divided into four distinct parts as shown in Figure 1. The *main view* visualizes the overall temperature at different granular levels (see Figure 1(A)). The *toolbar* holds different buttons that enable various interaction modes of our prototype (Figure 1(B)). The *details view* shows textual information about a selected data point (Figure 1(C)), i.e., the exact date and temperature of the selected/hovered day. When a full range has been selected (see Sect. 4.2), the highest and lowest temperature dates of the selection appear. Finally, the *selection view* shows a timeline visualization of the selected data (Figure 1(D)). In the following, we explain the main and selection views in more detail.

Main View Our initial inspiration came from pixel-based approaches [KKA95]. The advantage of such approaches is that they make maximal use of the available screen space, i.e., we could map the temperature of one day to a single pixel. However, interacting with pixel-based views is sometimes cumbersome due to the small pixel size. Moreover, we wanted to include aggregated data in the form of (meteorological) season and year averages. As each calendar year is divided into four seasons comprised of 365 days, we used a variation of icicle plots to show the daily temperature of a year while providing insight into the averages for the individual seasons and complete years, cf. Figure 2. To build an overview of the entire data set, we stack the visual representations for all years in ascending order as seen in Figure 1(A). We use vertical sliders to navigate the data when dealing with large data sets. One issue of this strategy is that the winter seasons are not mapped perfectly to the years, because they start at the beginning of December and end at the end of February. Therefore, instead of four, we have five segments in the seasons levels as each winter season is split into two parts.

Selection View The selection view (Figure 1(D)) uses a traditional timeline approach and accentuates the highest and lowest temperature dates of the selection as well, while providing context to the other days. The extreme temperatures are marked with blue and red vertical lines respectively. While the textual information of the *details view* gives exact values of such data, the *selection view* provides additional insight into the condition of such extreme temperatures. For example: is the coldest day the result of a gradual temperature drop, or was it a sudden event that happened within a time span of 1-2 days? Those questions can be answered with the help of the timeline plot as shown in Fig. 3.

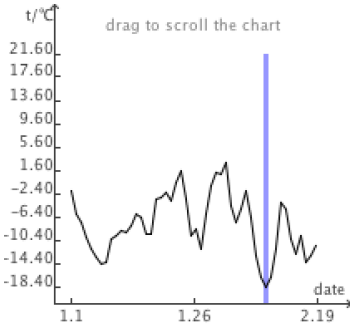


Figure 3: *The* selection view *uses a timeline plot to represent different selection ranges. The blue vertical line represents the lowest temperature in a given range. The timeline can be dragged and scrolled horizontally to view the rest of the data. Zooming is not supported.*

4.2. Interaction

Initially, our tool shows all scale levels at once. Hiding some levels might be useful to reduce clutter. For instance, if users are only interested in season temperatures, then they can hide the day levels—or even the year levels—to fit more data into the view. When a user decides to hide or show a scale level by clicking the toolbox buttons, our tool must redraw the entire view. In order to soften the effect of loosing the mental map in such cases, we use a simple morphing technique to smooth vertical transition changes.

Although we do not represent the days as single pixels in our approach, their width is still small enough to hinder easy mouse selection, especially if the neighboring days share similar temperature values. This makes it even harder to distinguish the boundaries of each day. On the one hand, we overcome this issue by using the *details view* which shows the data of the day node under the mouse cursor. On the other

hand, we have implemented an one-dimensional fisheye lens to enable comfortable selection by magnifying the day nodes interactively. In consequence, nodes in the center of the lens will be magnified whereas border nodes will be compressed in the same day level (cf. the conceptional behavior of our fisheye lens in Fig. 4 and the arrow in Fig. 1).

