SIGRAD

2016

Proceedings of SIGRAD 2016

May 23rd and 24th
Visby, Sweden

Edited by Masaki Hayashi

Linköping Electronic Conference Proceedings No. 127
ISSN: 1650-3686,
eISSN: 1650-3740
ISBN: 978-91-7685-731-1
URL: http://www.ep.liu.se/ecp/contents.asp?issue=127

FOREWORD

Welcome to SIGRAD 2016 –
the annual meeting of the Swedish Computer Graphics Association (SIGRAD) – taking place in 2016 at Uppsala University, Campus Gotland in Visby, Sweden.

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 2016 offers a scientific program representing a cross-section of research in a multitude of domains related to visualization and human computer interaction. In addition to the regular submissions concerning general computer graphics practices, this year's conference also investigates synergies between industry, pedagogical practitioners and academic researchers through a series of presentations, keynote speeches and special sessions. The selection of 6 papers presented at the conference come from researchers in 4 different countries including Japan and China which is the first participation from Asian countries in the history of the conference. These papers range from those concerning general computer graphics practices to a practical applications and services that may benefit from the use of visualizations and computer graphics technologies. The extended participation of students at all levels of academia in research has been encouraged this year and 2 papers are selected which are first-authored by students studying at Master's Degree level. Peer reviewing was conducted by a highly qualified Program Committee consisting of 24 reviewers. Each paper was reviewed, on average, by three reviewers from the committee, with the majority of papers receiving four reviews.

This year we have a special session called “Swedish Research Overview Session”. With this new event, we invite all Swedish research groups to present their academically outstanding, previously published work at the annual conference. All papers in this session have been published in an academically outstanding journals or conferences not more than two years prior to the SIGRAD conference.

We especially welcome our invited speakers: Ingrid Carlbom (Guest Professor of Uppsala University) presenting a keynote entitled “The History of Computer Graphics”. And Carlotta Capurro (Art historian and digital restorer at Visual Dimension) who gives a keynote entitled “3D reconstruction – Innovative forms of studying and experiencing the past”.

Finally, we wish to thank Olle Jansson, the Vice-Chancellor of Campus Gotland for his generous support to the event.

The SIGRAD 2016 organisers

VENUE

UPPSALA UNIVERSITY, CAMPUS GOTLAND

Uppsala University is the oldest university in the Nordic region. Research of world class and high quality degree programmes bring value to society and business in a global context. The focus is on diversity and breadth, with international frontline research being undertaken within nine faculties. There is an unrivalled selection of degree programmes at both the undergraduate and master levels.

On July 1, 2013 Gotland University merged with Uppsala University. This heralded the formation of the University's twelfth campus, Uppsala University - Campus Gotland. The vision for Campus Gotland is to create a unique profile within Uppsala University and a long-term sustainable and competitive educational centre on the island of Gotland. In this way, a contribution is made to building Sweden into a leading knowledge-centred nation. In the longer term, the objective is to increase the presence of students and teachers on campus and to develop Liberal Arts Education, network-based learning and collaboration, regionally, nationally and internationally.

PROGRAM

SIGRAD 2016

MAY 23rd

9.30 Registration

10.10 **Keynote:** The Story of Computer Graphics
Ingrid Carlbom, Uppsala University

11.30 Visiting Gotland Game Coference (GGC)

14.00 **Special session "Swedish Research Overview Session"**

(1) Multi-Touch Table System for Medical Visualization, *Patric Ljung , Anders Ynnerman, Thomas Rydell, Anders Persson, Aron Ernvik, Camilla Forsell and Claes Lundström*
(2) Coverage-Based Opacity Estimation for Interactive Depth of Field in Molecular Visualization, *Sathish Kottravel , Martin Falk, Erik Sundén and Timo Ropinski*
(3) Masked depth culling for graphics hardware, *Jon Hasselgren , Magnus Andersson and Tomas Akenine-Möller*
(4) Hybrid Data Visualization Based On Depth Complexity Histogram Analysis, *Alexander Bock , Stefan Lindholm, Martin Falk, Erik Sundén, Anders Ynnerman and Timo Ropinski*
(5) Fast Similarity Search in Scalar Fields using Merging Histograms, *Himangshu Saikia , H.-P. Seidel and T. Weinkauf*

15.20 Coffee break

15.40 **Special session "Swedish Research Overview Session" (continued)**

(1) Text Visualization Techniques: Taxonomy, Visual Survey, and Community Insights, *Andreas Kerren and K. Kucher*
(2) Real-time noise-aware tone mapping for HDR-video, *Jonas Unger, Gabriel Eilertsen and Rafał Mantiuk*
(3) Layered Reconstruction for Defocus and Motion Blur, *Jacob Munkberg , Karthik Vaidyanathan, Jon Hasselgren, Petrik Clarberg and Tomas Akenine-Möller*
(4) Compressive image reconstruction in reduced union of sub-spaces, *Jonas Unger, Ehsan Miandji and Joel Kronander*
(5) Multi-field Pattern Matching based on Sparse Feature Sampling, *Tino Weinkauf , Z. Wang and H.-P. Seidel*

PROGRAM

SIGRAD 2016

MAY 23rd (*continued*)

17.00 **Poster session**

(1) Automatic CG Talk Show Generation from the Internet Forum, *Masaki Hayashi, Steven Bachelder and Nakajima Masayuki (Uppsala University), Sweden*

(2) Digital archive of the SHISHIMAI using AR Toolkit, *Yuya Watanabe (ARCGEO Inc.) and Hidekazu Tsujiai (University of Toyama), Japan*

(3) Towards Full-Scale Ray Tracing in Games, *Afshin Ameri E. and Thomas Larsson (Malardalen University), Sweden*

(4) Pattern Generation with Cellular Automata in Hexagonal Modular Spaces, *Mikael Fridenfalk (Uppsala University), Sweden*

(5) Low Quality Mobile Image Data Processing Under Uneven Shading, *Xiaohua Zhang (Hiroshima Institute of Technology,Japan), Ning Xie (Tongji University, China), Masayuki Nakajima, Masaki Hayashi and Steven Bachelder (Uppsala University), Sweden*

19.30 Welcome party

PROGRAM

MAY 24th

09.30 **Keynote:** *3D reconstruction-Innovative forms of studying and experiencing the past, Carlotta Capurro (Visual Dimension in Belgium)*

10.30 Coffee break

10.50 **Paper session 3**

(1) Dynamic Creation of Multi-resolution Triangulated Irregular Network, *Emil Bertilsson and Prashant Goswami (Blekinge Technology Institute), Sweden*
(2) A Radial Basis Function Approximation for Large Datasets, *Zuzana Majdisova and Vaclav Skala (University of West Bohemia), Czech Republic*
(3) Vector Field Interpolation with Radial Basis Functions, *Michal Smolik and Vaclav Skala (University of West Bohemia), Czech Republic*
(4) Output Sensitive Collision Detection for Unisize Boxes, *Gabriele Capannini and Thomas Larsson (Malardalen University), Sweden*
(5) Analysis of camera work in horror movies, *Liselotte Heimdahl (studen), Yoshihisa Kanematsu (Tokyo Metropolitan University), Naoya Tsuruta (Tokyo University of Technology), Ryuta Motegi (Tokyo Metropolitan University), Kunio Kondo and Koji Mikami (Tokyo University of Technology), Japan*
(6) Interactive 4D MRI blood flow exploration and analysis using line predicates, *Jochen Jankowai (Linköping Institute of Technology, Sweden, student), Rickard Englund (Linköping Institute of Technology,Sweden), Timo Ropinski (Ulm Institute of Media Informatics, Germany) and Ingrid Hotz (Linköping Institute of Technology, Sweden)*

12.50 Closing

ORGANIZATION

HONORARY CHAIR

Hans Svensson, Uppsala University

GENERAL CHAIR

Steven Bachelder, Uppsala University

PROGRAM CHAIR

Masaki Hayashi, Uppsala University

FINANCE CHAIR

Masayuki Nakajima, Uppsala University

INTERNATIONAL PROGRAMME COMMITTEE

Morten Fjeld Chalmers, University of Technology
Lars Kjelldahl, KTH Royal Institute of Technology
Andreas Kerren, Linnaeus University
Eike Anderson, NCCA, Bournemouth University
Thomas Larsson, Mälardalen University
Jonas Unger, LiU
Michael Doggett, Lund University
Thomas Ertl, VISUS, University of Stuttgart
John Dingliana, Trinity College Dublin
Matt Cooper, Linköpings Universitet
Stefan Seipel, University of Gävle
Veronica Sundstedt, Blekinge Institute of Technology
Ruediger Westermann, TUM
Ulf Assarsson, Chalmers University of Technology
Timo Ropinski, Linköping University
Alan Chalmers, University of Warwick
Kai-Mikael Jää-Aro, Södertörn University
Tomas Akenine-Möller, Lund University / Intel Corporation
Fotis Liarokapis, Masaryk University
Nils Andersson, EON Development AB
Anders Ynnerman, Linköpings Universitet
Anders Hast, Uppsala University
Tino Weinkauf, KTH Royal Institute of Technology
Christopher Peters, KTH Royal Institute of Technology

Table of Contents

Poster Session

Keynotes

Ingrid Carlbom (Uppsala University)

The History of Computer Graphics

Abstract: It all began with Ivan Sutherland's Sketchpad in 1963 and this year the computer graphics industry is estimated to exceed $140 Billion worldwide, according to one market analyst. In this talk I will trace over 50 years of development in computer graphics, with examples from CAD/CAM, flight simulation, medical visualization, archeology, animation, and of course special effects in movies. I will put an emphasis on what has been accomplished, highlighting key milestones and my favorite animations.

Short bio: Ingrid Carlbom received a PhD in computer science under Professor Andries van Dam at Brown University. She joined Uppsala University as Guest Professor in 2008 after a 30 year research career in US research laboratories, most recently at Bell Laboratories. She served as Director of ACM SIGGRAPH, and on the editorial boards of IEEE Computer Graphics and Applications, IEEE Transactions on Visualization and Computer Graphics, and Computers & Graphics, and served as Co-Editor-in-Chief of Graphical Models. Her current research interests include automatic malignancy grading of prostate cancer and cranio-maxillofacial surgery planning. She is (co-)author of 80 refereed publications and patents. She received a Doctor of Philosophy Honoris Causa from Uppsala University, and a Distinguished Graduate School Alumna Award from Brown University. She is a member of ACM, ACM SIGGRAPH, and IEEE.

Carlotta Capurro (Visual Dimension in Belgium)

3D reconstruction – Innovative forms of studying and experiencing the past

Short bio: Carlotta Capurro is an art historian and digital restorer at Visual Dimension bvba (Oudenaarde, Belgium). She received her Bachelor's degree in Conservation of Cultural Heritage and her master's in Art History at the University of Genoa (Genoa, Italy). In Visual Dimension she does historical and iconographical research and data acquisition, 3D reconstructions and digital restoration on artefacts, monuments and sites, and works on storytelling for digital applications. She is investigating the ways in which technology is optimally used to enhance visitor experience and comprehension in museums and sites.

Special session "Swedish Research Overview Session"

The Swedish Computer Graphics Association strives to make the SIGRAD conference an annual meeting place for all national researchers in the field. With the new Swedish Research Overview Session, we invite all Swedish research groups to present their academically outstanding, previously published work at the annual conference. This serves to highlight excellent Swedish research, stimulate the discussion and initiate collaborations on a national level.
The response has been overwhelmingly positive. We got very high quality submissions of which we selected 10 papers to be presented at SIGRAD.
All papers in this session have been published in an academically outstanding journal or conference not longer than two years before the SIGRAD conference. At least one of the authors is a researcher at a Swedish university or research institute.

Multi-Touch Table System for Medical Visualization

Anders Ynnerman, Thomas Rydell, Anders Persson, Aron Ernvik, Camilla Forsell, Patric Ljung and Claes Lundström

Published in: Eurographics 2015
Presented at SIGRAD by researchers from Linköping University

Medical imaging plays a central role in a vast range of healthcare practices. While the usefulness of 3D visualizations is well known, the adoption of such technology has previously been limited in many medical areas. This paper, awarded the Dirk Bartz Prize for Visual Computing in Medicine 2015, describes the development of a medical multi-touch visualization table that successfully has reached its aim to bring 3D visualization to a wider clinical audience. The descriptions summarize the targeted clinical scenarios, the key characteristics of the system, and the user feedback obtained.

Coverage-Based Opacity Estimation for Interactive Depth of Field in Molecular Visualization

Sathish Kottravel, Martin Falk, Erik Sundén, Timo Ropinski

Published in: IEEE Pacific Visualization Symposium 2015
Presented at SIGRAD by researchers from Linköping University

In this paper, we introduce coverage-based opacity estimation to achieve Depth of Field (DoF) effects when visualizing molecular dynamics (MD) data. The proposed algorithm is a novel object-based approach which eliminates many of the shortcomings of state-of-the-art image-based DoF algorithms. Based on observations derived from a physically-correct reference renderer, coverage-based opacity estimation exploits semi-transparency to simulate the blur inherent to DoF effects. It achieves high quality DoF effects, by augmenting each atom with a semi-transparent shell, which has a radius proportional to the distance from the focal plane of the camera. Thus, each shell represents an additional coverage area whose opacity varies radially, based on our observations derived from the results of multi-sampling DoF algorithms. By using the proposed technique, it becomes possible to generate high quality visual results, comparable to those achieved through ground-truth multi-sampling algorithms. At the same time, we obtain a significant speedup which is essential for visualizing MD data as it enables interactive rendering. In this paper, we derive the underlying theory, introduce coverage-based opacity estimation and demonstrate how it can be

applied to real world MD data in order to achieve DoF effects. We further analyze the achieved results with respect to performance as well as quality and show that they are comparable to images generated with modern distributed ray tracing engines.

Masked depth culling for graphics hardware

Magnus Andersson, Jon Hasselgren, Tomas Akenine-Möller

Published in: SIGGRAPH Asia 2015
Presented at SIGRAD by researchers from Lund University

Hierarchical depth culling is an important optimization, which is present in all modern high performance graphics processors. We present a novel culling algorithm based on a layered depth representation, with a per-sample mask indicating which layer each sample belongs to. Our algorithm is feed forward in nature in contrast to previous work, which rely on a delayed feedback loop. It is simple to implement and has fewer constraints than competing algorithms, which makes it easier to load-balance a hardware architecture. Compared to previous work our algorithm performs very well, and it will often reach over 90\% of the efficiency of an optimal culling oracle. Furthermore, we can reduce bandwidth by up to 16\% by compressing the hierarchical depth buffer.

Hybrid Data Visualization Based On Depth Complexity Histogram Analysis

Stefan Lindholm, Martin Falk, Erik Sundén, Alexander Bock, Anders Ynnerman and Timo Ropinski

Published in: Computer Graphics Forum 2014
Presented at SIGRAD by researchers from Linköping University

In many cases, only the combination of geometric and volumetric data sets is able to describe a single phenomenon under observation when visualizing large and complex data. When semi-transparent geometry is present, correct rendering results require sorting of transparent structures. Additional complexity is introduced as the contributions from volumetric data have to be partitioned according to the geometric objects in the scene. The A-buffer, an enhanced framebuffer with additional per-pixel information, has previously been introduced to deal with the complexity caused by transparent objects. In this paper, we present an optimized rendering algorithm for hybrid volume-geometry data based on the A-buffer concept. We propose two novel components for modern GPUs that tailor memory utilization to the depth complexity of individual pixels. The proposed components are compatible with modern A-buffer implementations and yield performance gains of up to eight times compared to existing approaches through reduced allocation and reuse of fast cache memory. We demonstrate the applicability of our approach and its performance with several examples from molecular biology, space weather and medical visualization containing both, volumetric data and geometric structures.

Fast Similarity Search in Scalar Fields using Merging Histograms

Himangshu Saikia, Hans-Peter Seidel, Tino Weinkauf

Published in: TopoInVis 2015 and EuroVis 2014
Presented at SIGRAD by researchers from KTH Royal Institute of Technology

Similarity estimation in scalar fields using level set topology has attracted a lot of attention in the recent past. Most existing techniques match parts of contour or merge trees against each other by estimating a best overlap between them. Due to their combinatorial nature, these methods can be computationally expensive or prone to instabilities. In this paper, we use an inexpensive feature descriptor to compare subtrees of merge trees against each other. It is the data histogram of the voxels encompassed by a subtree. A small modification of the merge tree computation algorithm allows for obtaining these histograms very efficiently. Furthermore, the descriptor is robust against instabilities in the merge tree. The method is useful in an interactive environment, where a user can search for all structures similar to an interactively selected one. Our method is conservative in the sense that it finds all similar structures, with the rare occurrence of some false positives. We show with several examples the effectiveness, efficiency and accuracy of our method.

Text Visualization Techniques: Taxonomy, Visual Survey, and Community Insights

Kostiantyn Kucher and Andreas Kerren

Published in: IEEE Pacific Visualization Symposium, 2015
Presented at SIGRAD by researchers from Linnaeus University

Text visualization has become a growing and increasingly important subfield of information visualization. Thus, it is getting harder for researchers to look for related work with specific tasks or visual metaphors in mind. In this paper, we present an interactive visual survey of text visualization techniques that can be used for the purposes of search for related work, introduction to the subfield and gaining insight into research trends. We describe the taxonomy used for categorization of text visualization techniques and compare it to approaches employed in several other surveys. Finally, we present results of analyses performed on the entries data.

Real-time noise-aware tone mapping for HDR-video

Gabriel Eilertsen, Rafał Mantiuk, and Jonas Unger

Published in: SIGGRAPH ASIA 2015
Presented at SIGRAD by researchers from Linköping University

Real-time high quality video tone-mapping is needed for many applications, such as digital viewfinders in cameras, display algorithms which adapt to ambient light, in-camera processing, rendering engines for video games and video post-processing. We propose a viable solution for these applications by designing a video tone mapping operator that controls the visibility of the noise, adapts to display and viewing environment, minimizes contrast distortions, preserves or enhances image details, and can be run in real-time on an incoming sequence without any preprocessing. To our knowledge, no existing solution offers all these features. Our novel contributions are: a fast procedure for computing local display-adaptive tone-curves which minimize contrast distortions, a fast method for detail enhancement free from ringing artifacts, and an integrated video tone-mapping solution combining all the above features.

Layered Reconstruction for Defocus and Motion Blur

Jacob Munkberg, Karthik Vaidyanathan, Jon Hasselgren, Petrik Clarberg, Tomas Akenine-Möller

Published in: EGSR 2014, ACM TOG 2015, and SIGGRAPH Asia 2015
Presented at SIGRAD by researchers from Lund University

Light field reconstruction algorithms can substantially decrease the noise in stochastically rendered images. Recent algorithms for defocus blur alone are both fast and accurate. However, motion blur is a considerably more complex type of camera effect, and as a consequence, current algorithms are either slow or too imprecise to use in high quality rendering. We extend previous work on real-time light field reconstruction for defocus blur to handle the case of simultaneous defocus and motion blur. By carefully introducing a few approximations, we derive a very efficient sheared reconstruction filter, which produces high quality images even for a low number of input samples. Our algorithm is temporally robust, and is about two orders of magnitude faster than previous work, making it suitable for both real-time rendering and as a post-processing pass for offline rendering.

Compressive image reconstruction in reduced union of sub-spaces

Ehsan Miandji, Joel Kronander, Jonas Unger

Published in: Eurographics 2015
Presented at SIGRAD by researchers from Linköping University

We present a new compressed sensing framework for reconstruction of incomplete and possibly noisy images and their higher dimensional variants, e.g. animations and light-fields. The algorithm relies on a learning-based basis representation. We train an ensemble of intrinsically two-dimensional (2D) dictionaries that operate locally on a set of 2D patches extracted from the input data. We show that one can convert the problem of 2D sparse signal recovery to an equivalent 1D form, enabling us to utilize a large family of sparse solvers. The proposed framework represents the input signals in a reduced union of subspaces model, while allowing sparsity in each subspace. Such a model leads to a much more sparse representation than widely used methods such as K-SVD. To evaluate our method, we apply it to three different scenarios where the signal dimensionality varies from 2D (images) to 3D (animations) and 4D (light-fields).

Multi-field Pattern Matching based on Sparse Feature Sampling

Zhongjie Wang, Hans-Peter Seidel, Tino Weinkauf

Published in: IEEE VIS 2015
Presented at SIGRAD by researchers from KTH Royal Institute of Technology

We present an approach to pattern matching in 3D multi-field scalar data. Existing pattern matching algorithms work on single scalar or vector fields only, yet many numerical simulations output multi-field data where only a joint analysis of multiple fields describes the underlying phenomenon fully. Our method takes this into account by bundling information from multiple fields into the description of a pattern. First, we extract a sparse set of features for each 3D scalar field using the 3D SIFT algorithm (Scale-Invariant Feature Transform). This allows for a memory-saving description of prominent features in the data with invariance to translation, rotation, and scaling. Second, the user defines a pattern as a set of SIFT features in multiple fields by e.g. brushing a region of interest. Third, we locate and rank matching patterns in the entire data set. Experiments show that our algorithm is efficient in terms of required memory and computational efforts.