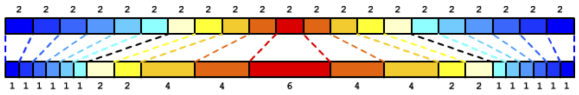


Figure 4: *A conceptual schema of the one-dimensional fisheye lens for day scale level with a size of 21 node elements. The labels above/below the nodes indicate how many pixels the related nodes encompass without (top) and with (bottom) using the lens.*

Besides single node selection, our tool offers selection possibilities across full horizontal or vertical ranges, i.e., users can select all the days or seasons of a year, or select one particular day, a season or all years of the entire data set. A 'horizontal range selection' can be used to answer questions about extreme temperatures within a year. A 'vertical range selection' can give answers to questions like: what were the hottest and coldest Christmas days in Gothenburg? The answer in this case is sufficiently perceivable in Fig 1. Here, the user has selected the vertical range for December 25 by using the toolbar and mouse selection (see Figure 1(B) and the arrow in the bottom-right corner of the *main view*). The *details view* shows us then the lowest and highest temperature recorded during the last 52 years.

4.3. Implementation Details

The prototype is implemented in Java in order to support multiple platforms [Liu12]. No third-party visualization libraries were used, i.e., the visualizations were programmed in pure Java Graphics2D without hardware acceleration. Our visualization approach mainly involves the drawing of rectangles. The number of rectangles to be drawn depends on the size of the data set and is calculated as $n = d + (1+5)y$, where d is the total number of data records (i.e., all days of the time range), y is number of years, and n is the total number of rectangles. For each year $y = d/365$, we need six rectangles to represent the year itself and the season level (leap year excluded). Thus, 7,420 rectangles have to be drawn if the data set includes records for 20 years, for instance. Rendering these objects once is no problem. However, the system would become too slow to handle the smooth interactions discussed in the interaction part of Section 4. Therefore, we have incorporated an image buffering mechanism that allows real time interaction even with data sets of $n \approx 37,000$.

5. Conclusions and Future Work

We presented a new visualization approach for the analysis of temperature trends of one specific location over years or even decades. An important aspect of this work are the different aggregation possibilities that are smoothly integrated within an interactive view. Even if our tool was originally designed for temperature data, it can also be used to show any type of quantitative data, such as atmosphere pressure, humidity, wind condition, or rainfall. However, we have not implemented efficient comparison techniques for such tasks by now. Those have to be added to the tool. Another aspect of future improvements are multiple selections that are needed to perform more complex analysis tasks and more support for zooming operations. In the light of those useful extensions, our current tool can be considered as an initial step towards analyzing climate data and should also be validated by user studies in the future.

References

[AMM*08] AIGNER W., MIKSCH S., MULLER W., SCHUMANN H., TOMINSKI C.: Visual methods for analyzing time-oriented data. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 14*, 1 (2008), 47–60. 2

[KKA95] KEIM D., KRIEGEL H.-P., ANKERST M.: Recursive pattern: A technique for visualizing very large amounts of data. In *Proceedings of the IEEE Conference on Visualization (Vis '95)* (Oct 1995), pp. 279–286. 2, 3

[LAB*09] LAMMARSCH T., AIGNER W., BERTONE A., GARTNER J., MAYR E., MIKSCH S., SMUC M.: Hierarchical temporal patterns and interactive aggregated views for pixel-based visualizations. In *Proceedings of the 13th International Conference on Information Visualisation (IV '09)* (2009), pp. 44–50. 2

[Liu12] LIU J.: Visualization of weather data: Temperature trend visualization. Bachelor's thesis, Linnaeus University, School of Computer Science, Physics and Mathematics, Växjö, Sweden, 2012. 4

[LSL*09] LADSTÄDTER F., STEINER A. K., LACKNER B. C., PIRSCHER B., KIRCHENGAST G., KEHRER J., HAUSER H., MUIGG P., DOLEISCH H.: Exploration of climate data using interactive visualization. *Journal of Atmospheric and Oceanic Technology 27*, 4 (2009), 667–679. 2

[MFSW97] MINTZ D., FITZ-SIMONS T., WAYLAND M.: Tracking air quality trends with SAS/GRAPH. In *Proceedings of the 22nd Annual SAS User Group International Conference (SUGI '97)* (1997), SAS, pp. 807–812. 2

[NSBW08] NOCKE T., STERZEL T., BÖTTINGER M., WROBEL M.: Visualization of climate and climate change data: An overview. In *Digital Earth Summit on Geoinformatics 2008: Tools for Globale Change Research* (2008), Wichmann, Heidelberg, pp. 226–232. 2