SIGRAD 2016
M. Hayashi (Editors)

Dynamic Creation of Multi-resolution Triangulated Irregular Network

Emil Bertilsson[†1] and Prashant Goswami[‡1]

[1]Blekinge Institute of Technology, Sweden

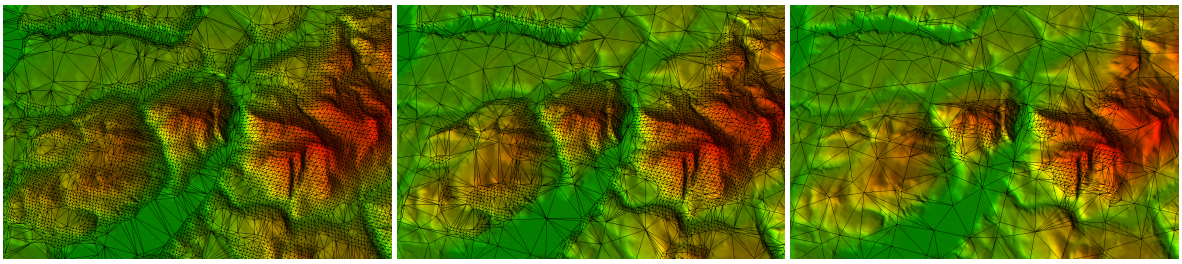


Figure 1: *Terrain mesh simplification achieved using the proposed algorithm in real time with pixel errors (from left to right) 0.5, 1.0 and 2.0.*

Abstract

Triangulated irregular network (TIN) can produce terrain meshes with a reduced triangle count compared to regular grid. At the same time, TIN meshes are more challenging to optimize in real-time in comparison to other approaches. This paper explores efficient generation of view-dependent, adaptive TIN meshes for terrain during runtime with no or minimal preprocessing. This is achieved by reducing the problem of mesh simplification to that of inexpensive 2D Delaunay triangulation and lifting it back to 3D. The approach and its efficiency is validated with suitable datasets.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics an Realism—Visible line/surface algorithms

1. Introduction

Terrain rendering can be found in a variety of applications and the detail of these terrain meshes has vastly increased over the years. Computer games and simulators often have heavy requirements on the visualization of the terrain, as well as to obtain a good resolution that can be presented without overwhelming the other parts of the application. Whether the focus lies on preprocessing level-of-detail (LOD) data or generating it during start up or in real-time, many different techniques have been developed to address the problem of representing and visualizing large quantities of terrain data.

These techniques can be divided into three major groups: regular meshes, semi-regular meshes and fully irregular meshes [PG07]. A regular mesh or digital elevation model (DEM) contains points sampled at equal distance and therefore same sized triangles. Though simple and easy to implement, the main disadvantage of purely regular triangulated meshes is the limited differentiation of the terrain features in triangulation. While there is data redundancy in areas of uni-

† emil.bertilsson91@gmail.com
‡ prashant.goswami@bth.se

form terrain, the mesh could be under represented in regions carrying high complexity.

In theory, fully irregular and semi-regular approaches differ in their capability to produce a minimal complexity mesh representation for any given error measurement. They utilize splitting, merging and subdivision to produce a triangle count closer to what is required to present the terrain features. While TIN methods can provide highly optimized meshes, most of them rely on offline preprocessing due to the heavy nature of computation involved.

When computing multi-resolution terrain models, visual artifacts can arise along the edges where areas of different resolutions meet. These inconsistencies in the mesh are handled by different forms of stitching the areas together [LKES09] or a structure that handles it automatically, providing an equal base for triangulation. Adjoining LODs are often constrained to differ in the tree by at most one level in order to avoid cracks and T-junctions [BGP09].

The novel approach presented in this paper creates a view-dependent multi-resolution TIN mesh on large models based on the user-specified tolerance and is updated throughout the execution of the application. To this end, we adopt the concept of dividing the terrain into larger regions called *patches* which are individually and independently triangulated. A vertex selection algorithm, oriented towards fast processing while approximating the mesh at a reasonable quality during runtime, combined with a Delaunay triangulation uses these patches to produce meshes during runtime. Other more complex and carefully created ways of approximating terrain can be found in [PAL02], [YLS05], [BG04], [EKT01], [TGM*06], [AW03], [HT97]. The main contributions and advantages of this method are:

1. Run-time generation of fully irregular multi-resolution meshes with no or minimal preprocessing.
2. Fast triangulation of a 3D view-dependent point selection by reducing it to a 2D Delaunay triangulation problem.
3. Independent patch processing without any mutual dependencies on the neighboring patches.

The system performs asynchronous LOD updating and supports GPU-based rendering. Delaunay triangulation was chosen as the basis due to its properties [Mus97] and simplicity, that allows for independent triangulation of points. The properties of Delaunay triangulation makes for a good basis for further parallelism in general, or delegation of the triangulation to the GPU.

2. Related Work

Terrain rendering with adaptive resolutions has been studied extensively, with different aims and angles. The earlier approaches [GGS95], [LKR*96], [Hop98], [CPS97], [XV96], [DFP95] have since been followed by more complex algorithms and more recently algorithms that can utilize the GPU to different extents. LOD techniques and multi-resolution meshes have received plentiful research and as such, many relevant papers have been published. We keep the discussion here more focused on techniques that are more related to ours.

The amount of redundant triangles created by these techniques vary on different circumstances and settings. A semi-regular mesh performs well but struggles where different terrain features meet. The transition from a low elevation area to a higher elevation area causes extra complexity due to edge following, which can give suboptimal subdivisions [CGG*03]. Fully irregular meshes struggle with the same kind of optimizations as semi-regular approaches mainly due to the arbitrary neighbours [PG07] inherent to the connection of nodes in a fully irregular mesh.

Many high performance techniques [CGG*03], [LP02], [SS09], [BGP09], [GMBP10] utilize offline precomputed LODs which are then combined at runtime to render the scene. This supplies a frame-to-frame good triangulation, achieved by performing splitting, merging and subdivision on the original mesh. The focus of this paper lies on triangulated irregular network (TIN) and generating view dependent meshes valid for a few frames up to potentially thousands of frames. The reader is referred to [DWS*97], [PG07], [SS09], [GH99] for a more in-depth analysis into other ways of achieving a multi-resolution terrain mesh.

In [LH04] geometric clipmapping for view dependent LOD is discussed and presented as a means to create a pyramidal structure to hold the cached nested regions. It operates on regular triangle grids on-the-fly without any preprocessing, using a refinement scheme of power-of-two to tessellate the terrain. Unlike the proposed algorithm in this paper, their algorithm generates intermediate areas to stitch together regions of different resolution, resembling [COL96] to ensure continuity. The technique has a number of advantages over similar algorithms, such that it can more efficiently compress and store the triangle pyramid during runtime.

Several papers discuss optimizing TIN meshes and different techniques to reduce the triangle count. One of the more interesting ones is BDAM [CGG*03], which deals with small triangle patches consisting of a few hundreds of triangles as primitives rather than single triangles. These patches, which are optimized TINs, are precomputed offline and are stored in a bintree, similar to [DWS*97]. To render the scene, a hierarchical view frustum culling is performed and the patches that are visible are then assembled. This is highly advantageous compared to other techniques that operate on a finer granularity level. However, this approach does require a lot more memory storage, and can only operate on static terrain meshes that have been preprocessed into small patches.

Another technique that utilizes the tree structure is QuadTIN [PAL02], where a quadtree hierarchy is generated over any TIN surface offline and an adaptive LOD is

stored along with the quadtree. Even though the algorithm can operate on any TIN surface, the mesh must be predefined so that the preprocessing pass can generate and store the quadtree. In [BG04], indexing the vertices in the TIN mesh is achieved in such a way that multi-resolution is achieved. By searching the mesh and defining an ordered index list in which consecutive triangles share an edge or a vertex, the mesh can be coarsened using a space-filling curve. The usage of space-filling curves is continuous and is oriented so that the highest index and the lowest index in adjacent triangles coincide.

[Lo12], [KK02] aimed at using point insertions to triangulate randomly generated spatial points using Delaunay triangulation, with slightly different approaches. The algorithm presented in [Lo12] groups together points into cells of roughly equal sizes and triangulates the cells in parallel, without any data dependency between them. On the other hand, [KK02] does not utilize cells or tetrahedrons (and operates on two dimensions rather than three), but instead performs recursive checks after each insertion. But since the points used in the algorithm presented in this paper constitute a terrain, the points have properties that random points do not. Subsequently, simplifications can be made that render the more complex algorithms in [KK02] and especially [Lo12] less fit.

While the modern techniques operate at the level of meta units for more GPU-oriented rendering, simplification can still benefit from improved methods that are capable of reducing triangle counts for TINs at runtime without incurring significant overhead. Our method works in this direction and aims at bridging the aforementioned gap. By defining a distance metric for when the mesh is updated, we can cache our triangulation which in turn provides sufficient time to triangulate the irregular mesh during runtime.

3. Method

The presented approach relies on the standard rendering feature of view-dependent pixel error to define terrain resolution. However, one big difference from earlier works is the absence of the quadtree like hierarchical data structure. The terrain is divided into square regions called *patches* and each patch is triangulated individually, allowing patches to be calculated in parrallel. The concept of patches is not new and has been explored in [CGG*03], [LKES09], [LH04], among others. The absence of a tree structure to store the patches implies that all patches belong to the same level in the tree and have the same dimensions (though varying in the level of simplification). However, a hierarchical tree structure could additionally be imposed on top of the existing patches leading to further simplification of the mesh. This is recommended in cases where terrain size is very large. On the other hand, the tree structure constructs LODs for internal nodes and hence the preprocessing step would be necessary, should it be imposed on this algorithm.

The following sections will describe how points are selected inside a patch and how the shared border points are differentiated, before sending the points for Delaunay triangulation. Finally, the overall system will be explained in Sec. 4 and how the different parts are incorporated in the algorithm as well as how the algorithm is separated from the main thread.

3.1. Patch Points Selection

In order to achieve a more coarsened patch, fewer points are selected than what is available in the original dataset. Likewise, refinement is done by adding more points than what was present before refining the patch. In order to determine which points are required for each patch, the points in the original dataset are processed. The object-space ascent/descent of a terrain point with respect to its neighbors is projected onto the screen and measured against the error metric. However, where the ascent or descent is gradual, this approach may fail to include key points which may lead to oversimplification of the terrain. By also using the two closest points that have already been selected, these gradual differences are managed.

The point selection algorithm operates on single patches and computes the same border points for adjoining patches to achieve fully independent computation. This allows calculations in parallel as the process only requires access to the list of vertices and the position of the camera. In order to improve the quality of the triangulation, additional constraints were imposed where the middle point of a border was always added as well as restricting the maximum possible distance between selected points.

Algorithm 1 Point selection

```
1:  procedure CONSTRUCTPATCH(patch p, pixelError e)
2:      // Set of interior points included in the patch
3:      P := ∅
4:      // Set of border points included in the patch
5:      B := ∅
6:      for all indices i ∈ p do
7:          if i is a corner index then
8:              B.insert(i)
9:          else if i is a border index &
10:           (heightdiff(i − 1, i) > e ||
11:           heightdiff(i, B.back()) > e) then
12:             B.insert(i)
13:             // x: closest point from previous lines
14:         else if heightdiff(i − 1, i) > e ||
15:           heightdiff(i − width, i) > e ||
16:           heightdiff(i, P.back()) > e ||
17:           heightdiff(i, x) > e then
18:             P.insert(i)
19:     return P ∪ B
```

In Alg. 1, the step by step process is shown where the

function outputs a selection of indices based on the original dataset. The procedure $heightdiff(i, j)$ returns the absolute height difference between grid points i and j. The selection algorithm is an iterative process and tests whether or not the next point in succession will exceed the error metric when compared with other close by points. These points are the immediate neighbours taken from the previous column and row, as well as the closest selected point on the same row and previous rows. The selection process is visualized in Fig. 2.

3.2. Border Points Selection

In many cases, not all patches will require new triangulation when navigating across the mesh. If patches are selectively updated without caring for dependencies, incoherent borders are inevitable where one patch is updated while an adjacent patch is not. Since the border points are shared between adjacent patches, the logic for selecting border points was separated from the internal selection process. This allowed us to ensure continuous transitions without visual artifacts. The extent of the overlap between two patches is limited to a single row of border points.

Unlike the points located inside the patches, the border points only test their height differences with other border points from the original height dataset that lie along the same edge. This is required so as to produce identical border selection for two patches sharing a border. By associating the patches with the position of the camera when the update request was sent, the pixel error for producing the border points for any adjacent patch can always be reproduced. By doing so, high resolution patches can have seamless transitions to low resolution patches. This also implies that the border points will remain in the same state until both patches that share that border receive a new update request.

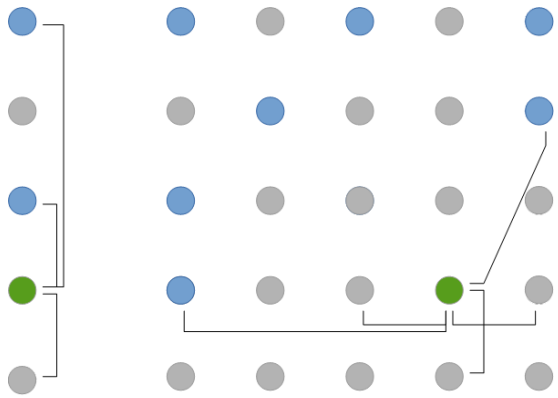


Figure 2: *Visualization of the selection process applied to a border (left image) and the interior of a patch (right image). The current point (green) is affected by its immediate neighbours as well as the closest selected points (blue).*

Fig. 2 shows how the selection algorithm operates on points belonging to a patch. In the figure, the unselected points are marked as grey, selected points as blue and the currently processed point as green. As discussed in both this section and Sec. 3.1, the current point is affected by its immediate neighbours as well as the closest selected points.

3.3. Delaunay Triangulation

Once the points in the patch are obtained from the selection process described in Sec. 3.1 and Sec. 3.2, the next step is to triangulate them. Delaunay triangulation operates by selecting triangles such that the minimum angle is maximized, so as to avoid creating thin triangles. The triangulation is applicable to points with and without internal structure, enabling triangulation for any form of point input data. A Delaunay triangulation has a circle associated with each triangle, such that no circle contains points from another triangle. To this end, techniques such as 3D Delaunay triangulation could be applied. However, given the large execution times as demonstrated in [Lo12], it would no longer allow real-time mesh simplification. We make the observation that the terrain datasets are in fact 2.5D and running the expensive 3D Delaunay triangulation could be avoided. Further, the points in a patch are originally picked from DEM and hence no two points have identical 2D coordinates. This implies that we can project the selected points of a patch on the $x-z$ plane dropping their height values, triangulate them using 2D Delaunay method, finally giving the points back their y values, lifting them back to 3D.

With this, the problem is simplified to two dimensions which reduces the complexity of the algorithm while still achieving a good triangulation, see also Fig. 5. This implementation strategy allowed calculation with 2D lines and circles, rather than 3D lines and spheres. Delaunay triangulation of the selected points in each patch is carried out in parallel using multi-core resources, thereby further reducing the overhead.

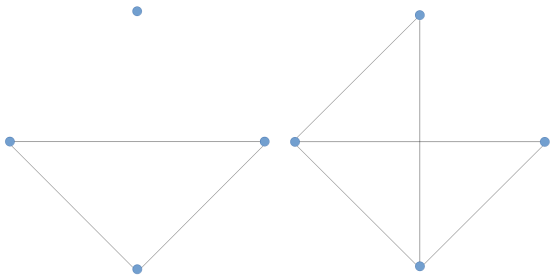


Figure 3: *Identical angles between points that can cause the algorithm to create conflicting triangles.*

A slight downside to the structure of points when creating TINs collected from a DEM was observed in instances where four points formed a square. The angles of the two hypotenuse lines would then be identical. Depending on the

order in which these points were processed, an overlap could occur which is illustrated in Fig. 3. This problem was mitigated by randomly offsetting the 2D points sent to the triangulation, thereby eliminating these identical angles.

In order to obtain a smoother representation at coarse LODs, the unselected neighbours could be used to interpolate the height data of a selected point. Doing so would however require updating to the vertex buffer during the update step, rather than keeping it static with changes localized to the index buffer.

4. System

While the entire pipeline could be carried out in the main thread, multi-threading is employed to make use of the multi-core architecture. The main thread is responsible for visibility checks, handling user input and rendering. A concurrent asynchronous thread is launched that manages the patches that require an update, selecting the points required to approximate the mesh in its current state, Delaunay triangulation and updating the index buffer. With this architecture, the main thread can be kept unblocked and the frame rates are more consistent. The entire procedure is outlined in Alg. 2.

The asynchronous thread starts off by determining which of the patches require new triangulation (line 3), by projecting the largest difference between points onto the screen. The difference value is then compared with the previous value from when the patch was last updated, to calculate the error metric for the patch and to ensure border consistency. If the difference does not exceed the user-defined threshold, the patch is excluded from the update. Only patches that are visible within the camera frustum are handled by the asynchronous thread.

After determining the patches that require an update, the point-selection process for each patch is launched as explained in Sec. 3.1 and Sec. 3.2. The list of patches is also sorted (line 12) to ensure that the patches closest to the camera are updated first. Each thread is given access to a region within the vertex list that is used to triangulate the patches and by sorting which set of boundaries each thread has access to, the order is rearranged.

The varying separation between the selected vertices could result in lost detail in patches with lower resolution, as pixels could receive different normals between updates, resulting in inconsistent lighting. To overcome these lighting issues, the normals are written into a texture and applied to the pixels in the pixel shader. The vastly increased end result can be observed in Fig. 4, where an area further away from the camera is examined more closely. This strategy is similar to normal mapping [TWBO03] but since all vertices have world space coordinates, these coordinates can be utilized to sample the appropriate normal from the texture without needing tangent plane to convert the coordinates to model space.

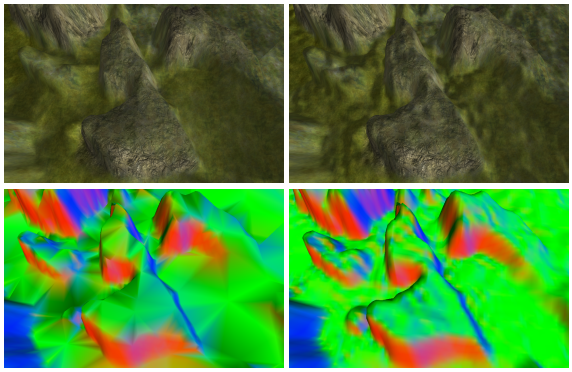


Figure 4: *View of coarsened patches where the top left image shows normals on a vertex level and the top right image shows normals applied on a pixel level. Bottom images display the normals used to apply light to the scene.*

Algorithm 2 Multi-resolution TIN system

```
1:  procedure UPDATEMESH( )
2:      //Camera position when initiating update
3:      determineOutdatedPatches(camPos)
4:      //Set of visible (outdated) patches
5:      P := ∅
6:      for all patches p ∈ outdatedPatchList do
7:          P.append(constructPatch(p, p.pixelError))
8:      //Set of 2D vertices used in the Delaunay step
9:      V := ∅
10:     for all indices i ∈ P do
11:         V.append(vertices[i])
12:     sortByDistance()
13:     for all patches p ∈ sortedList do
14:         //Triangle list
15:         T := ∅
16:         for all vertices v ∈ p do
17:             triangulate(v, V, T, p)
18:         mapToIndexBuffer(T)
```

The triangulation uses the selected 2D points and computes the patches both independently and in parallel. Once the triangulation has been completed for a patch, it is mapped into the corresponding area in the index buffer. The index buffer has a certain amount of elements allocated for each patch to ensure no data is overwritten and that no additional information is required to determine where to write the data. This does however mean that memory is allocated to manage a fully refined patch for every single area, thereby utilizing an abundant allocation scheme which in almost all cases wastes some memory. Although allocating more memory than required, culling, data retrieval and data management in general is simplified with this approach.

The result of the triangulation is cached, since the mesh is often valid for multiple frames (ranging from a few to thousands) depending on the camera movement. No tree structure for the patches was implemented, thereby forcing frustum checks against each patch. This could be improved by keeping a hierarchical data structure, that performs top-down visibility check. On the other hand, the current arrangement enables the algorithm to eliminate preprocessing completely, wherein the patches are adaptively triangulated on-the-fly.

5. RESULTS

The proposed algorithm was implemented in C++ using DirectX 11 and its programmable shader language HLSL. The tests were conducted on a 3.20GHz Intel Xeon processor with a NVIDIA Quadro 4000 graphics card.

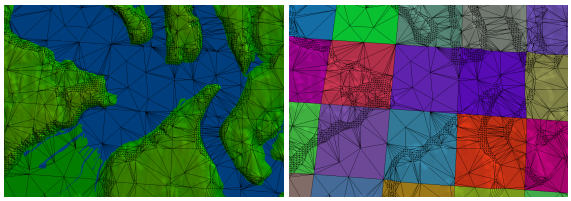


Figure 5: *Terrain mesh simplification achieved using the proposed algorithm in real time, where the triangulation can be seen on top of the rendered terrain (left image) along with patch association (right image).*

In terms of required preprocessing before the technique can be utilized, the presented algorithm requires very little time. Comparing with other TIN algorithms [BG04], [CGG*03], [HT97], [PAL02] that process up to the entire mesh outside of runtime, the presented algorithm only requires the vertex buffer holding the terrain mesh in its original format before it can be launched. The height data can be read from a height map or generated procedurally. Furthermore, no visual artifacts or inconsistencies are produced between different LODs, see also Fig. 5. For each dataset, the presented tests below are all based on the same conditions and the same predefined movement across the terrain, from a ground level perspective. The average frames rates for various dataset with increasing user-specified pixel error are presented in Tab. 1. The variation of the mesh quality with the pixel error is shown in Fig. 1.

A correlation between the number of triangles produced and the time spent on triangulation can be observed when comparing Fig. 6 and Fig. 7, where a lower percentage spent on triangulation corresponds to a lower triangle count. The graphs concern the same tests and are measured over the Puget Sound model, with four different pixel errors. The main aspect that is affected by the number of points is the geometrical calculations in the Delaunay triangulation. Not only is the algorithm forced to process a large amount of points for each triangle in dense patches, scarce patches will require many iterations before the slowly growing Delaunay circle can locate a final point to a triangle. Another reason why the higher error metrics achieve an overall lower triangles per second count, is the linear scalability of the selection process combined with the non-linear scalability of the triangulation process. The selection process still operates on the same data while the triangulation process operates on whatever amount data it receives. Furthermore, a lower amount of triangles are generated for higher error values which reduces the relative time spent triangulation, thereby resulting in a lower triangle per second value.

	Size of dataset		
Pixel error	256 × 256	512 × 512	1024 × 1024
0.25	3765	2657	1264
0.5	3938	2818	1232
1.0	3953	2868	1290
2.0	4035	2904	1378

Table 1: *The average frames per second (fps) for various dataset sizes as a function of the user-defined pixel error.*

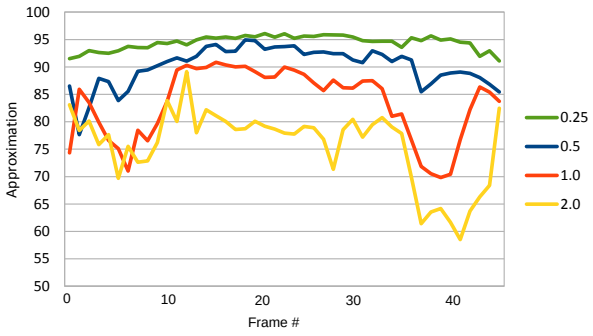


Figure 6: *The percentage of the algorithm that is spent on triangulation across the simulation with different error metrics.*

The dimensions of the patches also affect the performance of the algorithm, where patches that are too large process too many points and patches that are too small increase the overhead. As shown in Tab. 2, the most suited patch dimension is 32 × 32 and was therefore used in previous tests. When determining the suitable patch dimension, the tests were conducted on the Puget Sound model with a pixel error of 0.5.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a novel LOD-based approach for TIN simplification that is capable of generating and stitching various resolutions at runtime. Our approach can employ simplification by leveraging 2D Delaunay triangulation while simultaneously reducing triangle count by

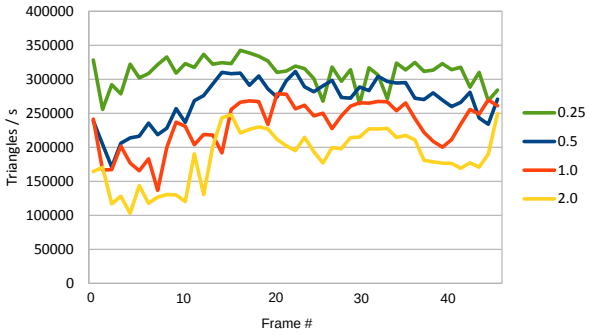


Figure 7: *The performance across the simulation with different error measurements.*

Patch dim	Max	Min	Average
8 × 8	3.45	0.41	1.94
16 × 16	1.57	0.43	0.90
32 × 32	1.19	0.19	0.68
64 × 64	2.45	0.25	1.01

Table 2: *The maximum, minimum and average number of seconds that the algorithm took to complete one update.*

examining the features based on a user-defined error metric. The technique uses multi-core resources to process selection of vertices representing the terrain patches and to triangulate them in parallel.

In terms of future work, there are a few aspects of which to focus on, such as delegating the entire triangulation to the GPU and adding a postprocessing step to improve the quality of the triangulation. Our algorithm was designed around parallelism so triangulating on the GPU is an interesting extension to investigate. If a more complex approximation technique is used, then focus should still revolve around fast processing, without sacrificing the approximation quality.

References

[AW03] AMARATUNGA K., WU J.: Wavelet triangulated irregular networks. *International Journal of Geographical Information Science 17*, 3 (2003), 273 – 289. 2

[BG04] BARTHOLDI J. J. I., GOLDSMAN P.: Multiresolution indexing of triangulated irregular networks. *IEEE Transactions on Visualization and Computer Graphics 10*, 4 (2004), 484 – 495. 2, 3, 6

[BGP09] BÖSCH J., GOSWAMI P., PAJAROLA R.: RASTeR: Simple and efficient terrain rendering on the GPU. In *Proceedings EUROGRAPHICS Areas Papers, Scientific Visulization* (2009), pp. 35 – 42. 2

[CGG*03] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Bdam - batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum 22*, 3 (2003), 505 – 514. 2, 3, 6

[COL96] COHEN-OR D., LEVANONI Y.: Temporal continuity of levels of detail in delaunay triangulated terrain. pp. 37 – 42. 2

[CPS97] CIGNONI P., PUPPO E., SCOPIGNO R.: Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer 13*, 5 (1997), 199 – 217. 2

[DFP95] DE FLORIANI L., PUPPO E.: Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics (TOG) 14*, 4 (1995), 363 – 411. 2

[DWS*97] DUCHAINEAU M., WOLINSKY M., SIGETI D., MILLER M., ALDRICH C., MINEEV-WEINSTEIN M.: Roaming terrain: real-time optimally adapting meshes. IEEE Computer Society Press, pp. 81 – 88. 2

[EKT01] EVANS W., KIRKPATRICK D., TOWNSEND G.: Right-triangulated irregular networks. *Algorithmica 30*, 2 (2001), 264 – 286. 2

[GGS95] GROSS M. H., GATTI R., STAADT O.: Fast multiresolution surface meshing. pp. 135 – 142. 2

[GH99] GUMHOLD S., HÃIJTTNER T.: Multiresolution rendering with displacement mapping. ACM, pp. 55 – 66. 2

[GMBP10] GOSWAMI P., MAKHINYA M., BÖSCH J., PAJAROLA R.: Scalable parallel out-of-core terrain rendering. In *Proceedings Eurographics Symposium on Parallel Graphics and Visualization* (2010), pp. 63–71. 2

[Hop98] HOPPE H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. IEEE Computer Society Press, pp. 35 – 42. 2

[HT97] HUANG S.-J., TSENG D.-C.: Construction of multiresolution terrain models using hierarchical delaunay triangulated irregular networks. vol. 4, pp. 1999 – 2001 vol.4. 2, 6

[KK02] KOLINGEROVÃĄ I., KOHOUT J.: Optimistic parallel delaunay triangulation. *The Visual Computer 18*, 8 (2002), 511 – 529. 3

[LH04] LOSASSO F., HOPPE H.: Geometry clipmaps: terrain rendering using nested regular grids. *ACM Transactions on Graphics (TOG) 23*, 3 (2004), 769 – 776. 2, 3

[LKES09] LIVNY Y., KOGAN Z., EL-SANA J.: Seamless patches for gpu-based terrain rendering. *The Visual Computer 25*, 3 (2009), 197 – 208. 2, 3

[LKR*96] LINDSTROM P., KOLLER D., RIBARSKY W., HODGES L., FAUST N., TURNER G.: Real-time, continuous level of detail rendering of height fields. ACM, pp. 109 – 118. 2

[Lo12] LO S. H.: Parallel delaunay triangulation in three dimensions. *Computer Methods in Applied Mechanics and Engineering 237-240* (2012), 88 – 106. 3, 4

[LP02] LINDSTROM P., PASCUCCI V.: Terrain simplification simplified: a general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 239 – 254. 2

[Mus97] MUSIN O.: Properties of the delaunay triangulation. ACM, pp. 424 – 426. 2

[PAL02] PAJAROLA R., ANTONIJUAN M., LARIO R.: Quadtin: quadtree based triangulated irregular networks. IEEE Computer Society, pp. 395 – 402. 2, 6

[PG07] PAJAROLA R., GOBBETTI E.: Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer 23*, 8 (2007), 583 – 605. 1, 2

[SS09] SCHWARZ M., STAMMINGER M.: Fast gpu-based adaptive tessellation with cuda. *Computer Graphics Forum 28*, 2 (2009), 365 – 374. 2

[TGM*06] TANG T., GONG H., MO Y., DUAN F., ZHAO W.: Dynamic data retrieval and distance decay of triangulated irregular network(tin) in three dimensional visualizations. *Geographic Information Sciences 12*, 1 (2006), 21. 2

[TWBO03] TASDIZEN T., WHITAKER R., BURCHARD P., OSHER S.: Geometric surface processing via normal maps. *ACM Transactions on Graphics (TOG) 22*, 4 (2003), 1012 – 1033. 5

[XV96] XIA J. C., VARSHNEY A.: Dynamic view-dependent simplification for polygonal models. pp. 327 – 334. 2

[YLS05] YANG B., LI Q., SHI W.: Constructing multi-resolution triangulated irregular network model for visualization. *Computers and Geosciences 31*, 1 (2005), 77 – 86. 2

SIGRAD 2016
M. Hayashi (Editors)

A Radial Basis Function Approximation for Large Datasets

Z. Majdisova[1] and V. Skala[1]

[1]Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Univerzitni 8, CZ 30614 Plzen, Czech Republic

Abstract

Approximation of scattered data is often a task in many engineering problems. The Radial Basis Function (RBF) approximation is appropriate for large scattered datasets in d-dimensional space. It is non-separable approximation, as it is based on a distance between two points. This method leads to a solution of overdetermined linear system of equations.

In this paper a new approach to the RBF approximation of large datasets is introduced and experimental results for different real datasets and different RBFs are presented with respect to the accuracy of computation. The proposed approach uses symmetry of matrix and partitioning matrix into blocks.

Categories and Subject Descriptors (according to ACM CCS): G.1.2 [Numerical Analysis]: Approximation—Approximation of Surfaces and Contours

1. Introduction

Interpolation and approximation are the most frequent operations used in computational techniques. Several techniques have been developed for data interpolation or approximation, but they mostly expect an ordered dataset, e.g. rectangular mesh, structured mesh, unstructured mesh etc. However, in many engineering problems, data are not ordered and they are scattered in $d-$dimensional space, in general. Usually, in technical applications the conversion of a scattered dataset to a semi-regular grid is performed using some tessellation techniques. However, this approach is quite prohibitive for the case of $d-$dimensional data due to the computational cost.

Interesting techniques are based on the Radial Basis Function (RBF) method which was originally introduced by [Har71]. They are widely used across of many fields solving technical and non-technical problems. The RBF applications can be found in neural networks, data visualization [PRF14], surface reconstruction [CBC*01], [TO02], [PS11], [SPN13], [SPN14], solving partial differential equations [LCC13], [HSfY15], etc. The RBF techniques are really meshless and are based on collocation in a set of scattered nodes. These methods are independent with respect to the dimension of the space. The computational cost of this techniques increase nonlinearly with the number of points in the given dataset and linearly with the dimensionality of data.

There are two main groups of basis functions: global RBFs and Compactly Supported RBFs (CS-RBFs) [Wen06]. Fitting scattered data with CS-RBFs leads to a simpler and faster computation, but techniques using CS-RBFs are sensitive to the density of scattered data. Global RBFs lead to a linear system of equations with a dense matrix and their usage is based on sophisticated techniques such as the fast multipole method [Dar00]. Global RBFs are useful in repairing incomplete datasets and they are insensitive to the density of scattered data.

For the processing of scattered data we can use the RBF interpolation or the RBF approximation. The RBF interpolation, e.g. presented by [Ska15], is based on a solution of a linear system of equations:

$$\mathbf{Ac} = \mathbf{h}, \tag{1}$$

where $\mathbf{A}$ is a matrix of this system, $\mathbf{c}$ is a column vector of variables and $\mathbf{h}$ is a column vector containing the right sides of equations. In this case, $\mathbf{A}$ is an $N \times N$ matrix, where N is the number of points in the given scattered dataset, the variables are weights for basis functions and the right sides of equations are values in the given points. The disadvantage of RBF interpolation is the large and usually ill-conditioned matrix of the linear system of equations. Moreover, in the case of an oversampled dataset or intended reduction, we want to reduce the given problem, i.e. reduce the number of weights and used basis functions, and preserve good preci-

sion of the approximated solution. The approach which includes the reduction is called the RBF approximation. In the following section, the method recently introduced in [Ska13] is described in detail. This approach requires less memory and offer higher speed of computation than the method using Lagrange multipliers [Fas07]. Further, a new approach to RBF approximation of large datasets is presented in the Section 3. These approach uses symmetry of matrix and partitioning matrix into blocks.

2. RBF Approximation

For simplicity, we assume that we have an unordered dataset $\{\mathbf{x}_i\}_1^N \in E^2$. However, this approach is generally applicable for d-dimensional space. Further, each point $\mathbf{x}_i$ from the dataset is associated with a vector $\mathbf{h}_i \in E^p$ of the given values, where p is the dimension of the vector, or scalar value, i.e. $h_i \in E^1$. For an explanation of the RBF approximation, let us consider the case when each point $\mathbf{x}_i$ is associated with a scalar value h_i, e.g. a $2^1/_2D$ surface. Let us introduce a set of new reference points $\{\boldsymbol{\xi}_j\}_1^M$, see Figure 1.

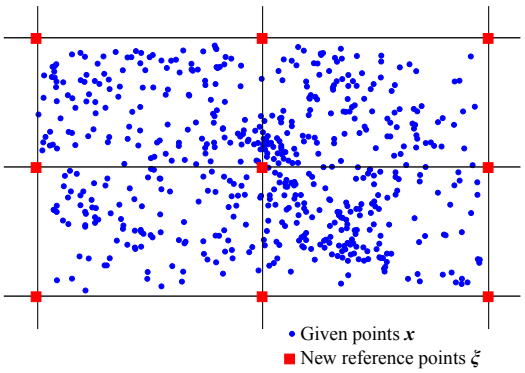


Figure 1: *The RBF approximation and reduction of points.*

These reference points may not necessarily be in a uniform grid. It is appropriate that their placement reflects the given surface (e.g. the terrain profile, etc.) as well as possible. The number of reference points $\boldsymbol{\xi}_j$ is M, where $M \ll N$. Now, the RBF approximation is based on the distance computation of the given point $\mathbf{x}_i$ and the reference point $\boldsymbol{\xi}_j$.

The approximated value is determined similarly as for interpolation (see [Ska15]):

$$f(\mathbf{x}) = \sum_{j=1}^{M} c_j \phi(r_j) = \sum_{j=1}^{M} c_j \phi(\|\mathbf{x} - \boldsymbol{\xi}_j\|), \tag{2}$$

where $\phi(r_j)$ is a used RBF centered at point $\boldsymbol{\xi}_j$ and the approximating function $f(\mathbf{x})$ is represented as a sum of these RBFs, each associated with a different reference point $\boldsymbol{\xi}_j$, and weighted by a coefficient c_j which has to be determined.

It can be seen that we get an overdetermined linear system of equations for the given dataset:

$$\begin{aligned} h_i = f(\mathbf{x}_i) &= \sum_{j=1}^{M} c_j \phi(\|\mathbf{x}_i - \boldsymbol{\xi}_j\|) \\ &= \sum_{j=1}^{M} c_j \phi_{i,j} \qquad i = 1, \ldots, N. \end{aligned} \tag{3}$$

The linear system of equations (3) can be represented in a matrix form as:

$$\mathbf{A}\mathbf{c} = \mathbf{h}, \tag{4}$$

where the number of rows is $N \gg M$ and M is the number of unknown weights $[c_1, \ldots, c_M]^T$, i.e. the number of reference points. Equation (4) represents system of linear equations:

$$\begin{pmatrix} \phi_{1,1} & \cdots & \phi_{1,M} \\ \vdots & \ddots & \vdots \\ \phi_{i,1} & \cdots & \phi_{i,M} \\ \vdots & \ddots & \vdots \\ \phi_{N,1} & \cdots & \phi_{N,M} \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_M \end{pmatrix} = \begin{pmatrix} h_1 \\ \vdots \\ h_i \\ \vdots \\ h_N \end{pmatrix}. \tag{5}$$

The presented system is overdetermined, i.e. the number of equations N is higher than the number of variables M. This linear system of equations can be solved by the least squares method as $\mathbf{A}^T\mathbf{A}\mathbf{c} = \mathbf{A}^T\mathbf{h}$ or singular value decomposition, etc.

3. RBF Approximation for Large Data

In practice, the real datasets contain a large number of points which results into high memory requirements for storing the matrix $\mathbf{A}$ of the overdetermined linear system of equations (5). For example when we have dataset contains $3,000,000$ points, number of reference points is $10,000$ and double precision floating point is used then we need 223.5 GB memory for storing the matrix $\mathbf{A}$ of the overdetermined linear system of equations (5). Unfortunately, we do not have an unlimited capacity of RAM memory and therefore calculation of unknown weights c_j for RBF approximation would be prohibitively computationally expensive due to memory swapping, etc. In this section, a proposed solution to this problem is described.

In Section 2, it was introduced that overdetermined system of equations can be solved by the least squares method. For this method the $M \times M$ square matrix:

$$\mathbf{B} = \mathbf{A}^T\mathbf{A} \tag{6}$$

is to be determined. Advantages of matrix $\mathbf{B}$ are that it is a symmetric matrix and moreover only two vectors of length N are needed te determine of one entry, i.e.:

$$b_{ij} = \sum_{k=1}^{N} \phi_{ki} \cdot \phi_{kj}, \tag{7}$$

where b_{ij} is the entry of the matrix $\mathbf{B}$ in the $i-$th row and $j-$th column.

To save memory requirements and data bus (PCI) load block operations with matrices are used. Based on the above properties of the matrix $\mathbf{B}$, only the upper triangle of this matrix is computed. Moreover the matrix is partitioned into $M_B \times M_B$ blocks, see Figure 2, and the calculation is performed sequentially for each block:

$$\mathbf{B}_{kl} = (\mathbf{A}_{*,k})^T (\mathbf{A}_{*,l})$$
$$k = 1, \ldots, \frac{M}{M_B}, \qquad l = k, \ldots, \frac{M}{M_B}, \tag{8}$$

where $\mathbf{B}_{kl}$ is sub-matrix in the $k-$th row and $l-$th column and $\mathbf{A}_{*,k}$ is defined as:

$$\mathbf{A}_{*,k} = \begin{pmatrix} \phi_{1,(k-1)\cdot M_B+1} & \cdots & \phi_{1,k\cdot M_B} \\ \vdots & \ddots & \vdots \\ \phi_{i,(k-1)\cdot M_B+1} & \cdots & \phi_{i,k\cdot M_B} \\ \vdots & \ddots & \vdots \\ \phi_{N,(k-1)\cdot M_B+1} & \cdots & \phi_{N,k\cdot M_B} \end{pmatrix}. \tag{9}$$

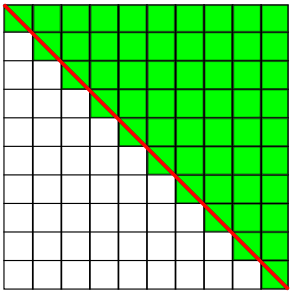


Figure 2: *$M \times M$ square matrix which is partitioned into $M_B \times M_B$ blocks. Main diagonal of matrix is represented by red color and illustrates the symmetry of matrix. Blocks, which must be computed, are represented by green color.*

The size of block M_B is chosen so that M_B is multiple of M and there is no swapping, i.e.:

$$M_B \cdot (M_B + 2 \cdot N) \cdot prec < \text{size of RAM [B]}, \tag{10}$$

where *prec* is size of data type in bytes.

4. Experimental results

The presented modification of the RBF approximation method has been tested on synthetic and real data. Let us introduce results for two real datasets.

The first dataset was obtained from LiDAR data of the Serpent Mound in Adams Country, Ohio[†]. The second dataset is LiDAR data of the Mount Saint Helens in Skamania Country, Washington[†]. Each point of these datasets is associated with its elevation. Summary of the dimensions of terrain for the given datasets is in Table 1.

Table 1: *Summary of the dimensions of terrain for tested datasets. Note that one feet [ft] corresponds to 0.3048 meter [m].*

Dimensions	Serpent Mound	St. Helens
number of points	3,265,110	6,743,176
lowest point [ft]	166.7800	3,191.5269
highest point [ft]	215.4800	8,330.2219
width [ft]	1,085.1199	26,232.3696
length [ft]	2,698.9601	35,992.6861

For experiments, two different radial basis functions have been used, see Table 2. Shape parameters α for used RBFs were determined experimentally with regard to the quality of approximation and they are presented in Table 3. Note that value of shape parameter α is inversely proportional to range of datasets.