[SFdOL04] SHIMABUKURO M., FLORES E., DE OLIVEIRA M., LEVKOWITZ H.: Coordinated views to assist exploration of spatio-temporal data: A case study. In *Proceedings of the 2nd International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '04)* (2004), pp. 107–117. 2

[TSWS05] TOMINSKI C., SCHULZE-WOLLGAST P., SCHUMANN H.: 3D information visualization for time dependent data on maps. In *Proceedings of International Conference on Information Visualisation (IV '05)* (2005), pp. 6–8. 2

[VW14] VIÉGAS F., WATTENBERG M.: Wind Map, last accessed: 20-04-2014. http://hint.fm/wind/. 2

[Wea14] WEATHERSPARK: Beautiful Weather Graphs and Maps, last accessed: 20-04-2014. http://weatherspark.com. 2

SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

Modelling Emotional Behaviour in Virtual Crowds through Expressive Body Movements and Emotion Contagion

M. Ramos C. [1] and C. Peters [1] and A. Qureshi [2]

[1] School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm (Sweden)
[2] Department of Psychology, Edge Hill University, Ormskirk (United Kingdom)

Abstract

Virtual crowds are gaining more relevance in computer graphics. Research in this area has been of great interest for both industrial and scientific activities related with entertainment and virtual simulation, and nowadays it is possible to generate virtual crowds with believable and complex behaviour. However, expression of emotion and social behaviour for multiple characters is a research area that still remains to be further investigated. Particularly, emotional awareness and emotion contagion between artificial agents is gaining more interest in recent years. In this paper we present a work-in-progress that deals with the current state-of-the-art of emotional virtual crowds, and we present an overview of a computational model capable of generating virtual crowds with body emotional behaviour and emotion contagion. Also, we present a brief overview of a perceptual study in relation to the perception of social behaviour in virtual crowds. Our work seeks to shed more light into this area of computer graphics and to find new paths of research for future work.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Virtual Reality;

1. Introduction

In recent years, virtual crowds have gained an increasing research interest. The existing capabilities to create high-fidelity crowds allow for the generation of realistic and complex virtual worlds in real-time, but a challenging topic issue remains in regards with the behaviour of virtual crowds, especially emotional behaviour and social interaction. Considering the enormous complexity of crowds, this behaviour goes further beyond path-finding and local avoidance [TGMY09]. Social and emotional behaviours are two important aspects in relation to this [MMSN09], particularly the emotional awareness and the emotion contagion between artificial individuals [PDP*11].

Modelling better computational models for behaviour in virtual crowds, specially emotional behaviour, is of great interest not only in the areas of special effects and virtual realism related with films and video-games, but also in simulation of crisis-management where crowds play a core role, such as simulation of panic situations or emergency evacuations. Emotional contagion, defined as the tendency to catch another person's emotions [HTC94], has great importance in psychology and sociology as well, and several studies have proven the significance of this contagion effect in the way humans and other species behave, which gives more importance to the seek of better computational models dealing with this phenomenon.

In this paper, we present our work-in-progress related with emotional crowds. In the next section (Section 2) we present an overview of the state-of-the-art in the areas of emotion and virtual agents. In Section 3, we explain the process of generating an emotional virtual crowd, including the way we animated the characters from motion capture technologies and the description of a prototype of an emotion contagion model based on the work of Pereira et. al. [PDP*11]. We also include a brief overview of a recent perceptual study in Section 4, which serves also as an use example of our model, in this case with the purpose to investigate the effects in perception of contextual background crowds [RCQP]. Finally, in Section 5 we conclude with final remarks and open paths of research in these areas.

Figure 1: *Androgynous mannequin models composing the virtual crowd.*

2. State of the art

Extensive research has been done in the simulation of virtual agents and crowds [Rey87, MT97], including artificial behaviour for collision avoidance [GCK*09] and autonomous agents [ST07]. Also, there have been recent research regarding the perception of crowds in relation to emotional expression [EHEM13] and body language [PEE13]. More recent studies have investigated the generation [HMBP05] and mapping [CMPM12] of expressive motions between artificial agents and real people.

Emotion contagion has been widely investigated in relation to the theory of this phenomenon and its implications in human behaviour [HTC94]. Particularly for virtual crowds, there have been several efforts in developing generic computational models to simulate the effects of emotional contagion between artificial individuals [LLB11, PDP*11].