Table 2: *Used RBFs*

RBF	type	$\phi(\boldsymbol{r})$
Gaussian RBF	global	$e^{-(\alpha r)^2}$
Wendland's $\phi_{3,1}$	local	$(1-\alpha r)_+^4(4\alpha r+1)$

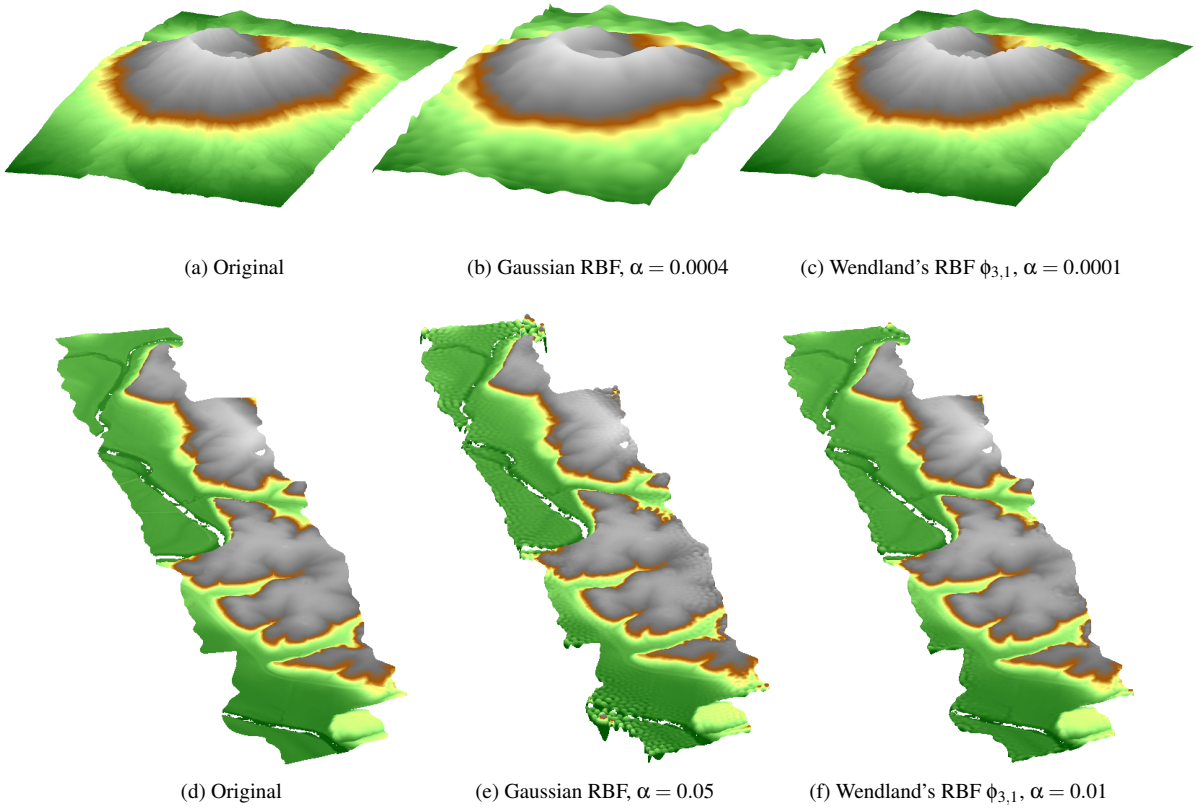
Table 3: *Experimentally determined shape parameters α for used RBFs*

RBF	shape parameter	
	Serpent Mound	**St. Helens**
Gaussian RBF	$\alpha = 0.05$	$\alpha = 0.0004$
Wendland's $\phi_{3,1}$	$\alpha = 0.01$	$\alpha = 0.0001$

The set of reference points equals the subset of the given dataset for which we determine the RBF approximation. Moreover, the distribution of reference points is uniform and the set of reference points has a cardinality 10,000 in both experiments.

Approximation of Mount Saint Helens for both BRFs and its original are shown in Figure 3a-3c. In Figure 3b can be seen that the RBF approximation with the global Gaussian RBFs cannot preserve the sharp rim of a crater. Further, visualization of magnitude of error at each point of the original points cloud is presented in Figure 4 and Figure 5. It can be seen that the RBF approximation with the global Gaussian

[†] http://www.liblas.org/samples/



(a) Original (b) Gaussian RBF, $\alpha = 0.0004$ (c) Wendland's RBF $\phi_{3,1}$, $\alpha = 0.0001$

(d) Original (e) Gaussian RBF, $\alpha = 0.05$ (f) Wendland's RBF $\phi_{3,1}$, $\alpha = 0.01$

Figure 3: *Serpent Mound in Adams Country, Ohio (top) and Mount Saint Helens is Skamania Country, Washington (bottom)*

RBFs returns worse result than RBF approximation with local Wendland's $\phi_{3,1}$ basis functions in terms of the error. In Table 4 can be seen the value of mean absolute error, its deviation and mean relative error for both approximations.

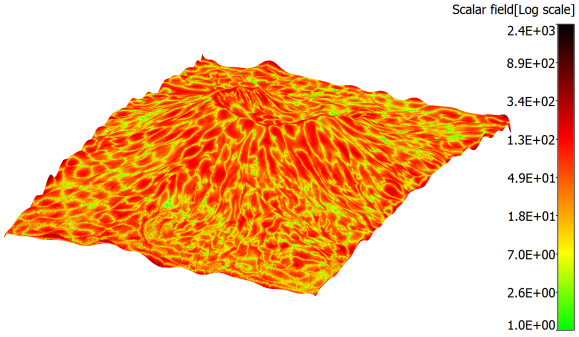


Figure 4: *Approximation of Mount Saint Helens with* $10,000$ *global Gaussian basis functions with shape parameter* $\alpha = 0.0004$ *false-colored by magnitude of error.*

Results of the RBF approximation for Serpent Mound and its original are shown in Figure 3d-3f. It can be seen that

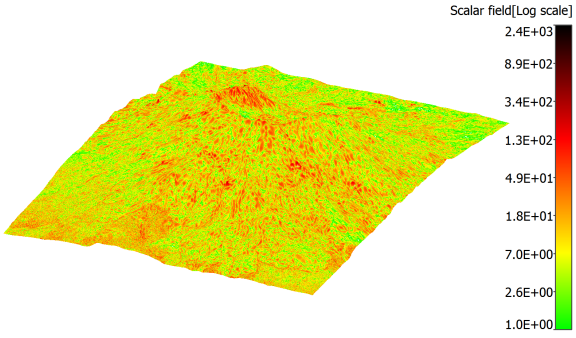


Figure 5: *Approximation of Mount Saint Helens with* $10,000$ *local Wendland's* $\phi_{3,1}$ *basis functions with shape parameter* $\alpha = 0.0001$ *false-colored by magnitude of error.*

the approximation using local Wendland's $\phi_{3,1}$ basis function (Figure 3f) returns again better result than approximation using the global Gaussian RBF (Figure 3e) in terms of the error. It is also seen in Figure 6 and Figure 7 where magnitude of error at each point of original points cloud is visualized. Moreover, we can see that the highest errors occur

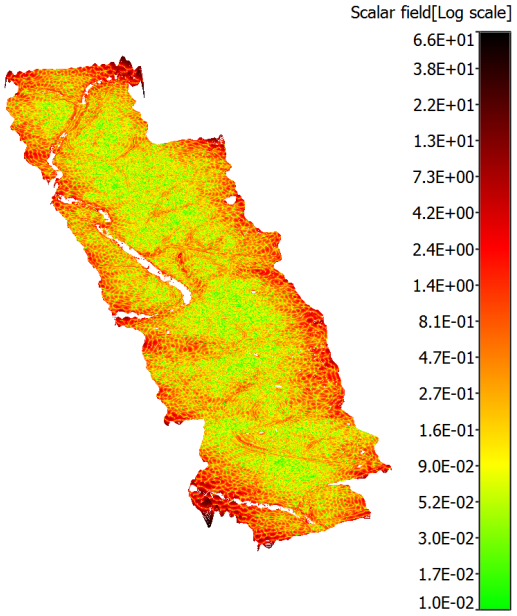


Figure 6: *Approximation of the Serpent Mound with* $10,000$ *global Gaussian basis functions with shape parameter* $\alpha = 0.05$ *false-colored by magnitude of error.*

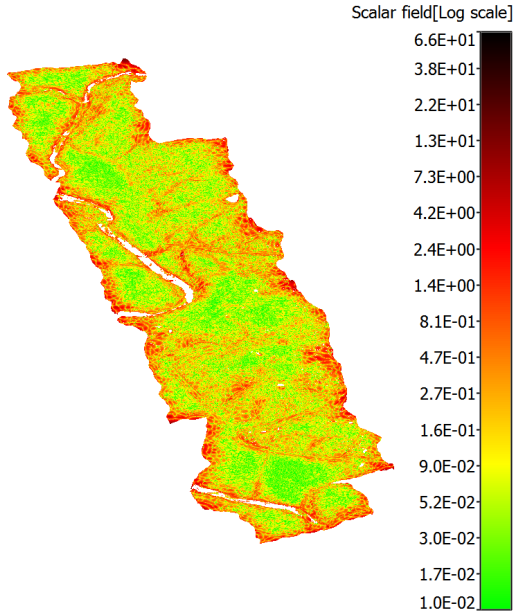


Figure 7: *Approximation of the Serpent Mound with* $10,000$ *local Wendland's* $\phi_{3,1}$ *basis functions with shape parameter* $\alpha = 0.01$ *false-colored by magnitude of error.*

on the boundary of terrain, which is a general problem of RBF methods. Value of mean absolute error, its deviation and mean relative error due to elevation for both used RBFs are again mentioned in Table 4.

Mutual comparison both datasets in terms of the mean relative error (Table 4) indicates that mean relative error for Serpent Mount is smaller than for Mount Saint Helens. It is caused by the presence of vegetation, namely forest, in LiDAR data of the Mount Saint Helens. This vegetation operates in our RBF approximation as noise and therefore the resulting mean relative error is higher.

The implementation of the RBF approximation has been performed in Matlab and tested on PC with the following configuration:

- CPU: Intel® Core™ i7-4770 (4× 3.40GHz + hyper-threading),
- memory: 32 GB RAM,
- operating system Microsoft Windows 7 64bits.

For the approximation of the Serpent Mound with $10,000$ local Wendland's $\phi_{3,1}$ basis function with shape parameter $\alpha = 0.01$ the running times for different sizes of blocks were measured. These times were converted relative to the time for 100×100 blocks and are presented in Figure 8. We can see that for the approximation matrix which is partitioned into small blocks (i.e. smaller than 25×25 blocks) the time performance is large. This is caused by overhead costs. On the other hand, for the approximation matrix which is partitioned into large blocks (i.e. larger than 125×125 blocks) the running time begins to grow above the permissible limit due to memory swapping.

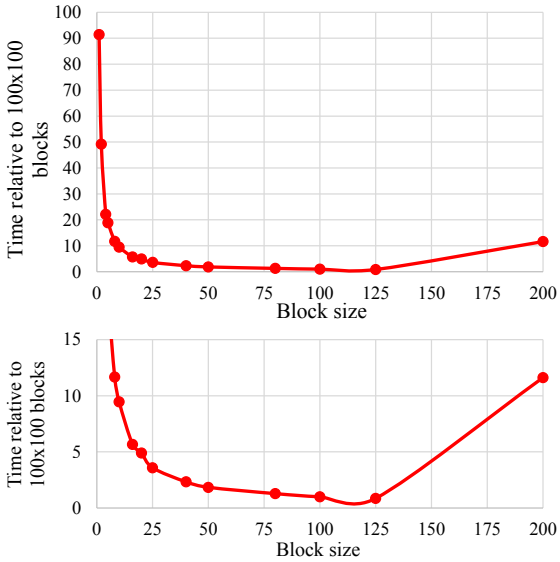


Figure 8: *Time performance for approximation of the Serpent Mound depending on the block size. The times are presented relative to the time for* 100×100 *blocks.*

Table 4: *The RBF approximation error for testing datasets and different radial basis functions. Note that one feet [ft] corresponds to 0.3048 meter [m].*

Error	Serpent Mound		St. Helens	
	Gaussian RBF	**Wendland's $\phi_{3,1}$**	**Gaussian RBF**	**Wendland's $\phi_{3,1}$**
mean absolute error [ft]	0.4477	0.2289	44.4956	12.1834
deviation of error [ft]	1.4670	0.1943	680.3659	169.2800
mean relative error [%]	0.0024	0.0012	0.0087	0.0023

5. Conclusions

This paper presents a new approach to the RBF approximation of large datasets. The proposed approach uses symmetry of matrix and partitioning matrix into blocks, thus preventing memory swapping. The experiments made proved that the proposed approach is able to determine the RBF approximation for large dataset. Moreover, from the experimental results we can see that use of a local RBFs is better than global RBFs, if data are sufficiently sampled. Futher, it is obvious that approximation using the global Gaussian RBFs has problems with the preservation of sharp edges. The experiments made also proved that RBF methods have problems with the accuracy of calculation on the boundary of an object, which is a well known property, and the magnitude of the RBF approximation error is influenced by the presence of a noise.

For the future work, the RBF approximation method can be explored in terms of lower sensitivity to noise, more accurate calculation on the boundary or better approximation of sharp edges and improvements of the computational cost without loss of approximation accuracy.

Acknowledgments

The authors would like to thank their colleagues at the University of West Bohemia, Plzen, for their discussions and suggestions, and also anonymous reviewers for the valuable comments and suggestions they provided. The research was supported by MSMT CR projects LH12181 and SGS 2016-013.

References

[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, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001* (2001), pp. 67–76. 1

[Dar00] DARVE E.: The fast multipole method: Numerical implementation. *Journal of Computational Physics 160*, 1 (2000), 195–240. 1

[Fas07] FASSHAUER G. E.: *Meshfree Approximation Methods with MATLAB*, vol. 6. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007. 2

[Har71] HARDY R. L.: Multiquadratic Equations of Topography and Other Irregular Surfaces. *Journal of Geophysical Research 76* (1971), 1905–1915. 1

[HSfY15] HON Y.-C., SARLER B., FANG YUN D.: Local radial basis function collocation method for solving thermo-driven fluid-flow problems with free surface. *Engineering Analysis with Boundary Elements 57* (2015), 2 – 8. {RBF} Collocation Methods. 1

[LCC13] LI M., CHEN W., CHEN C.: The localized {RBFs} collocation methods for solving high dimensional {PDEs}. *Engineering Analysis with Boundary Elements 37*, 10 (2013), 1300 – 1304. 1

[PRF14] PEPPER D. W., RASMUSSEN C., FYDA D.: A meshless method using global radial basis functions for creating 3-d wind fields from sparse meteorological data. *Computer Assisted Methods in Engineering and Science 21*, 3-4 (2014), 233–243. 1

[PS11] PAN R., SKALA V.: A two-level approach to implicit surface modeling with compactly supported radial basis functions. *Eng. Comput. (Lond.) 27*, 3 (2011), 299–307. 1

[Ska13] SKALA V.: Fast Interpolation and Approximation of Scattered Multidimensional and Dynamic Data Using Radial Basis Functions. *WSEAS Transactions on Mathematics 12*, 5 (2013), 501–511. 2

[Ska15] SKALA V.: Meshless interpolations for computer graphics, visualization and games. In *Eurographics 2015 - Tutorials, Zurich, Switzerland, May 4-8, 2015* (2015), Zwicker M., Soler C., (Eds.), Eurographics Association. 1, 2

[SPN13] SKALA V., PAN R., NEDVED O.: Simple 3d surface reconstruction using flatbed scanner and 3d print. In *SIGGRAPH Asia 2013, Hong Kong, China, November 19-22, 2013, Poster Proceedings* (2013), ACM, p. 7. 1

[SPN14] SKALA V., PAN R., NEDVED O.: Making 3d replicas using a flatbed scanner and a 3d printer. In *Computational Science and Its Applications - ICCSA 2014 - 14th International Conference, Guimarães, Portugal, June 30 - July 3, 2014, Proceedings, Part VI* (2014), vol. 8584 of *Lecture Notes in Computer Science*, Springer, pp. 76–86. 1

[TO02] TURK G., O'BRIEN J. F.: Modelling with implicit surfaces that interpolate. *ACM Trans. Graph. 21*, 4 (2002), 855–873. 1

[Wen06] WENDLAND H.: Computational aspects of radial basis function approximation. *Studies in Computational Mathematics 12* (2006), 231–256. 1

SIGRAD 2016
M. Hayashi (Editors)

Vector Field Interpolation with Radial Basis Functions

M. Smolik[1] and V. Skala[1]

[1]Faculty of Applied Sciences, University of West Bohemia, Plzen, Czech Republic

Abstract

This paper presents a new approach for the Radial Basis Function (RBF) interpolation of a vector field. Standard approaches for interpolation randomly select points for interpolation. Our approach uses the knowledge of vector field topology and selects points for interpolation according to the critical points location. We presents the results of interpolation errors on a vector field generated from an analytical function.

Categories and Subject Descriptors (according to ACM CCS): G.1.1 [Numerical analysis]: Interpolation—Interpolation formulas

1. Introduction

Interpolation is probably the most frequent operation used in computational methods. Several methods have been developed for data interpolation, but they expect some kind of data "ordering", e.g. structured mesh, rectangular mesh, unstructured mesh, etc. However, in many engineering problems, data are not ordered and they are scattered in $d-$dimensional space, in general. Usually, in technical applications, the scattered data are tessellated using triangulation but this approach is quite prohibitive for the case of $d-$dimensional data interpolation because of the computational cost.

Interpolating scattered vector data on a surface becomes frequent in applied problem solutions. There are applications for vector field decomposition [EJF09], for vector field design system for surfaces that allows the user to control the number of singularities in the vector field and their placement [ZMT06]. [MZT*14] uses the vector field interpolation for estimating robust point correspondences between two sets of points.

2. Vector Field

Vector fields on surfaces are important objects, which appear frequently in scientific simulation in CFD (Computational Fluid Dynamics) or modeling by FEM (Finite Element Method). To be visualized, such vector fields are usually linearly approximated for the sake of simplicity and performance considerations.

The vector field can be easily analyzed when having an approximation of the vector field near some location point. The important places to be analyzed are so called critical points. Analyzing the vector field behavior near these points gives us the information about the characteristic of the vector field.

2.1. Critical Point

Critical points $\boldsymbol{x}_0$ of the vector field are points at which the magnitude of the vector vanishes

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{0}, \tag{1}$$

i.e. all components are equal to zero

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{2}$$

A critical point is said to be isolated, or simple, if the vector field is non vanishing in an open neighborhood around the critical point. Thus for all surrounding points $\boldsymbol{x}_\varepsilon$ of the critical point $\boldsymbol{x}_0$ the equation (1) does not apply, i.e.

$$\frac{d\boldsymbol{x}_\varepsilon}{dt} \neq \boldsymbol{0}, \tag{3}$$

At critical points, the direction of the field line is indeterminate, and they are the only points in the vector field were field lines can intersect (asymptotically). The terms singular point, null point, neutral point or equilibrium point are also frequently used to describe critical points.

These points are important because together with the

nearby surrounding vectors, they have more information encoded in them than any such group in the vector field, regarding the total behavior of the field.

2.2. Linearization of Vector Field

Critical points can be characterized according to the behavior of nearby tangent curves. We can use a particular set of these curves to define a skeleton that characterizes the global behavior of all other tangent curves in the vector field. An important feature of differential equations is that it is often possible to determine the local stability of a critical point by approximating the system by a linear system. These approximations are aimed at studying the local behavior of a system, where the nonlinear effects are expected to be small. To locally approximate a system, the Taylor series expansion must be utilized locally to find the relation between $\boldsymbol{v}$ and position $\boldsymbol{x}$, supposing the flow $\boldsymbol{v}$ to be sufficiently smooth and differentiable. In such case, the expansion of $\boldsymbol{v}$ around the critical points $\boldsymbol{x}_0$ is

$$\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{v}(\boldsymbol{x}_0) + \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}_0). \tag{4}$$

As $\boldsymbol{v}(\boldsymbol{x}_0)$ is according to (1) equal zero for critical points, we can rewrite equation (4) using matrix notation

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} \tag{5}$$

$$\boldsymbol{v} = \boldsymbol{J} \cdot (\boldsymbol{x} - \boldsymbol{x}_0), \tag{6}$$

where $\boldsymbol{J}$ is called Jacobian matrix and characterizes the vector field behavior around a critical point $\boldsymbol{x}_0$.

2.3. Classification of Critical Points

There exist a finite set of fundamentally different critical points, defined by the number of inflow and outflow directions, spiraling structures etc., and combinations of these. Since the set is finite, each critical point can be classified. Such a classification defines the field completely in a close neighborhood around the critical point. By knowing the location and classification of critical points in a vector field, the topology of the field is known in small areas around these. Assuming a smooth transition between these areas, one can construct a simplified model of the whole vector field. Such a simplified representation is useful, for instance, in compressing vector field data into simpler building blocks [PS97].

The critical points are classified based on the vector field around these points. The information derived from the classification of critical points aids the information selection process when it comes to visualizing the field. By choosing seed points for field lines based on the topology of critical points, field lines encoding important information is ensured.

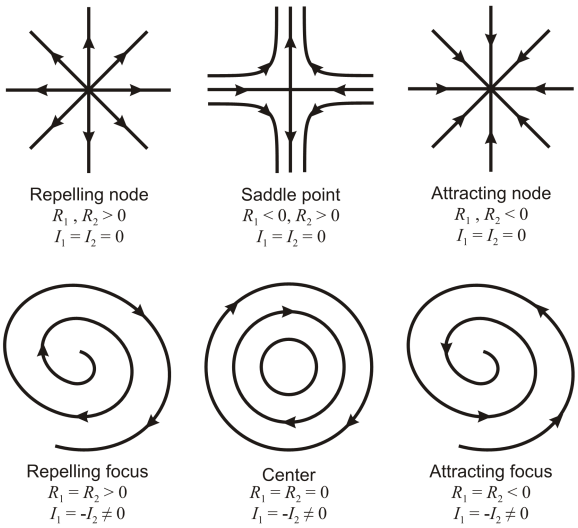


Figure 1: Classification of 2D first order critical points. R_1, R_2 denote the real parts of the eigenvalues of the Jacobian matrix while I_1, I_2 denote their imaginary parts (from [HH89]).

A more advanced approach is to connect critical points, and use the connecting lines and surfaces to separate areas of different flow topology [HH89], [WTS*05].

The fact that a linear model can be used to study the behavior of a nonlinear system near a critical point is a powerful one [HH89]. We can use the Jacobian matrix to characterize the vector field and the behavior of nearby tangent curves, for nondegenerate critical point.

The eigenvalues and the eigenvectors of Jacobian matrix are very important for vector field classification and description, see Figure 1. A real eigenvector of the Jacobian matrix defines a direction such that if we move slightly from the critical point in that direction, the field is parallel to the direction we moved. Thus, at the critical point, the real eigenvectors are tangent to the trajectories that end on the point. The sign of the corresponding eigenvalue determines whether the trajectory is outgoing (repelling) or incoming (attracting) at the critical point. The imaginary part of an eigenvalue denotes circulation about the point.

3. Radial Basis Functions

The Radial basis functions (RBF) is a technique for scattered data interpolation [PS11] and approximation [Fas07], [Ska15]. The RBF interpolation and approximation is computationally more expensive, because input data are not ordered and there is no known relation between them. Although the RBF has higher computational cost, it can be used for d-dimensional problem solution in many applications, e.g. solution of partial differential equations, image re-

construction, neural networks, fuzzy systems, GIS systems, optics etc.

The RBF is a function whose value depends only on the distance from some center point. Due to the use of the distance functions, the RBFs can be easily implemented to reconstruct the surface using scattered data in 2D, 3D or higher dimensional spaces. It should be noted that the RBF interpolation is not separable.

Radial function interpolants have a nice property of being invariant under all Euclidean transformations, i.e. translations, rotations and reflections. It means that it does not matter whether we first compute the RBF interpolation function and then apply a Euclidean transformation, or if we first transform all the data and then compute the radial function interpolants. This is result of the fact that Euclidean transformations are characterized by orthogonal transformation matrices and are therefore 2 norm invariant. Radial basis functions can be divided into two groups according to their influence. First group are "global" RBF [Sch79], for example:

$$\begin{aligned}
&\text{Thin Plate Spline (TPS)} && \varphi(r) = r^2 \log r \\
&\text{Gauss function} && \varphi(r) = e^{-(\varepsilon r)^2} \\
&\text{Inverse Quadric (IQ)} && \varphi(r) = \frac{1}{1+(\varepsilon r)^2} \\
&\text{Inverse Multiquadric (IMQ)} && \varphi(r) = \frac{1}{\sqrt{1+(\varepsilon r)^2}} \\
&\text{Multiquadric (MQ)} && \varphi(r) = \sqrt{1+(\varepsilon r)^2}
\end{aligned} \tag{7}$$

where ε is the shape parameter of radial basis function [FP08].

The "local" RBF were introduced by [Wen06] as Compactly Supported RBF (CSRBF) and satisfy the following condition

$$\varphi(r) = (1-r)_+^q P(r) = \begin{cases} (1-r)^q P(r) & 0 \le r \le 1 \\ 0 & r > 1 \end{cases} \tag{8}$$

where $P(r)$ is a polynomial function and q is a parameter. Typical examples of CSRBF are

$$\begin{aligned}
\varphi_1(r) &= (1-\varepsilon r)_+ \\
\varphi_2(r) &= (1-\varepsilon r)_+^3 (3\varepsilon r+1) \\
\varphi_3(r) &= (1-\varepsilon r)_+^5 (8(\varepsilon r)^2 + 5\varepsilon r + 1) \\
\varphi_4(r) &= (1-\varepsilon r)_+^2 \\
\varphi_5(r) &= (1-\varepsilon r)_+^3 (4\varepsilon r+1) \\
\varphi_6(r) &= (1-\varepsilon r)_+^6 (35(\varepsilon r)^2 + 18\varepsilon r + 3) \\
\varphi_7(r) &= (1-\varepsilon r)_+^8 (32(\varepsilon r)^3 + 25(\varepsilon r)^2 + 8\varepsilon r + 1) \\
\varphi_8(r) &= (1-\varepsilon r)_+^3 \\
\varphi_9(r) &= (1-\varepsilon r)_+^3 (5\varepsilon r+1) \\
\varphi_{10}(r) &= (1-\varepsilon r)_+^7 (16(\varepsilon r)^2 + 7\varepsilon r + 1)
\end{aligned} \tag{9}$$

where ε is the shape parameter of radial basis function, see Figure 2 for visualization of (9).

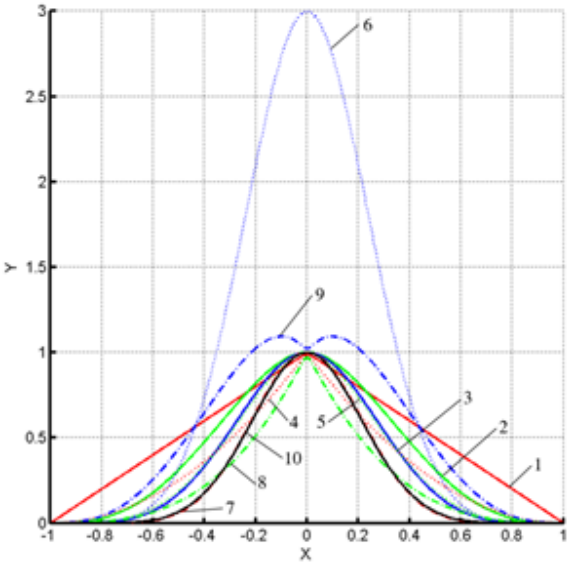


Figure 2: Examples of CSRBF (from [US04])

3.1. Radial Basis Function Interpolation

The RBF interpolation was originally introduced by [Har71] and is based on computing the distance of two points in the k-dimensional space and is defined by a function

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} \lambda_j \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) \tag{10}$$

where λ_j are weights of the RBFs, M is the number of the radial basis functions, i.e. the number of interpolation points, and φ is the radial basis function. For a given dataset of points with associated values, i.e. in the case of scalar values $\{\boldsymbol{x}_i, h_i\}_1^M$, the following linear system of equations is obtained

$$h_i = f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) \quad \text{for } \forall i \in \{1,\dots,M\} \tag{11}$$

where λ_j are weights to be computed, see Figure 3 for visual interpretation of (10) or (11) for a $2\frac{1}{2}D$ function.

Equation (11) can be rewritten in a matrix form as

$$\boldsymbol{A}\boldsymbol{\lambda} = \boldsymbol{h} \tag{12}$$

where matrix $\boldsymbol{A}$ is symmetrical, as $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|$.

The RBF interpolation can be done using "global" or "local" functions. When using "global" radial basis functions the matrix $\boldsymbol{A}$ will be full, but when using "local" radial basis

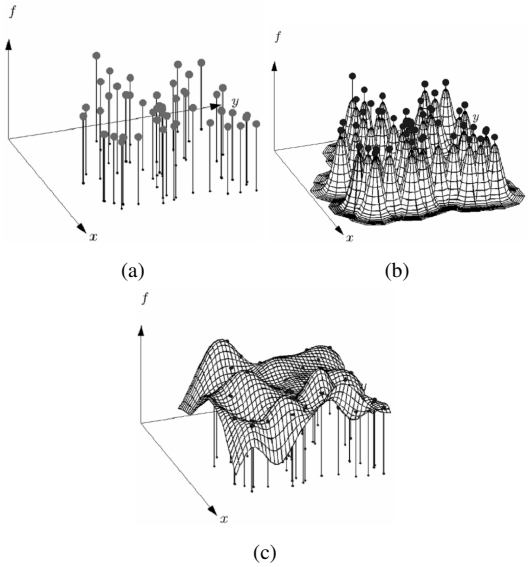


Figure 3: Data values $\{\boldsymbol{x}_i, h_i\}_1^M$ (Figure 3a), the RBF collocation functions (Figure 3b), the resulting interpolant (Figure 3c). (From [FW09]).

functions the matrix $\boldsymbol{A}$ will be sparse, which can be beneficial when solving the system of linear equations $\boldsymbol{A\lambda} = \boldsymbol{h}$.

In the case of the vector data, i.e. $\{\boldsymbol{x}_i, \boldsymbol{h}_i\}_1^M$ values $\boldsymbol{h}_i$ are actually vectors, the RBF is to be performed for each coordinate of $\boldsymbol{h}_i$.

4. Vector Field RBF Approximation

Vector fields are results of numerical simulations or data measuring process. This kind of vector field data has discrete representation, but an analytical formula describing the vector filed is much more useful. We will show how to approximate a vector field using radial basis functions.

A very important future of a vector field are its critical points. The interpolation must preserve positions and types of all critical points. Thus, the RBF interpolation should interpolate the vector field at all positions of critical points to preserve their positions. To preserve their types, we should include few more points in the neighborhood of each critical point to the interpolation. The number of points in the neighborhood was experimentally chosen to be 4, as more points does not improve the interpolation in any significant way. Points in the neighborhood of a critical point $\boldsymbol{x}_0 = [x_0, y_0]^T$ are chosen using the following formula

$$\begin{bmatrix} P_x^{(k)} \\ P_y^{(k)} \end{bmatrix} = \begin{bmatrix} x_0 + r\sin(k\frac{\pi}{2}) \\ y_0 + r\cos(k\frac{\pi}{2}) \end{bmatrix}. \tag{13}$$

where $k \in \{0,1,2,3\}$ and r is a small number depending on the distance of critical points, where the distance to the nearest critical point should be $\gg r$.

This set of critical points together with their neighborhood points can be interpolated using RBF (11), note that each component of vectors $\boldsymbol{v} = [v_x, v_y]^T$ is interpolated separately. This interpolation will preserve the location of critical points together with their types.

To get more accurate interpolation formula of a vector field at points $\boldsymbol{x} \in [x_{min}, x_{max}] \times [y_{min}, y_{max}]$ we can include some more random points from this interval into the interpolation. The improvement of quality depending on the number of additionally included points will be shown in the following chapter.

5. Results

The results will be demonstrated on an analytical vector field, as we can measure the interpolation errors precisely. The analytical vector field, that we choose as an example, is described with the following equation

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} x(\frac{1}{2}x^2 + \frac{1}{2}) + y(-x + (\frac{1}{2}y - 1)y + \frac{1}{2}) \\ \frac{1}{2}x^2 y + x(-\frac{1}{2}y^2 + y - \frac{1}{2}) + \frac{1}{2}y - 1 \end{bmatrix} \tag{14}$$

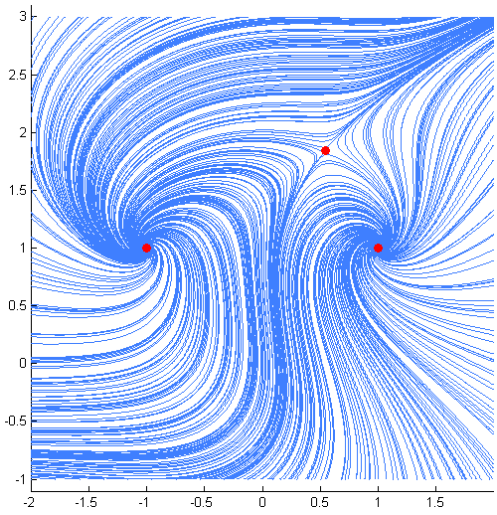
this vector field (14) has three critical points $\boldsymbol{x}_0$

$$\begin{aligned} &\text{source location:} && \boldsymbol{x}_0 = [-1,1]^T \\ &\text{source location:} && \boldsymbol{x}_0 = [1,1]^T \\ &\text{saddle location:} && \boldsymbol{x}_0 = [0.543689, 1.83929]^T. \end{aligned} \tag{15}$$

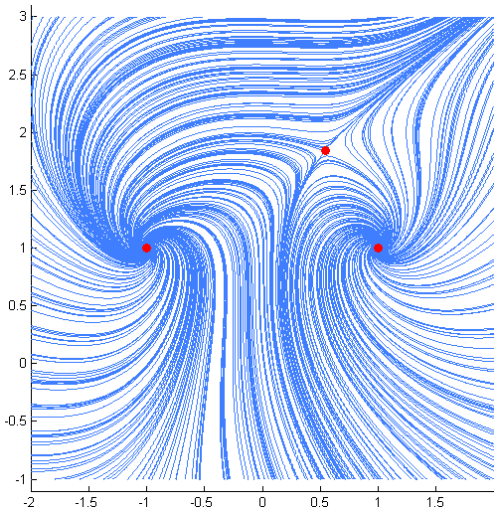
The vector field (14) will be interpolated and tested on interval $[-2,2] \times [-1,3]$, as all important features will be visible. The RBF function used for interpolation is a Gauss radial basis function and the shape parameter ε was experimentally selected as $\varepsilon = 1$.

Vector field (14) can be interpolated using 3 critical point positions and 12 more neighborhood points, i.e. 4 neighborhood points for each critical point. The neighborhood points are computed with (13) and the parameter $r = 0.1$. The v_x component of the vector field is interpolated with one RBF and the v_y component of the vector field is interpolated with one RBF as well. The phase portrait of original analytical vector field (14) is visualized in Figure 4a and the phase portrait of RBF interpolated vector field is visualized in Figure 4b. It can be seen, that both phase portraits look very similar and have the same vector field topology. Moreover, the critical points location is identical, as the average length of displacement error for all critical points is $7.0283 \cdot 10^{-8}$, which is only a numerical error of the critical points location algorithm.

We computed the interpolation error for v_x and v_y and visualized it in Figure 5. It can be seen that the interpolation error is getting higher as the distance from critical

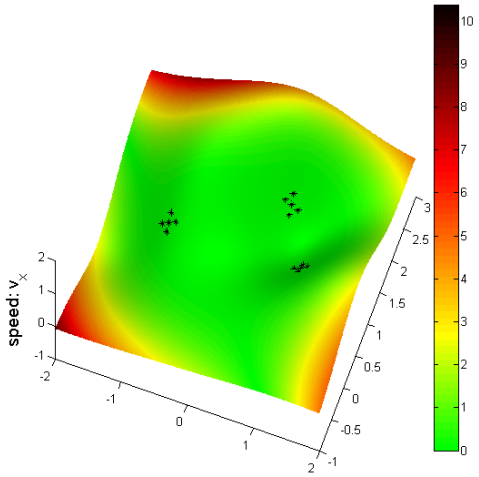


(a)

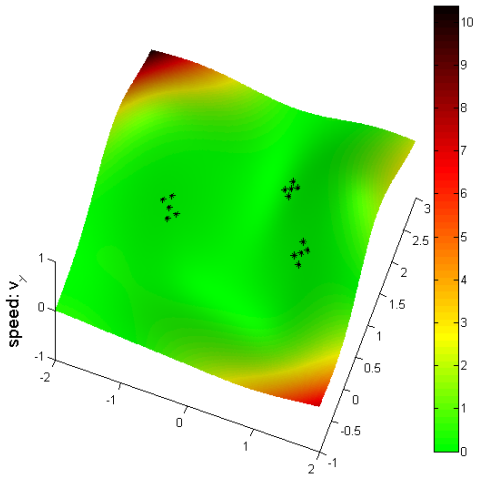


(b)

Figure 4: Phase portrait of the vector field (14) (Figure 4a) and phase portrait of a RBF interpolation using only 15 reference points (3 critical points plus three times 4 neighborhood points) (Figure 4b).



(a)



(b)

Figure 5: Interpolation error of RBF interpolation using only 15 reference points (3 critical points plus three times 4 neighborhood points). Interpolation error of v_x (Figure 5a) and interpolation error of v_y (Figure 5b).

points increases. The average error of vector length at interval $[-2,2] \times [-1,3]$ is 1.7943 (the vector length varies from 0 to 12.6194) and the average error of vector angular displacement is 0.1966 $[rad]$.

The vector field (14) was interpolated using 3 critical points locations plus three times 4 neighborhood points. We can include few more randomly distributed points into the interpolation to reduce the distance error from (14). We choose to generate additional 85 points from interval $[-2,2] \times [-1,3]$, so the interpolation of vector field will contain 10^2 points in total. This interpolation of vector field is visualized in a phase portrait, see Figure 6 and Figure 4a for comparison with original phase portrait.

We computed the interpolation error for v_x and v_y and visualized it in Figure 7. It can be seen that the interpolation error is close to zero except for locations on the border. The average error of vector length at interval $[-2,2] \times [-1,3]$ is 0.0549 (note that the vector length varies from 0 to 12.6194) and the average error of vector angular displacement is 0.0065 $[rad]$.

The average vector length error and the average vector angular displacement error were measured for different number of interpolated points. A number of points k is used as

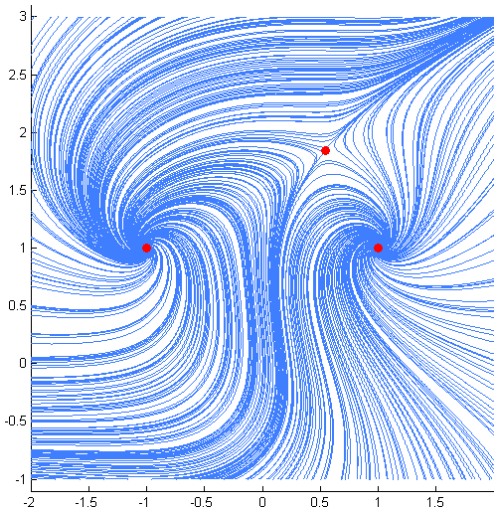


Figure 6: Phase portrait of a vector field RBF interpolation of (14) using 100 reference points (3 critical points plus three times 4 neighborhood points plus 85 randomly distributed points).

added points for the RBF interpolation, thus the RBF interpolation uses $(k+3+3\cdot 4)$ points for interpolation of vector field, i.e. k randomly distributed points from interval $[-2,2]\times[-1,3]$ plus 3 critical points plus three times 4 neighborhood points. Number k was tested from 0 to 400 fifty times for each k with step $\Delta k = 1$ and results are visualized in Figure 8.

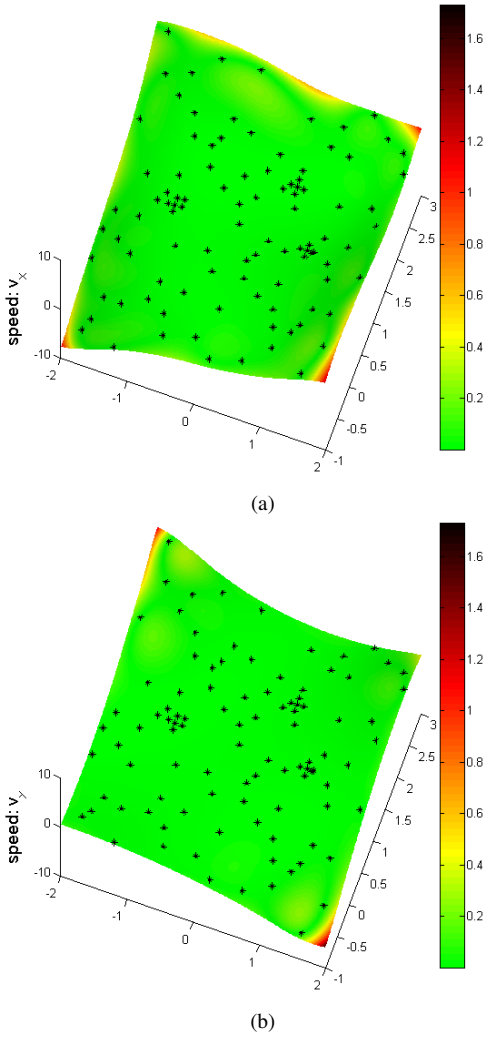
It can be seen that both errors in Figure 8 decrease with increasing number k of added points for the interpolation of vector field. According to the required accuracy of the interpolation, the user can select the minimal necessary number of added points and perform the interpolation according to the algorithm proposed.

6. Conclusions

We presented a new and easy to implement approach for the vector field approximation using radial basis functions. In general, it can be used in any $d-$dimensional space, although the results were presented only for $2D$ vector field. The proposed RBF interpolation proved the ability to approximate a vector field when preserving the location of critical points and the vector field topology as well.

The proposed approach offers not only analytical description of the discrete data of vector field, but also a significant data compression. This might be a significant feature for "progressive vector field visualization" approach.

In future, the proposed approach will be deeply explored for $t-$varying data sets together with other aspects for very large vector field data set interpolation. The more sophisticated placement of interpolation points around critical points will be deeply explored as well.



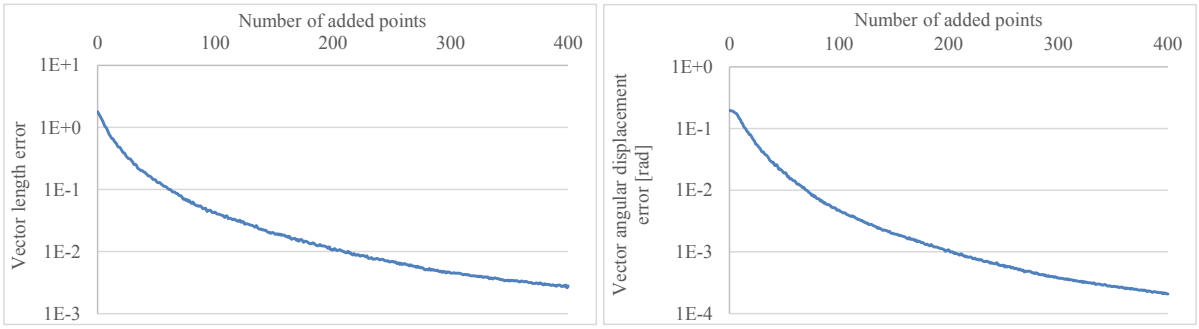
(a)

(b)

Figure 7: Interpolation error of RBF interpolation using 100 reference points (3 critical points plus three times 4 neighborhood points plus 85 randomly distributed points). Interpolation error of v_x (Figure 7a) and interpolation error of v_y (Figure 7b).