We continue in the next section by describing the process for creating the virtual crowd and its emotional behaviour.

3. Crowd generation

Since our work focuses in full-body motions of characters, we used a simple model of a free androgynous mannequin (see Figure 1) which was available from the online assets database TurboSquid. We chose this model since it did not include facial expressions or other details (for example, finger motion). In order to identify the direction of the face more easily, the original model was slightly modified by adding a small nose to its head.

The virtual crowd was generated in the Unity engine replicating the androgynous mannequin presented above. For the composition of the crowd, we considered two different types of characters: static small groups and mobile individual walkers (See Figure 1). Mobile characters were steered using an A* Pathfinding implementation and a graph-mesh with static way points and several destiny nodes.

3.1. Annotated affective data corpus

For the emotional animation of the virtual crowd, we used two annotated motion-capture libraries based on acted emotions: the Carnegie-Mellon Graphics Lab Motion Capture Database and the UCLIC Affective Posture and Body Motion Database [KDSBB06], both databases of acted expressions recorded with a VICON motion capture system. These databases included an extensive collection of emotional animation based on full-body motion. These animations were imported into 3D Studio Max, where they were mapped to the skeleton of the mannequin model and then exported to the Unity game engine.

We focused the emotions of virtual agents concerning just happy and sad full-body emotional expressions, two of the six Ekman basic emotions [Ekm92], in addition to a neutral emotional expression (See Figure 2).

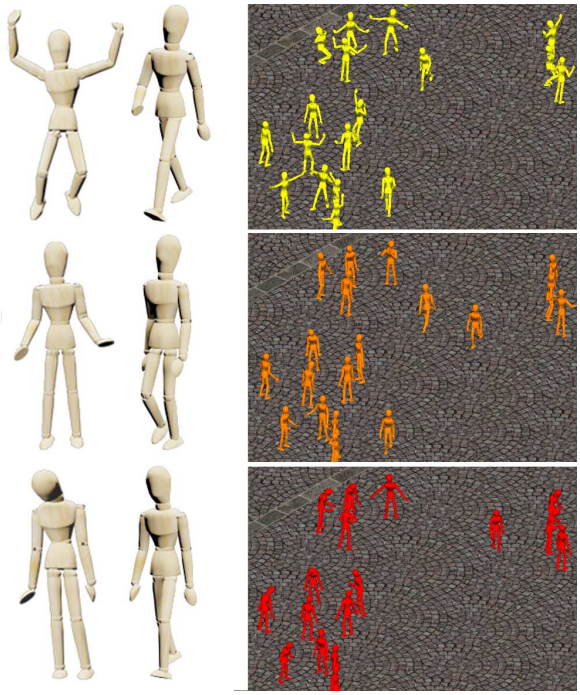


Figure 2: *Standing and walking characters of the crowd displaying happy (above), neutral (middle) and sad (below) full-body motion behaviours. The agents in the crowd were coloured automatically according to their emotion.*

The animations were selected from the motion-capture libraries using the annotations provided by their authors, and the transitions were blended with Mecanim (Unity's animation system) using finite state machines. We run a pre-study based on a perceptual experiment to confirm the correlation between each animation and each mood (See Section 4).

3.2. Emotion contagion

The emotional behaviour of the crowd was integrated at the individual agent level, and the model presented here is a simplification of a previous work on this area [PDP*11].

To define the emotional state of each agent, we defined a one-dimensional scale in the range [-1, 1] to represent each of the three moods that the model uses, being -1 sad, 0 neutral and 1 happy. The animation transitions between the emotional states were blended according to this scale. We also defined an additional parameter to probabilistically determine the susceptibility of contagion. Higher percentages imply more probabilities of being susceptible to catch others emotions. Thus, each agent was represented by the tuple <*e, s*>, being *e* the current value of its emotional state and *s* the probability of being emotionally affected by others.

The changes in mood inside the crowd emerge from the emotional awareness between agents, which is based on the emotional contagion model. With this, each agent is able to perceive, appraise and react to others emotions according to the three modules that compound the model: the *perception module*, the *appraisal module* and the *contagion module* (see Figure 3).

3.2.1. Perception module

To implement the *perception module*, a field of view (5 meters in scale) was defined for each agent to represent what the character sees and its field of awareness of others emotions. The perception occurs when an emotional agent, i.e. character A, intersects with the field of view of another agent, i.e. character B. When this happens, character B receives the value *e* of the emotional state of character A.

3.2.2. Appraisal module

Once an agent B has perceived the emotion of another agent A, the next step is handled by the *appraisal module*. At this stage, there is an evaluation to check the possibilities of emotional contagion. This evaluation is made through the susceptibility parameter *s* of the agent B. Through a random function based on an uniform distribution (*random()*), the module calculates a real number between 0 and 1 and compares it with the parameter *s*. The agent B will be affected by the mood of the agent A if *random()* < *s*.

To prevent loops and make the behaviour of the crowd more believable, after each evaluation and after any possible contagion, agent B sets a time lapse *t* in which it will be immune to emotional contagion.

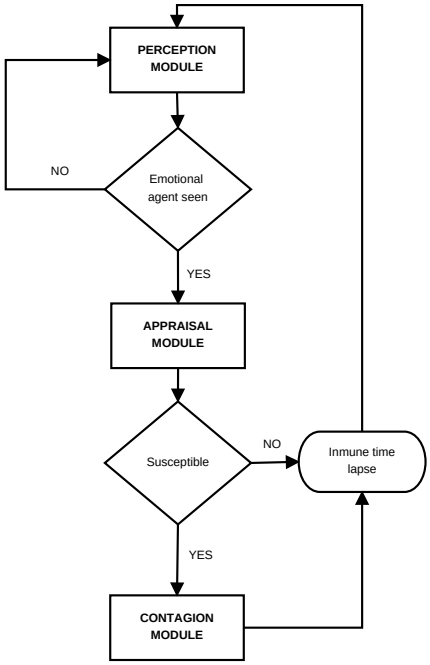


Figure 3: *Emotional contagion model diagram.*

3.2.3. Contagion module

In the last step, if the susceptibility was positive in the previous module, then contagion occurs, and the *contagion module* handles the change of mood. In this module we implemented two alternative approaches: 'strong contagion' and 'contagion by steps'.

In the first approach (strong contagion) the character that has been affected simply copies the value of the emotional state of the agent perceived, converging both emotionally. Thus, for example, if a happy character (1) is affected by a sad character (-1), the happy character will change its emotional state to sad (-1).

The second approach (contagion by steps), is less aggressive in terms of contagion, and in this case the emotional state of the character affected is moved one step down/up (-1/+1) in the one-dimensional mood scale, depending on the emotional state of the character perceived. In this case, if a happy character (1) is affected by a sad character (-1), the happy character will change its emotional state one step down, in this case, to neutral (0).

4. Perceptual study

With the virtual crowd model described in the previous section, we did a series of user studies to provide feedback for the computational model and to investigate different aspects of perception of emotions in virtual social context. We created several video scenes portraying virtual crowds and we

asked participants to rate the mood of them on a five point Likert scale (from 1 = negative to 5 = positive, where 3 = neutral). Specifically, in one of the experiments, participants (22M:6F) were asked to rate the mood of each of the three emotional animations to independently verified that the sad, happy and neutral animations were perceived as such when mapped onto the specifics of our characters. This study helped us to confirm the correlation between the animation of the motion-capture databases and the mood that we were intending to convey in each of the scenes.

In addition to this, more complex experiments were done regarding contextual aspect of virtual crowds. In a recent study, we investigated the effects of a background virtual crowd in the perception of the mood of several scenes [RCQP].

In the next section we present some ideas of future work in relation to these perceptual studies.

5. Future work

In this paper, we focused on the implementation of emotional behaviour in virtual crowds in terms of body movements and, specifically, in emotion contagion. The phenomenon of emotional contagion still needs to be further investigated, but the model presented here can be used as a base of more complex implementations. We can defined several paths of potential improvement. In our work we have just considered three emotional states (happy, neutral, sad), but the consideration of more complex emotions, such as anger or panic, could lead to a much better computational model and more realistic behaviour for virtual crowds. Also, it could be interesting to consider a more complex appraisal module with history of previous contagion for each character and with non-static susceptibility values. In relation to the mood scale, more efforts could be put in the representation of the emotional state of each agent through non-discrete values or multi-dimensional mood scales.