Acknowledgement

The authors would like to thank their colleagues at the University of West Bohemia, Plzen, for their discussions and suggestions, and anonymous reviewers for their valuable comments and hints provided. The research was supported by MSMT CR project LH12181 and SGS 2016-013.



(a) (b)

Figure 8: Average errors of the RBF interpolation of vector field (14) using k added reference points, i.e. 3 critical points plus three times 4 neighborhood points plus k randomly distributed points, where $k \in \{0,\dots,400\}$. The vector field length error, note that the vector length varies from 0 to 12.6194 (Figure 8a) and the vector field angular displacement error (Figure 8b).

References

[EJF09] EDWARD J. FUSELIER G. B. W.: Stability and error estimates for vector field interpolation and decomposition on the sphere with rbfs. *SIAM Journal on Numerical Analysis 47*, 5 (2009), 3213–3239. 1

[Fas07] FASSHAUER G. E.: *Meshfree approximation methods with MATLAB*, vol. 6. World Scientific, 2007. 2

[FP08] FORNBERG B., PIRET C.: On choosing a radial basis function and a shape parameter when solving a convective PDE on a sphere. *J. Comput. Physics 227*, 5 (2008), 2758–2780. 3

[FW09] FLYER N., WRIGHT G. B.: A radial basis function method for the shallow water equations on a sphere. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* (2009), The Royal Society, pp. rspa–2009. 4

[Har71] HARDY R. L.: Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research 76*, 8 (1971), 1905–1915. 3

[HH89] HELMAN J., HESSELINK L.: Representation and display of vector field topology in fluid flow data sets. *IEEE Computer 22*, 8 (1989), 27–36. 2

[MZT*14] MA J., ZHAO J., TIAN J., YUILLE A. L., TU Z.: Robust point matching via vector field consensus. *IEEE Transactions on Image Processing 23*, 4 (2014), 1706–1721. 1

[PS97] PHILIPPOU P. A., STRICKLAND R. N.: Vector field analysis and synthesis using three-dimensional phase portraits. *CVGIP: Graphical Model and Image Processing 59*, 6 (1997), 446–462. 2

[PS11] PAN R., SKALA V.: A two-level approach to implicit surface modeling with compactly supported radial basis functions. *Engineering with Computers 27*, 3 (2011), 299–307. 2

[Sch79] SCHAGEN I.: Interpolation in two dimensionsâĂŤa new technique. *IMA Journal of Applied Mathematics 23*, 1 (1979), 53–59. 3

[Ska15] SKALA V.: Meshless interpolations for computer graphics, visualization and games. In *Eurographics 2015 - Tutorials, Zurich, Switzerland, May 4-8, 2015* (2015). 2

[US04] UHLIR K., SKALA V.: Radial basis function use for the restoration of damaged images. In *International Conference on Computer Vision and Graphics, ICCVG 2004, Warsaw, Poland, September 2004, Proceedings* (2004), pp. 839–844. 3

[Wen06] WENDLAND H.: Computational aspects of radial basis function approximation. *Studies in Computational Mathematics 12* (2006), 231–256. 3

[WTS*05] WEINKAUF T., THEISEL H., SHI K., HEGE H., SEIDEL H.: Extracting higher order critical points and topological simplification of 3d vector fields. In *16th IEEE Visualization Conference (VIS 2005), 23-28 October 2005, Minneapolis, MN, USA* (2005), p. 71. 2

[ZMT06] ZHANG E., MISCHAIKOW K., TURK G.: Vector field design on surfaces. *ACM Trans. Graph. 25*, 4 (2006), 1294–1326. 1

SIGRAD 2016
M. Hayashi (Editors)

Output Sensitive Collision Detection for Unisize Boxes

Gabriele Capannini and Thomas Larsson

Mälardalen University, Sweden

Abstract

We show how a recent collision detection method, which is based on the familiar sweep and prune concept, can gain improved performance for the special class of simulations that only involves axis-aligned bounding boxes of the same size. The proposed modifications lead to a worst-case optimal output-sensitive algorithm in 2D. Furthermore, the experimental result shows that our method gives generous speedups in practice and that dynamic scenes with one million objects can be processed at interactive rates even on a laptop.

Categories and Subject Descriptors (according to ACM CCS): F.2.2 [Analysis Of Algorithms And Problem Complexity]: Nonnumerical Algorithms and Problems—Geometrical problems and computations; I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures;

1. Introduction

Fast collision detection is a required operation in many types of physical simulation, video games, and computer graphics applications. It is also an important operation in virtual reality systems and robotics. Clearly, to determine all contacts in a virtual environment with n moving bodies or geometrical objects at interactive rates is a challenging computational problem.

In 1992, Baraff introduced a well-known collision detection algorithm which is now known as the Sweep and Prune (SaP) method [Bar92]. The original algorithm has later been improved to deal with larger inputs and more challenging scenarios, for instance, by using extended data structures and parallelization [TBW09, LHLK10]. A recent variant gains efficiency through a dual-axis sweeping approach to defeat bottlenecks arising from large chunks of overlapping intervals along the projection axes [CL16].

In what follows, we propose a new variant of this dual-axis approach which uses an improved data structure to speed-up the SaP computation when all the simulated objects can be enclosed in equally sized axis-aligned bounding boxes (AABBs). Thus, we study the problem: Given n dynamic axis-aligned boxes of the same size, report all pairwise intersections. In particular, we are able to give an algorithm that runs in $O(n \cdot \log n + |C|)$ time, where $|C|$ is the size of the output, given simulation scenarios that essentially are two-dimensional. Furthermore, it is straightforward to apply our method also in fully three-dimensional simulations with an expected high performance in practice, but without the stated theoretical guarantee.

2. Background

Collision detection is a well-studied topic in the computer graphics community. What algorithm that is preferable depends on several factors such as the geometric representation, nature of body motions, type of query, required accuracy, and overall scene complexity. Application-specific knowledge can often be exploited to accelerate the process. Therefore, numerous algorithms and data structures have been presented for varying circumstances, contexts, and applications.

Strategies based on sorting or bucketing are often used, for example SaP, uniform grids, and hashing techniques. Many other approaches involve the use of hierarchical data structures, such as bounding volume hierarchies, k-d-trees, quadtrees, and octrees [Sam05]. However, it is beyond the scope of this study to discuss and evaluate such techniques. For a broader view of the collision detection problem, interested readers may turn to existing surveys (see e.g. [Eri04, TKH*05] and chapter 2 in [Wel13]).

Our efforts have been focused on finding efficient variants of SaP. Besides the initial research efforts [Bar92, CLMP95], several attempts have been made to improve the efficiency of SaP. Simulations of large datasets require a mechanism to handle the complexity arising from the growth of the number of interval overlaps along the projection axes. Therefore,

several recent methods use a combination of SaP and spatial subdivision [TBW09, LHLK10]. Alternatively, the SaP method can also be enhanced by realizing efficient range queries during the sweeping of a secondary axis [CL16].

Clearly, variants of SaP have been successfully applied to different kinds of multi-body simulations with varying scene complexity, object representation, and type of motion [CS06, TBW09, CL16]. Thus, the SaP family of methods appears to be quite general and powerful. In the next section, we show how SaP can be further optimized in cases where the sizes of the simulated objects are bounded in such a way that they can be approximated by unisize boxes.

3. Our Algorithm

This section introduces the proposed approach to SaP for 2D scenarios and equally sized AABBs. In this type of computation, boundaries of each box project on each coordinate axis an interval $a = [a^+, a^-]$, such that $a^+ < a^-$ and $|a| = a^- - a^+$ denotes the length of a. At each frame of the simulation, objects move so that their AABB projections can overlap and, when the projections of two boxes overlap on all axes, the AABBs collide. Clearly, two intervals a and b overlap iff:

$$a^+ \leq b^- \ \wedge \ b^+ \leq a^- \tag{1}$$

As Figure 1 shows, any two overlapping intervals a and b satisfy Equation 1.

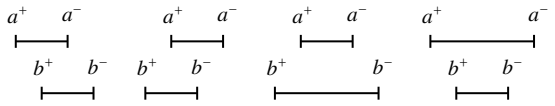


Figure 1: *There are four cases for which a and b overlap.*

The original SaP algorithm consists in sorting the box boundaries (a.k.a. *endpoints*) related to a coordinate axis, then sweeping the result to find out the colliding box pairs [Bar92]. An object a is added to a list (a.k.a. *activelist*) when its *low-endpoint* a^+ is picked and it is successively removed once its *high-endpoint* a^- is encountered. Furthermore, when the object is removed from the activelist, a set of full box-box tests is performed to discover possibly collisions between the removed object and the ones remaining in the activelist. During this phase, many false positives are encountered, namely the objects which do not intersect the one removed on the remaining axis. In a previous paper, we showed how our dual-axis approach can reduce the number of false positives substantially in 3D [CL16]. We propose, now, an approach which completely avoid false positives.

Claim 1 *By means of a stable sorting algorithm and under the unisize assumption, the endpoints projected on a coordinate axis can be ordered in such a way that if two intervals a and b overlap then:* $a^+ \prec b^+ \prec a^- \prec b^-$ *or* $b^+ \prec a^+ \prec b^- \prec a^-$.

In other words, Claim 1 states that the AABB intervals can be represented so that any of them does not fully include another one in the list of sorted endpoints. As a consequence, given an interval a, the corresponding set of overlapping intervals on the first axis is identified by the objects of which one endpoint is between a^+ and a^-. To ensure Claim 1, we arrange the endpoints in memory in a particular way: the $2n$ endpoints related to each axis are stored in separate arrays and, for each interval a of which object id is $i \in [0, n-1]$, a^+ and a^- are placed at position i and $i+n$ of each array, respectively (as shown in Figure 2).

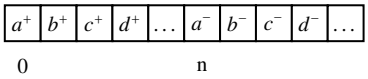


Figure 2: *Array representation of the intervals with low endpoints preceding the high endpoints.*

Moreover, the sorting algorithm used here is a variant of the stable Least Significant Digit Radix Sort (LSDRS) algorithm [Knu98], which returns the indexes of the sorted items instead of permuting the input. So, for each pair of intervals a and b where the endpoints differ, Claim 1 is straightforwardly proved by Equation 1 and the fact that all intervals have the same length, i.e., $|a| = |b|$. Indeed, under the unisize assumption, all the overlapping intervals are included in the first two cases shown in Figure 1. Otherwise, if some endpoints coincide for any couple of intervals, the stable sorting guarantees the order stated in Claim 1, i.e., the low-endpoints precede the high ones while endpoints of the same type are ordered according to their object id.

Algorithm 1 Bi-dimensional unisize SaP in 2D.

Input: $\Omega = \{0, \ldots, n-1\}$ ▷ object *id*s
Input: P_x ▷ first axis endpoints
Input: P_y ▷ second axis endpoints
Output: C ▷ colliding object pairs (init $\varnothing$)

1: $I_x \leftarrow \text{Sort}(P_x)$ ▷ first axis sorting
2: **for** $i \leftarrow 0$ **to** $2n-1$ **do**
3: $\quad R[I_x[i]] \leftarrow i$
4: $I_y \leftarrow \text{Sort}(P_y)$ ▷ second axis sorting
5: **for** $i \leftarrow 0$ **to** $2n-1$ **do** ▷ second axis sweeping
6: $\quad p \leftarrow I_y[i]$
7: $\quad$ **if** $p < n$ **then**
8: $\quad\quad S \leftarrow S \cup \{R[p], R[p+n]\}$
9: $\quad\quad$ **for each** $q \in S : R[p] < q < R[p+n]$ **do**
10: $\quad\quad\quad C \leftarrow C \cup (p, q)$
11: $\quad$ **else**
12: $\quad\quad S \leftarrow S \setminus \{R[p-n], R[p]\}$
13: **return** C

In Algorithm 1, which describes the entire approach, we exploit Claim 1 by sorting one of the two axes and populating the array $R[]$ with the positions of the ordered endpoints

(loop on Line 2). Hence, for any interval a, the values $R[a^+]$ and $R[a^-]$ are the boundaries of a list of $R[]$ values (a.k.a. *ranks*) of which intervals overlap a. The endpoints on the second axis are sorted (Line 4) and then swept in the loop starting on Line 5. As in the original SaP, the active objects are stored in the set S, but here, they are represented by means of their rank. In particular, the endpoint ranks $R[p]$ and $R[p+n]$ related to the fist axis interval of an object p are added to S when the low-endpoint of p is picked from the second sorted axis, see Line 8. These values, corresponding to $R[p-n]$ and $R[p]$ on Line 12, are successively removed when the high-endpoint of p is picked. For any object a, the corresponding colliding objects are the active ones of which rank q is $R[a^+] < q < R[a^-]$ when a^+ is picked. Such a filtering operation is made by means of a *range query* performed on S using $R[a^+]$ and $R[a^-]$ as search boundaries (Line 9). As a consequence of Claim 1 and given that objects in S overlap the current object a on the second axis, it follows that the set $S \cap R[a^+..a^-]$, returned by the range query, immediately represents the objects colliding with a.

4. Complexity Analysis

In what follows, the time complexity is analyzed in the RAM model with word size k [CR72]. It means that primitive operations on k-bit operands are performed in constant time.

The complexity of Algorithm 1 depends on LSDRS, which runs in linear time, and the two loops starting on Lines 2 and 5, respectively. The first one costs $O(n)$ as it consists of $2n$ assignments. The second loop costs $O(n)$ multiplied by the cost of the operations performed on S: *ins*, *del* and *range query*. The implementation of S, consists in a perfect k-ary tree with n leaves, see [CL16]. This tree (a.k.a. SuccTree) has two main features: each node is represented by only one bit and the available operations (listed above) make use of bit-level parallelism to run in asymptotically optimal time. This yields a complexity of $O(\log_k n)$ for insertion and deletion, while performing a range query costs $O(\ell \cdot \log_k n)$, where ℓ denotes the number of values returned by the query.

Here, we enhance such a tree by linking the currently stored values in an ordered linked list. This does not degrade the complexity of *ins* and *del*, but the required memory grows by a factor k due to the linked list. Thus, we can perform the range query on Line 9 in $O(\ell)$ time by accessing directly the leaf node corresponding to the value $R[a^+]$ and iterating through the list up to $R[a^-]$. Thus, the overall complexity of Algorithm 1 is $O(n \cdot \log_k n + |C|)$ where $|C|$ equals the total number of values returned by the range queries performed, i.e., the number of collisions.

As a final remark, under the assumption of non-penetrating rigid bodies, we have that the number of box pairs in contact $|C| = O(n)$, which means that the complexity of Algorithm 1 becomes $O(n \cdot \log_k n)$.

5. Experimental Evaluation

We implemented our solution in C/C++ using the gcc 5.3.0 compiler, and the runs were executed single-threaded on a 2.80 GHz Intel i7-4810MQ CPU with 16 GB RAM running Ubuntu 14.04. In all runs, it was confirmed that the output array with overlapping box-pairs was correct by comparing the results of the used algorithms.

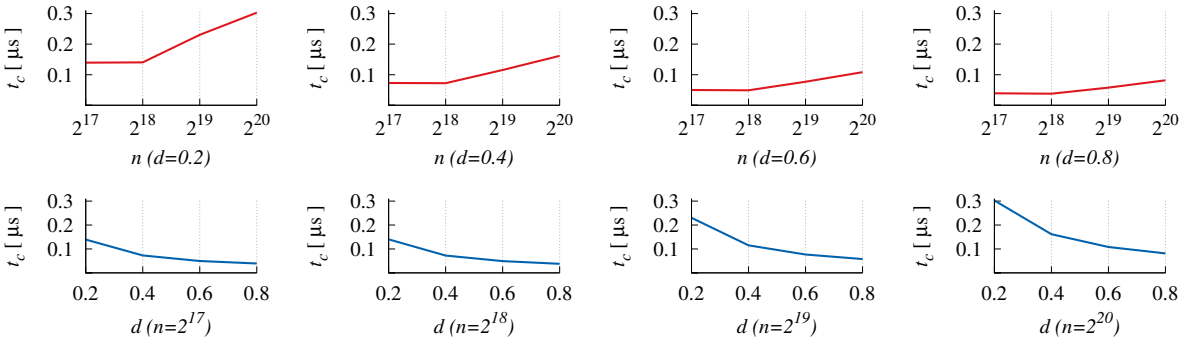
d	n	[Bar92]	[KMZ13]	Our
0.2	2^{17}	0.093	0.090	0.017
0.2	2^{18}	0.236	0.199	0.035
0.2	2^{19}	0.664	0.438	0.090
0.2	2^{20}	1.883	0.955	0.220
0.4	2^{17}	0.129	0.098	0.017
0.4	2^{18}	0.332	0.215	0.036
0.4	2^{19}	0.936	0.470	0.091
0.4	2^{20}	2.653	1.021	0.216
0.6	2^{17}	0.157	0.103	0.017
0.6	2^{18}	0.405	0.225	0.036
0.6	2^{19}	1.143	0.489	0.087
0.6	2^{20}	3.239	1.063	0.219
0.8	2^{17}	0.181	0.106	0.018
0.8	2^{18}	0.502	0.232	0.037
0.8	2^{19}	1.318	0.504	0.087
0.8	2^{20}	3.747	1.093	0.215

Table 1: *Elapsed collision detection time (in seconds).*

Our first experiment consisted of n equally-sized axis-aligned squares that moved randomly in a planar environment. The simulation was repeated for different spatial densities d, where d was the sum of the space occupied by the squares divided by the space of the environment. Initially, the squares were randomly placed in the environment under a uniform distribution. The squares bounced on the environment borders, but since no collision response was used, the squares were able to pass through each other. In general, this increases the number of overlaps that has to be reported in each frame of the simulation.

The average runtimes per frame are reported in Table 1. We compared our algorithm to the original SaP and to the algorithm for finding all intersecting pairs of iso-oriented boxes available in CGal [KMZ13]. As can be seen, our method was significantly faster in all cases. It outperformed the original SaP by more than an order of magnitude (for large datasets). Compared to the algorithm provided in the CGal library, we observed speedups around $5\times$. Furthermore, in contrast to the other used algorithms, our solution maintained almost the same performance regardless of the density.

To examine the relation between the empirical performance of our solution and its theoretical complexity, we now



d	0.2	0.2	0.2	0.2	0.4	0.4	0.4	0.4	0.6	0.6	0.6	0.6	0.8	0.8	0.8	0.8
n	2^{17}	2^{18}	2^{19}	2^{20}	2^{17}	2^{18}	2^{19}	2^{20}	2^{17}	2^{18}	2^{19}	2^{20}	2^{17}	2^{18}	2^{19}	2^{20}
$\|C\|$	53	105	210	420	105	210	420	840	158	315	631	1260	211	421	841	1681
t_p	7	14	48	127	7	15	48	135	7	15	48	136	8	15	48	137

Figure 3: *Time-per-collision t_c calculated in µs as the pairing phase runtime t_p over the number of collisions $|C|$ by varying the number of objects n and their density d. In the table at the bottom, t_p and $|C|$ are shown in ms and thousands, respectively.*

consider the behaviour of our algorithm in more detail. Section 4 shows that the loop starting on Line 5 of Algorithm 1 (referred as *pairing phase* in the rest of the section) dominates the complexity of the algorithm. Hence, in what follows, we focus our attention only on the performance of that phase. Since $|C| = O(n^2)$ and given that the complexity is $O(n \cdot \log_k n + |C|)$, the number of collisions $|C|$ dominates the entire formula in high-density simulations. In simpler scenarios, however, the object density is low and the number of collisions decreases and the time spent for updating S becomes more relevant than the time spent for enumerating the colliding pairs.

Graphs depicted in Figure 3 show the throughput of the pairing phase computed as the runtime over the number of collisions detected by varying d and n. Given a fixed density d, as n increases, the time-per-collision grows as well, because the number of *ins* and *del* operations performed on S increases. Clearly also $|C|$ increases, but, as shown in the table in Figure 3, it grows proportionally to n (due to the fact that objects can pass through each other and they are uniformly distributed in the world space) while the time spent for updating S grows as $O(n \cdot \log_k n)$. Given a fixed input size n, as d grows, the time-per-collision is expected to be almost constant as the number of insertion and deletion performed on S is the same for all densities. Instead, as the results show, it decreases due to the implementation of S. In fact, as the object density augments, the values stored in the SuccTree get more dense, which increases the efficiency of the *ins* and *del* operations as shown in [CL16]. Furthermore, the average time spent for adding and removing the objects to the SuccTree is better amortized due to the increased number of collision detected.

Finally, comparing the time-per-collision in all graphs, we observe that the worst time-per-collision results are related to large input sizes with low densities. This is due to the new implementation of S which, by means of the linked list, is able to drastically reduce the complexity of the range query (see Section 4), but, on the other hand, its implementation requires an array of $O(n)$ entries. When a range query is performed, such an array is accessed in an ordered manner by "jumping" to the next item from the lower boundary until the upper one is reached. Especially when the object density is low, two consecutive items get farther away from each other so that jumping to the next item implies a higher number of cache misses, which degrades the time-per-collision. To get a better analysis of this behavior, a more accurate complexity model is probably required. As a consequence, further investigations could be done by means of the cache-oblivious model [FLPR99].

To further demonstrate the performance of our algorithm, we ran one more experiment. The simulation lasted for 500 frames and 100 000 moving cubes were included. Since the motions of the objects were restricted to a plane, the simulation space was essentially two-dimensional. At the beginning, the objects were laid out uniformly in a square with velocity vectors directed towards its center. In this way the simulation gave rise to an intense clustering with lots of collisions before the objects began to spread out. Furthermore, no collision response was used; we simply counted the number of detected collisions in each discrete time step. Note that

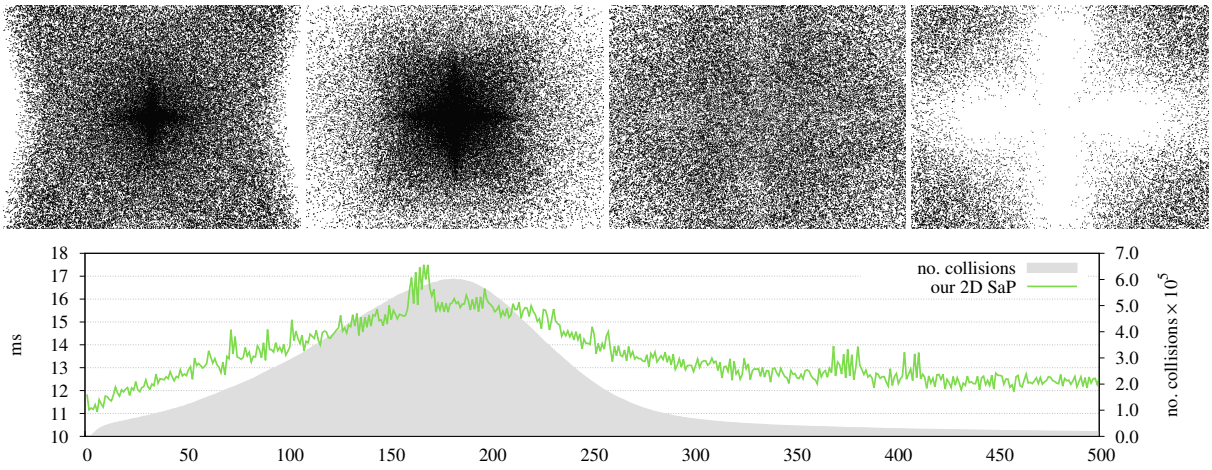


Figure 4: *Simulation of a scenario with clustering ($n = 100\,000$). The simulated objects are unisize 3D boxes, where motion is restricted to a two-dimensional surface, which means we can apply our 2D SaP method. The frames visualized are:* 100, 200, 300, *and* 400. *The plot shows collision detection times per frame (left y-axis) and the number of collisions (right y-axis).*

this choice is likely to force the algorithm to work harder, since it leads to a higher number of collisions than would be the case if the objects bounced off of each other. Four captured images of this scenario are given together with a plot of the results in Figure 4. Changes in the runtime are plotted in green, and the corresponding changes in the detected number of collisions are shown in grey. Clearly, the observed runtimes indicated real-time collision detection performance throughout the entire scenario. The worst, average, and median frame times were 17.5, 13.5, and 13.2 ms, respectively.

6. Possible Applications

We believe that there exist challenging scenarios satisfying the assumption of unisize boxes which are important in certain kinds of interactive computer graphics applications. For animated objects of roughly the same size, the size of the AABB could be set to enclose all possible poses and orientations of the objects. The size could also be extended further to enable detection of nearest neighbours, which might be useful for collision avoidance and path planning.

For instance, in large dense simulations of human crowds or animal groups, collision avoidance might become a major computational bottleneck [LM13]. As an illustration of a possible realistic scenario, consider the photograph of herding wildebeests in Figure 5. Although the movement of the herd seems quite cautious, there can be a lot of action in, e.g., wildebeest stampedes.

Besides cases that are essentially two-dimensional, there are fully three-dimensional simulations that could benefit from our unisize box assumption as well. For example, consider simulation of kinematic chains, representing objects such as ropes, cables, or protein structures, where each segment has a similar size as the others [AGN*04]. Moreover, in certain types of particle simulations, it could also be favourable to consider using a fixed box size.

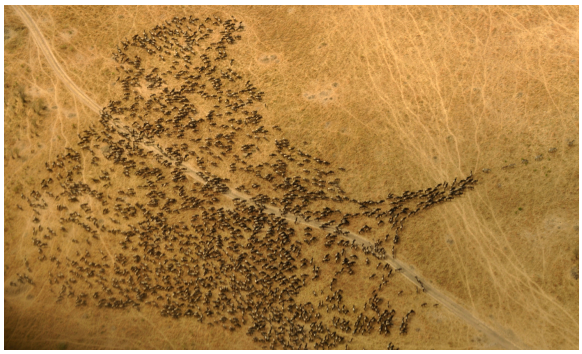


Figure 5: *Photograph of a wildebeest migration [Ram10].*

7. Conclusions

The proposed modification of our more general SaP algorithm leads to an output sensitive collision detection method that is able to handle large datasets in dense environments efficiently. Straightforwardly, by adding an additional interval overlap test for box pairs overlapping in the first two dimensions, the algorithm can provide a healthy speedup also in 3D simulations of unisize boxes. The complexity analysis, however, is not transferable.

To further improve the performance of our approach, we

will turn to parallelization. Both CPU and GPU architectures offer interesting parallel features that we would like to exploit to realize a scalable parallel solution. In particular, the two sets of endpoints in Algorithm 1 can be sorted concurrently so as to reach a scalability of at most two, while the loop on Line 2 can be fairly divided among all the available CPU cores since it consists of a set of independent assignments. When we implemented such an initial parallelization, we obtained an average overall speedup of only 1.25×. Consequently, our future work will focus on finding a more complete data-parallel and scalable approach that addresses both the sorting and the pairing phase.

Another opportunity would be to evaluate the algorithm in more realistic applications, e.g., real-time simulation of fast-moving massive crowds. Moreover, it would also be interesting to further analyse the consequences of the assumption of unisize objects in the three-dimensional case as well as considering how to properly generalize our algorithm to handle continuous motion of the objects.

Acknowledgements

The authors are supported by the SSF grant no. IIS11-0060.

References

[AGN*04] AGARWAL P., GUIBAS L., NGUYEN A., RUSSEL D., ZHANG L.: Collision detection for deforming necklaces. *Computational Geometry: Theory and Applications 28*, 2-3 (2004), 137–163. 5

[Bar92] BARAFF D.: *Dynamic Simulation of Non-Penetrating Rigid Bodies*. PhD thesis, Cornell University, 1992. 1, 2, 3

[CL16] CAPANNINI G., LARSSON T.: Efficient collision culling by a succinct bi-dimensional sweep and prune algorithm. In *Proceedings of the 32nd Spring Conference on Computer Graphics (SCCG)* (2016). 1, 2, 3, 4

[CLMP95] COHEN J. D., LIN M. C., MANOCHA D., PONAMGI M.: I-Collide: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics* (1995), pp. 189–196. 1

[CR72] COOK S. A., RECKHOW R. A.: Time-bounded random access machines. In *Proceedings of the fourth annual ACM symposium on Theory of computing* (1972), pp. 73–80. 3

[CS06] COMING D. S., STAADT O. G.: Kinetic sweep and prune for multi-body continuous motion. *Computers & Graphics 30*, 3 (2006), 439–449. 2

[Eri04] ERICSON C.: *Real-Time Collision Detection*. Morgan Kaufmann, 2004. 1

[FLPR99] FRIGO M., LEISERSON C. E., PROKOP H., RAMACHANDRAN S.: Cache-oblivious algorithms. In *40th Annual Symposium on Foundations of Computer Science* (1999), pp. 285–297. 4

[KMZ13] KETTNER L., MEYER A., ZOMORODIAN A.: *Intersecting Sequences of dD Iso-oriented Boxes*, 4.2 ed. CGal Editorial Board, 2013. 3

[Knu98] KNUTH D. E.: *The art of computer programming, volume 3: sorting and searching (2nd Edition)*. Addison-Wesley, 1998. 2

[LHLK10] LIU F., HARADA T., LEE Y., KIM Y. J.: Real-time collision culling of a million bodies on graphics processing units. *ACM Transactions on Graphics 29*, 6 (2010), 154:1–154:8. 1, 2

[LM13] LI B., MUKUNDAN R.: A comparative analysis of spatial partitioning methods for large-scale, real-time crowd simulation. In *21st International Conference on Computer Graphics, Visualization and Computer Vision* (2013), pp. 104–111. 5

[Ram10] RAMAN T. R. S.: Photograph of wildebeest herding and following a few leading zebra in the Masai Mara Kenya, 2010. CC BY 3.0 (http://creativecommons.org/licenses/by/3.0), via Wikimedia Commons. 5

[Sam05] SAMET H.: *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2005. 1

[TBW09] TRACY D. J., BUSS S. R., WOODS B. M.: Efficient large-scale sweep and prune methods with AABB insertion and removal. In *Proceedings of the IEEE Virtual Reality Conference* (2009), pp. 191–198. 1, 2

[TKH*05] TESCHNER M., KIMMERLE S., HEIDELBERGER B., ZACHMANN G., RAGHUPATHI L., FUHRMANN A., CANI M.-P., FAURE F., MAGNENAT-THALMANN N., STRASSER W., VOLINO P.: Collision detection for deformable objects. *Computer Graphics Forum 24*, 1 (2005), 61–81. 1

[Wel13] WELLER R.: *New Geometric Data Structures for Collision Detection and Haptics*. Springer, 2013. 1

SIGRAD 2016
M. Hayashi (Editors)

Analysis of Camera Work in Horror Movies

Liselotte Heimdahl[1] Yoshihisa Kanematsu[2] Naoya Tsuruta[1] Ryuta Motegi[2] Koji Mikami[1] and Kunio Kondo[1]

[1]Tokyo University of Technology, Japan
[2]Tokyo Metropolitan University, Japan

Abstract

Camera work parameters such as shot size, camera movement, and camera angle are crucial parts for creating digital contents such as movies and games. After creating a story that is to be conveyed to the audience, it is important to be able to present the intended feeling in the contents. By using the same content but by modifying the parameters, such as from a long distance shot to a close up, we can get a great difference in the impact of the viewer's feelings. The purpose of this study is to analyze camera work parameters in horror movies and find features that can be used as reference in future research for automatically reproducing suitable camera work in digital contents.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Camera Work—Animation

1. Introduction

In this research, we focus on camera work and shot design within horror movies. Horror movies are interesting in the way that they evolve as the movie proceed. Usually, a horror movie consists of a number of scenes and cuts. These scenes and cuts are in turn used to lead the audience through a path of emotion. To get the most out of this experience, each scene and cut has to be planed well and produced not only to convey the right emotion, but also to convey the emotion at the right time and with the right force. If a horror movie only consisted of horror scenes then each scene would not have the same affect on the audience as if there were only a couple of horror scenes placed at the right moment. One example is a scene where a character is going to jump over a hole in the ground. If the character conveys the same feeling before, during, and after the jump, the audience are given very little to work with. Since the audience reacts to change, a variety of character emotions are needed to emphasize the desired emotion to the audience [FJ95].

Another example is that, if the same type of scare would be used throughout a whole movie it would numb the audience. They would simply get used to it so that it would not affect them as much in the end as it did in the beginning. Since you want to build up to the end of the movie, such a method would not work.

Some research has already been performed concerning classification of camera work. The goal of this study is to analyze how camera work is evolving throughout horror movies. A concept of five different stages called "The stages of fear" has already been defined and descriptions of these can be found in section 2 [DWM12]. By using "fear stages", we want to compare the scenes within each and every type of stage to be able to find out if there are any notable differences between them, and also if the results differ depending on where on the timeline the stage was found.

For this study, we focused on horror scenes, and we also collected information from the other types of scenes for future use. The system described in this paper will therefore also be of use when analyzing cuts from other types of scenes.

This research is meant to be a starting point for further research, which is one of the reasons for the fact that only one movie has been analyzed. However, the movie used in this research is a horror movie that have received a high score on the site imdb.com. IMDB is a site with a large amount of users, and it is also possible to see how many votes each movie has received. The amount of votes can then be used to evaluate how reliable the score of the movie is.

In this research we investigate if it is possible to find trends in camera work that later can be used to create a database. The database will store the information of when

which type of camera work would be suitable and then place and move the camera accordingly.

2. Related work

The five stages of fear is defined by using a set of visual and audio cues [DWM12]. By defining what stages are used and to what extent in a movie, they were able to analyze and compare several horror movies. However they did not go in to the usage of camera work within a specific stage more then in some cases, which were very brief.

A short description of the five stages of fear can be found bellow.

(1) Terror - The terror stage is characterized by it's ability to evoke anxiety and anticipation in the viewer. More than simply show the viewer a concrete threat or danger, the fear is directed towards a situation and possible danger further on. The rhythm is slow but tends to rise as the stage proceed.

(2) Horror - The horror stage is characterized by suddenness and fast pace. The fear is directed towards a present danger such as another character or object. It is fast paced and used to startle and/or chock the viewer.

(3) Repulsion - Characterized by evoking the feeling of disgust and detachment in the viewer. The emotion can be directed to either an object or a situation. It could contain moments showing, physical pain, gore, unacceptable social behavior or aversive stressful situations, such as being trapped in the dark.

(4) Recovery - Evokes a feeling of safety after dealing with a threat. It can also be the realization of something that gives hope to solve a problem or that the threat is not a treat anymore.

(5) Background - This stage is associated with the feeling of safety, and there will never be any sense of danger. However, other emotions that are not related to fear, disgust or anxiety can also be found, such as happiness or sadness.

Xu et al. have created a system for analyzing the camera work in Robot anime [XKM*15]. The method used proved to be useful for this research in the sense of what type of information that would be good to analyze. The information could not be used as it was, because of the great difference in genre, but it served good as guidelines. In the same research a scrapbook system can also be found for analyzing various information camera information. Other research using similar scrapbook system has also been done and served as reference for creating a scrapbook in our research as well [KK08] [MKT*14]. The scrapbook system will be further described in section 3.

Burelli et al. conducts research about generating camera work depending on the player type [BY13]. The result were automatically created camera work in games that makes it easier for the player to play and even enhances the performance. However it is not clear that this is what takes to enhance the gaming experience per se. In this research we want to establish how to present the content in a way to better affect the user.

Figure 1: *A detail shot by framing, A tale of two sisters. Janghwa, Hongryeon 2003.*

Figure 2: *A detail shot by focus, A tale of two sisters. Janghwa, Hongryeon 2003.*

3. Investigation of Camera Work in Movies

The chosen movie was divided into scenes by using Movie Maker, a movie editing program. Movie Maker could also be used to see the the length of each clip and it has the functionality to easily create screenshots. By dividing the movie into cuts the movie becomes easier to navigate for collecting the shot and camera data.

A model for collecting and record the data from the chosen scenes had to be established. Two types of data was collected; camera and shot information as can be found in Table 1 and Table 2. The shot information consists of; shot size, camera angle and camera motion. Shot size was defined as in [Mil01]. Reference for shot size can be found in Figure 4. The label "varies" was given to those shots where the main targets shot size was spanning over more than one shot size as a result of either camera movement or character movement. As a supplement to these, the detail shot was also examined. Not everyone defines a detail shot the same. In this research it is defined as; a shot where the camera is not focus-

Table 1: *Collected camera information*

shot size	extreme close-up (ECU), very close-up (VCU), big close up (BCU), close-up (CU), medium close-up (MCU), medium shot (MS), knee shot (Knee), medium long shot (MLS), long shot (LS), extreme long shot (ELS), varies
camera angle	top, high, low, medium, point of view (POV), varies
camera motion	static, handheld, pan, tilt-up, tilt-down, rotation, tracking, push-in, push-out, crane, steadicam (one or multiple)

ing on the main target's face and/or when some other object is framing the picture.

In some cases if a target is not present in the shot it has also been labeled "environment".

The camera movement and camera angles and was defined according to Sijll with addition of the top angle [Sij05]. Reference for the camera angles can be found in Figure 3. If the angle varied in a way that it crossed the border between low and medium or medium and high it was given the label "varies". We did not make any distinction between the camera changing from low to high or high to low etc. since this was not something that was commonly found. By separating it into more labels, the data would have become insufficient to be able to analyze at this moment.

The shot information is stored as a xml file, and it consists out of information that reflects the actual content of the shot. The shot information can be seen in Table 1 and Table 2.

It was also necessary to classify if the scene type was a dialogue scene or some other kind of scene since it was found that the camera work seemed to differ a lot between these two when the data was analyzed. During conversation cuts it a higher tendency to use fixed camera, within the same stage could be noticed and should be further analyzed in the future. This is something that could be explained by a higher tendency for none moving targets. As a consequence of this, only the scene type "other" was analyzed further in this research.

We created a scrapbook type of program to be able to find even more trends and where the remaining information also will come to use. The current state of the scrapbook's ability to process the information is still limited, and it was therefor not used for achieving the current results. It is possible to enter the data found in Table 1 and 2 to get a list of cuts containing the entered input. If you now enter Fear stage as "Terror" in the scrapbook, you will get a list of all cuts with it's respective camera and shot information. But the input information is still limited which is why we analyzed the information directly from the xml file. The only difference is

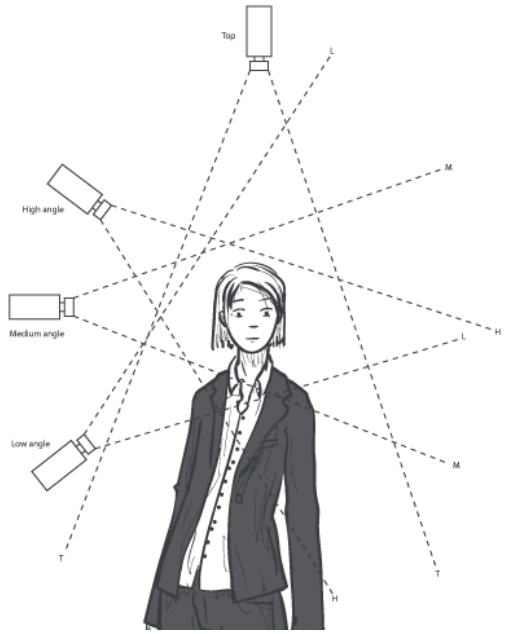


Figure 3: *Reference for camera angle*

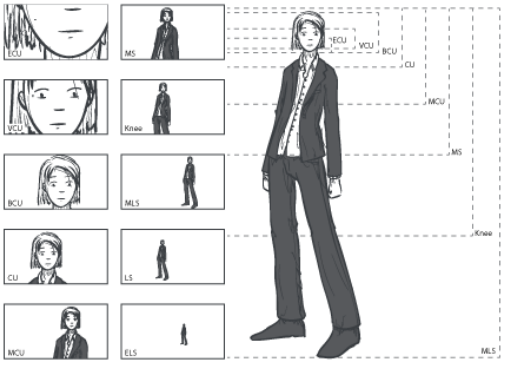


Figure 4: *Reference for shot size*

that it took longer time to find the relevant data. As for now, the shot information used is limited to stage, scene type and detail.

The scrapbook system is also intended to be able to use by a producer as an aid when thinking about how to produce their own camera work.

As for the emotions, we chose anger, disgust, fear, happiness, sadness, and surprise which has been said to be universal facial expressions [EFE13]. Then we also included the emotions interest and neutral for them being well suited for our intended research.

After the camera and shot information had been collected,

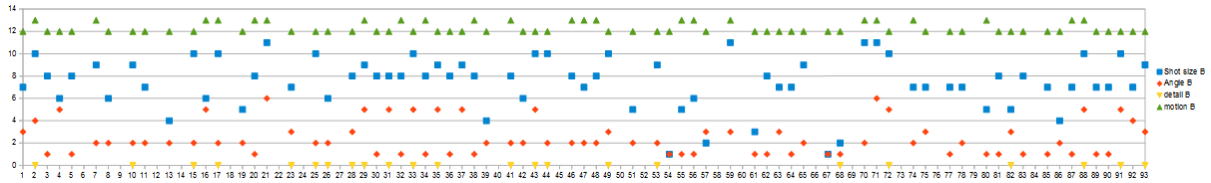


Figure 5: *Background - timeline graph*

Table 2: *Collected shot information*

title	Name of the movie
scene number	1, 2, 3 etc.
scene description	short describing text
stage	Horror, Terror, Repulsion, Recovery or Background
scene type	dialogue, other
number of people in scene	1, 2, 3 etc.
cut number	1, 2, 3 etc.
cut duration	ex. 00:01:00 for 1 second
scene area	indoor, outdoor, changing
main target	protagonist, antagonist, supporting character
sub target	protagonist, antagonist, supporting character
main target emotion	anger, disgust, fear, happiness, sadness, surprise, interest, neutral
sub target emotion	anger, disgust, fear, happiness, sadness, surprise, interest, neutral
main target action	enter stage, leave stage, attack, struggle, defend, run, hide, examine, pick up obj, place obj, communicate, interact, move
sub target action	enter stage, leave stage, attack, struggle, defend, run, hide, examine, pick up obj, place obj, communicate, interact, move
other	detail, over the shoulder, environment

and stored in the form of a xml file, the scenes and cuts where further analyzed and compared within each and every type of stage. The following is a sample of a scene from the xml file.

```
    <scene id ="101" >
      <description>Protagonist and antago-
nist fighting.</description>
      <stage>horror</stage>
      <sceneType>other</sceneType>
      <numberOfPeople>3</numberOfPeople>
      <area>indoor</area>
      <cut id="1">
        <duration>00:00:63</duration>
        <camera>
          <targets>
            <target class = "main">
              <type>protagonist</type>
              <emotion>fear</emotion>
              <action>interact</action>
            </target>
          </targets>
          <shotSize>VCU</shotSize>
          <angle>high</angle>
          <motion>fixed</motion>
          <importantObj>scissor</importantObj>
          <other>detail</other>
        </camera>
      </cut>
    </scene >
```

A total of 556 cuts from the movie “A tale of two sisters” were analyzed. We were focusing on the Terror, Horror and Repulsion stages, but Recovery and Background stages were also analyzed.