With respect to further uses of this model, there are a number of potential user studies that can be done in order to investigate different aspects of crowd behaviour and how people perceive it. The study mentioned in Section 4 is an example of a study regarding the effects of virtual crowds in different scene contexts, but other experiments related with perception in panic situations or during mood changes could lead to very interesting results as well. Also, immersing the participants with the virtual crowd through virtual reality could also be an exciting area of investigation that could open many interesting paths of research in the area of simulations, video games and special effects.

Acknowledgements

Thanks to Nadia Berthouze and her team at UCL for the use of the UCL Interaction Centre (UCLIC) Affective Body Position and Motion Database.

References

[CMPM12] CASTELLANO G., MANCINI M., PETERS C., MCOWAN P.: Expressive copying behavior for social agents: A perceptual analysis. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* (2012), 776–783. 2

[EHEM13] ENNIS C., HOYET L., EGGES A., MCDONNELL R.: Emotion capture: Emotionally expressive characters for games. *Proceedings of the ACM SIGGRAPH Conference on Motion in Games* (2013), 53–60. 2

[Ekm92] EKMAN P.: An argument for basic emotions. *Cognition and Emotion* (1992), 169–200. 2

[GCK*09] GUY S. J., CHHUGANI J., KIM C., SATISH N., LIN M. C., MANOCHA D., DUBEY P.: Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *ACM Siggraph/Eurographics Symposium on Computer Animation* (2009), ACM, pp. 177–187. 2

[HMBP05] HARTMANN B., MANCINI M., BUISINE S., PELACHAUD C.: Design and evaluation of expressive gesture synthesis for embodied conversational agents. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems* (New York, NY, USA, 2005), AAMAS '05, ACM, pp. 1095–1096. 2

[HTC94] HATFIELD E., T. CACIOPPO J.: *Emotional Contagion*. Cambridge university press, 1994. 1, 2

[KDSBB06] KLEINSMITH A., DE SILVA P., BIANCHI-BERTHOUZE N.: Cross-cultural differences in recognizing affect from body posture. *Interact. Comput. 18*, 6 (Dec. 2006), 1371–1389. 2

[LLB11] LHOMMET M., LOURDEAUX D., BARTHES J.-P.: Never alone in the crowd: a microscopic crowd model based on emotional contagion. *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology* (2011). 2

[MMSN09] MCHUGH J., MCDONNELL R., SULLIVAN C. O., NEWELL F.: Perceiving emotion in crowds: the role of dynamic body postures on the perception of emotion in crowded scenes. *Experimental Brain Research 204 (3)* (2009), 361–372. 1

[MT97] MUSSE S., THALMANN D.: A model of human crowd behavior : Group inter-relationship and collision detection analysis. In *Computer Animation and Simulation âĂŹ97*, Thalmann D., Panne M., (Eds.), Eurographics. Springer Vienna, 1997, pp. 39–51. 2

[PDP*11] PEREIRA G., DIMAS J., PRADA R., A. SANTOS P., PAIVA A.: A generic emotional contagion computational model. *Affective Computing and Intelligent Interaction 6974* (2011), 256–266. 1, 2, 3

[PEE13] PAMERNECKAS J., ENNIS C., EGGES A.: Perception of complex emotional body language of a virtual character with limb modifications. *Proceedings of the ACM Symposium on Applied Perception* (2013). 2

[RCQP] RAMOS C. M., QURESHI A., PETERS C.: Evaluating the perception of group emotion from full body movements in the context of virtual crowds (2014). 1, 4

[Rey87] REYNOLDS C.: Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 25–34. 2

[ST07] SHAO W., TERZOPOULOS D.: Autonomous pedestrians. *Graph. Models 69*, 5-6 (Sept. 2007), 246–274. 2

[TGMY09] THALMANN D., GRILLON H., MAIM J., YERSIN B.: Challenges in crowd simulation. *CyberWorlds* (2009), 1–12. 1