Two types of graphs were created. The first type is the timeline graph as can be seen in Figure 5, where it is possible to see how the camera information collected from the shots varies over time and how the angles, camera movements, etc. relates to each other. In the second type of graph the camera work information can be seen in percentage and it was used to compare the usage of camera work between the stages.

The timeline graphs were used to faster find patterns which could be occurring due to some trends in camera work. However, this graph was not intended for analyzing the content on it’s own. The numbers all represents various information as can be seen bellow.

1. Camera movement; 12 - fixed camera, 13 - smooth movement such as tilt and rotation, and 14 - hand camera.
2. Shot size; 1 - ELS, 2 - LS, 3 - MLS, 4 - Knee, 5 - MS, 6 - MCU, 7 - CU, 8 - BCU, 9 - VCU, 10 - ECU, 11 - varies
3. Angle; 1 - Low, 2 - Medium, 3 - High, 4 - Top, 5 - POV, 6 - varies
4. Detail; 1 - detail shot

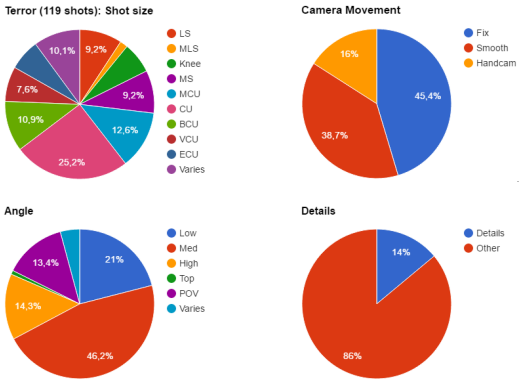


Figure 6: *Terror - shot information*

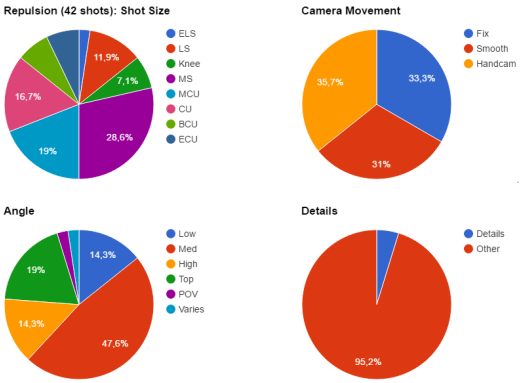


Figure 8: *Repulsion - shot information*

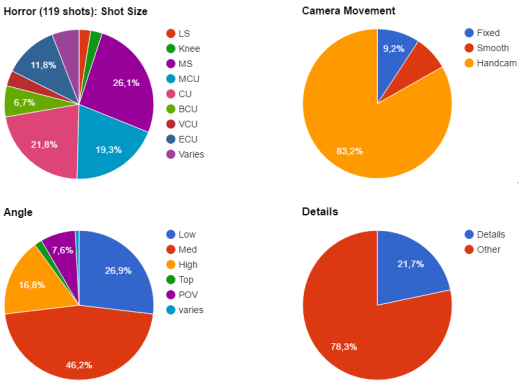


Figure 7: *Horror - shot information*

After finding some possible patterns using the graph it was further investigated by using the xml file where the shot information can be found.

4. Analysis of Camera Work

4.1. Trends for camera work in Horror Movies

To be able to find trends for the camera work, the collected data, as can be seen in Table 1 and 2, was presented in the form of graphs. Four graphs were created for every stage of fear; Shot Size, Angle, Camera Movement and Detail and the data is represented in percentage. The graphs are can be seen in Figure 6-10.

4.2. Analysis of camera work divided by fear stage

4.2.1. Terror - shot analysis

119 cuts from 30 scenes were analyzed for this stage. When it comes to the terror stage, no type of camera work is overused as can be seen in Figure 6. The shot size ranges from LS to ECU with a rather even distribution except for CU which have been found to be used a lot in the other stages of fear, Figure 6-10. The same can also be said for the camera angles and camera movement.

Concerning the camera movement in the terror stage, it greatly differs from the movement found in the horror stage, Figure 7. Whereas the horror stage has a large tendency for hand camera movement, the terror stage uses smooth movement, created by steadicam, rotation and tilt using a stand etc., or fixed camera to a greater extent.

The hand camera movements in the terror stages tends to be used together with POV or views close to POV and their adjacent shots. A common trait during these shots is that the main targets emotion is often fear, which was found using the scrapbook.

For the other camera movements it can be said that they are used to enhance the slow pace of the terror stage accordingly. Therefore the use of hand camera is something that can be seen as strange due to a cue being slow pace. However, another trait of the terror stage is an accelerating rhythm leading up to a horror stage, which can also be found when analyzing these cuts.

4.2.2. Horror - shot analysis

119 shots from 5 scenes were analyzed for this stage. As can be seen in Figure 7, what stands out is the usage of hand camera movement. Dominguez et al. writes that the horror stage is characterized by fast pace and high energy [DWM12]. Something that goes well with the use of hand camera movements. Not all shots however utilize this camera movement. Fixed camera and smooth movement created by steadicam, rotation and tilt using a stand etc. can also be found. When examining the timeline graph for the horror stage the later types of camera movements are usually used together with detail shots. These detail shots in contrast to those, found together with hand camera motion, shows information that are important to the current event of actions.

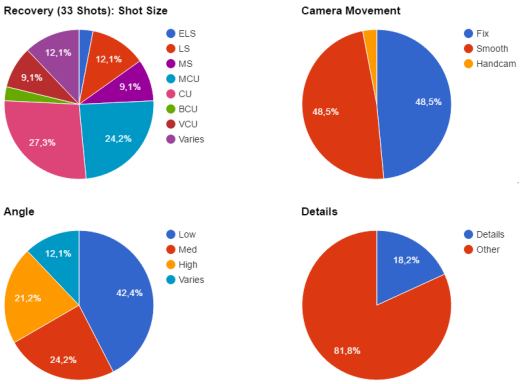


Figure 9: *Recovery - shot information*

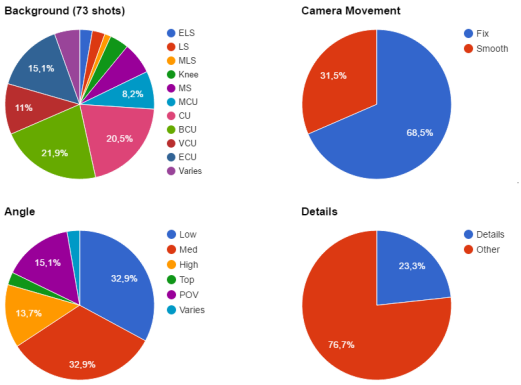


Figure 10: *Background - shot information*

For example, in a fight scene between the protagonist and antagonist, a detail shot of a scissor can be found. This scissor is important and establish information that the two characters takes advantage of in the following cuts to come. Therefore it is important to give the information to the audience in a clear and effective way. If it was presented by using a hand camera it would take longer time to perceive the information, and since it is an action scene it would not be desirable to spend to much time on showing such details. In short, it would slow the pace of the scene even if hand camera motion is usually used with the opposite intention.

4.2.3. Repulsion - shot analysis

The repulsion stage showed to be difficult to analyze due to the lack of relevant cuts. Since the dialogue cuts were not included a total of 42 cuts from 2 scenes were analyzed for this stage. However, some tendencies can still be found when examining Figure 8.

As can be understood by the name repulsion, it is used to make the audience repel from what is seen on the screen. However like concluded by Domenguez et al. the lowest grossing movies among those analyzed were the ones with the largest percentage of repulsion stages [DWM12]. Therefore it should not be good to repel the audience to much which could explain the more common use of ELS and LS when comparing with other stages. While you can perfectly fine apprehend what is going on, it still gives a sense of distance to the actual event. The same can be said about the usage of top view, since it takes away the feeling of being at the scene.

4.2.4. Recovery - shot analysis

33 shots from 9 scenes were analyzed for this stage. This is actually less cuts than analyzed in the repulsion stage section. However a larger number of scenes makes it easier to analyze and find trends.

As can bee seen in Figure 9, this is the stage of fear where the shot size varies the most in one shot. But then it is also the stage where the range of shot sizes is the smallest. There is a tendency for using MCU and CU the most and then LS and VCU for contrast. The sudden realization of safety also goes well with changing shot size such is given by using push in etc.

The camera work tends to smooth movement and fixed camera. The hand camera motion that can be seen in the graph is a reused shot from a different stage.

4.2.5. Background - shot analysis

73 shots from 21 scenes were analyzed for this stage. This is the stage that tends to utilize a fixed camera the most as can be seen when examining Figure 10. Handheld camera tends to not be used at all.

Just like the horror stage, the background stage is characterized by the common usage of detail shots. A large difference however is the length of the cuts. Whereas the detail shots of the horror stage tend not to exceed 1 second, the detail shots found in the background stage can range from around 1 second and up to over 20 seconds. This is not a problem because of the lack of danger and general slow pace that characterizes the background stage.

5. Conclusion

As a conclusion for this research, by analyzing cuts after dividing them by the stages of fear, trends in camera work can be found. The same system could therefore be used to further analyze a greater number of cuts from a larger set of horror movies to be able to find even more trends.

Continued work with the scrapbook system would also help to further analyze the shot and camera information. This would include the feature of being able to see screenshots from the cuts in the scrapbook. By doing this it would not be necessary to go back and forth between Movie Maker and the graphs and it would save a lot of time when analyzing

a greater number of cuts. Implementing the feature of being able to generate graphs directly from the scrapbook would also be helpful for speeding up the process.

We also wanted to find notable differences depending on where the particular stage was found on the movie's timeline. To be able to find these kinds of trends a greater number of movies have to be analyzed.

References

[BY13] BURELLI P., YANNAKAKIS G. N.: Adapting virtual camera behaviour through player modelling. *User Modeling and User-Adapted Interaction 12*, 1 (2013), 1–34. 2

[DWM12] DOMINGUEZ E., WATANABE T., MIKAMI K.: A content-based classification of horror movie scenes using fear stages, 2012. 1, 2, 5, 6

[EFE13] EKMAN P., FRIESEN W. V., ELLSWORTH P.: Emotion in the human face: Guidelines for research and an integration of findings. ISBN 1483147630, 9781483147635. 3

[FJ95] FRANK T., JOHNSTON O.: The illusion of life: Disney animation. ISBN 978-0-7868-6070-8. 1

[KK08] KANEMATSU Y., KANEKO M.: Research on digitizing lighting information from the movies. In *NICOGRAPH International 2008 proceedings(CD-ROM)* (May 2008). The Society for Art and Science. 2

[Mil01] MILLERSON G.: Video production handbook - third edition. ISBN 978-0-7868-6070-8ISBN 0-240-51597-8. 2

[MKT*14] MOTEGI R., KANEMATSU Y., TSUCHIDA T., MIKAMI K., KONDO K.: Color scheme scrapbook using a character color palette template. *International Society on Geometry and Graphics* (2014), 167–174. 2

[Sij05] SIJLL J. V.: Cinematic storytelling: The 100 most powerful film conventions every filmmaker must know (2nd edition edition). ISBN 978-1932907056. 3

[XKM*15] XU H., KANEMATSU Y., MOTEGI R., TSURUTA N., MIKAMI K., KONDO K.: A supporting system for creating camera blocking of the humanoid robot anime's battle scenes (in japanese). In *ADADA Japan* (2015). 2

SIGRAD 2016
M. Hayashi (Editors)

Interactive 4D MRI blood flow exploration and analysis using line predicates

J. Jankowai[†1], R. Englund[1], T. Ropinski[1,2], I. Hotz[1]

[1]Department of Science and Technology, University of Linköping, Sweden
[2]Institute of Media Informatics, Ulm University, Germany

Abstract

We present an interactive exploration tool for 4D PC-MRI blood flow data that incorporates established rendering and filtering methods and combine them into one application. These methods include advanced line illumination, interactively adjustable spatial context visualization and blood flow analysis using line predicates.

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

1. Introduction

The presented system is a tool for interactive exploration and analysis of cardiac blood flow including enhanced rendering and filtering- and grouping-methods. According to the WHO's statistic [Wor15], coronary artery disease, or ischaemic heart disease, was the leading cause of death worldwide with a 13% share in 2012. An early diagnosis can help initiating preventive treatment and prognosis in time.

In a healthy heart, the interplay of hemodynamics and cardiac morphology are very well attuned to one another, resulting in an efficient blood transfer from the heart into the body [KYW*00]. In the case of a cardiovasular disease, one of those two factors may change, leading to characteristic changes in flow patterns in the vessels and heart. In return, these alterations can further change the morphology of the heart which can e.g. lead to heavy deformations of the vessels as it is the case in an aneurism. Such unwanted flow patterns include an increased amount of vortices, increased shear forces on the hearts' vessels or blood remaining in a chamber for more than 1 cycle (increased residence time). Visual data exploration can support the detection and evaluation of such flow patterns. Whether or not a certain flow pattern indicates a potential risk depends on factors such as prevalence, position, severity, and size. In addition, an explorative tool such as ours allows for the examination of data from patients that are known to have a certain cardiovascular disease and discover flow characteristics of that disease.

The data utilized in this work was obtained by flow-sensitive **p**hase-**c**ontrast **m**agnetic **r**esonance **i**maging (4D PC-MRI). In this method, vessel morphology information and 3D time-resolved blood flow velocities are measured in-vivo and simultaneously over a set of full cardiac cycles. This process is repeated for approximately 10-20 minutes and the results of all measured cycles averaged [SAG*14].

4D PC-MRI measurements allow for the creation of cardiac blood flow data giving additional insight into the hemodynamics rather than only morphology. This data can be visualized and analyzed in a 3D context using pathlines. A pathline can be thought of as the path a particle takes in a flow field. Commonly, physicians examine blood flow on 2D cross-sections. These 2D cross-sections are either used as projection or as seeding planes for pathlines by some applications. This only allows for examination in a region of interest and can be very time-consuming and arduous.

In practice, it is very cumbersome and inefficient for a physician to look at a plethora of lines obstructing and cluttering one another since relevant flow structures remain hidden in the excess of information. Pathline predicates [SGSM08, SS06, BPMS12] can be used to filter out the noise mentioned above and also extract and group lines with characteristic features. A pathline predicate is a user-defined property or criterion by which a set of pathlines can be classified and filtered (either the pathline fulfills the predicate or not). Then, each pathline group represents a subflow with respect to a certain behaviour of interest. These include

† Master student at University of Linköping, Sweden

flow paths, velocity, vorticity, or residence time. In the case of blood flow analysis, the physician usually wants to combine some of these properties to a more complex query. In addition to creating these queries, the presented system is extensible so that more predicates can be added easily.

Apart from the challenge of semantic filtering our system deals with some technical and perceptional challenges. A general problem when it comes to rendering pathlines is clutter due to the high number of lines. Without proper lighting, this cluttering results in a loss of spatial perceptibility. Many graphics libraries, such as OpenGL, do not provide automatic access to lighting for polylines. To aid this situation and improve spatial perception, a lighting technique for polylines has been implemented.

The software framework used in this project is called **In**teractive **Vi**sualization **Wo**rkshop (**Inviwo**) and is an extendable open-source framework written in C++ for rapid prototyping of interactive applications [InV15].

In summary, the contributions include:

- Straight-forward and computationally cheap anatomical context visualization
- Enhanced spatial perception and decluttering through line illumination and tube representatives
- Blood flow filtering and analysis based on line predicates

2. Related work

With 4D MRI being a young imaging technique it has not yet found its way into mainstream clinical routines. So far, mostly 4D MRI experts have been concerned with visualizing and analysing the obtained data in the domain of research.

Standard flow visualization techniques like streamline and pathlines [BB99, Buo98] as well as colour-coded 2D planes, vector plots or velocity profiles [MCA*03] have been adopted for 4D MRI blood flow analysis. Even though these methods have been advanced [BBLM10, HSU*10, MKE11] they still require much parameter tuning and in-depth user-knowledge in order to find specific flow patterns in the data.

Over the past few years, 4D MRI blood flow visualization has become an increasingly interesting research area. The main objective is to help the user by increasing usability and comprehensiveness of exisiting methods. To aid this situation, van Pelt et al. introduced a virtual probing approach that allows for flow exploration by interactive seed injection onto the flow field and examination of the flow using illustrative rendering and animation [vPBB*11]. These rendering techniques had been presented in previous work and include arrow-trails to depict time-dependent blood flow dynamics and exploded planar reformats to connect 3D and 2D views of the flow. In addition, they proposed a method to simplify the selection of 2D vessel cross-sections [vPBB*10].

Addressing occlusion and clutter when morphology and blood flow behaviour are visualized in the same context, Gasteiger et al. introduce a ghosted view method displaying the vessel surface whilst revealing the underlying blood flow depending on the orientation between viewer and surface [GNKP10]. Neugebauer et al. present an approach to encode a multitude of parameters on the 2D plane isolating the main vessel from an aneurism. In addition, interaction widgets customized to the needs of examining cerebral aneurisms are introduced [NJB*11].

There is an apparent emergence of approaches to improve the depiction of 3D integral lines or flow parameters on 2D planes. This, however, does not solve the problem of interpreting the flow in terms of flow patterns and overall flow behaviour, leaving its solution mostly to the user and making the analysis outcome heavily dependent on the user.

An approach for assisting the user in this task has been proposed by Heiberg et al. where swirling flow is automatically detected using a vector pattern matching technique [HEWK03]. Krishnan et al. visualize blood flow with similar paths by segmenting integral lines starting from a 2D plane based on their anatomical target area. The clusters are displayed on the emitter planes [KGG*12]. Targeting the quantification of blood flow, Eriksson et al. cluster pathlines according to their start and target regions into specific groups and derive the volume of the different compartments from this [ECD*10].

The aforementioned methods deal with one specific flow behaviour while a more flexible approach that allows to structure the flow according to several properties can be useful. Targeting this, Salzbrunn et al. present a line predicate approach for flow analysis [SGSM08, SS06] which is closely related to the pathline attributes introduced by Shi et al. [STH*09], linking views to select pathlines with specific parameters in non-medical data. Born et al. utilize line predicates to sort the set of integral lines capturing the complete flow dynamics into bundles with similar properties [BPMS12]. Our method builds on these ideas.

With respect to the enhancement of perception of line bundles, Banks et al. first proposed a method for lighting line primitives by treating them as infinitesimal cylinders and applying the maximum reflection principle [Ban94]. Mallo et al. later improved on this idea by simplifying the Blinn/Phong model for said cylinders and thereby improving the diffuse reflection model [MPSS05].

Our method for anatomical context visualization is based on the idea of Cipolla et al. defining the contour or silhouette of an object as the set of points on the object where the view direction is orthogonal to the surface normal [CG00].

3. Flow Visualization

In the following we shortly describe the underlying principles used for the flow visualization.

Unsteady velocity fields are defined as: $\mathbf{v} : D \times I \rightarrow \mathbb{R}^3$ with $\mathbf{v} \subset \mathbb{R}^3$ being a 3D vector field such that

$$\mathbf{v}(g,\tau) \quad = \quad (v_x(g,\tau), v_y(g,\tau), v_z(g,\tau))^T$$

where $g \in D$, and current time $\tau \in I = [t_0, t_n]$.

Pathlines depict the path of a virtual particle placed into an unsteady flow field at a certain time. It describes the curve $L(g,\tau)$ which is tangent to the vector field everywhere for a point at time τ. This means

$$\dot{L}(g,\tau) = \mathbf{v}(g,\tau) \quad (1)$$

Utilizing this definition, a pathline l passing point a at time τ can be defined as

$$\begin{aligned} l_{a,\tau} : I_{a,\tau} &\rightarrow D, \\ t &\mapsto l_{a,\tau}(t), \\ l_{a,\tau}(\tau) &= a, \\ \frac{\partial l_{a,\tau}}{\partial t}(t) &= \mathbf{v}(l_{a,\tau}(t),t) \end{aligned} \quad (2)$$

where $\mathbf{v}$ denotes the vector field and $I_{a,\tau} \subset I$ the maximal lifespan of the particle in D during I [SGSM08]. The last two terms in Eq. 2 describe an initial value problem to which a pathline can be considered a numerical solution.

Line Predicates are boolean functions that determine whether or not a pathline $l \in \mathcal{P}$ fulfills a certain criterion:

$$\begin{aligned} P : \mathcal{P} &\rightarrow \{\text{true, false}\}, \\ l &\mapsto P(p) \end{aligned}$$

where P is the line predicate and $\mathcal{P}$ represents the set of all integral lines to be examined.
Multiple predicates can be combined logically in order to create more complex queries. Consider a set set of predicates $\mathcal{S}$ containing predicates P_1 and P_2. These can be evaluated successively, connected by a logical operator, as follows:

$$\mathcal{S} = \{P_1 \wedge P_2, P_1 \wedge \overline{P_2}, \overline{P_1} \wedge \overline{P_2}, P_1 \wedge \overline{P_2}\} \quad (3)$$

4. Data

The data set at hand was provided by the **C**enter for **M**edical **I**mage Science and **V**isualization (CMIV) at Linköping University [BPE*15] and contains several discrete volume data sequences of length 41, where each item in the sequence represents one timestep in the cardiac cycle. A complete cardiac cycle in this data set takes about 1036.9 milliseconds, yielding a temporal resolution of 25.291 milliseconds. Of these sequences, two are needed to create the spatial context and the pathlines and are hence the most relevant ones for the application. The first data sequence contains one volume per timestep storing velocity (direction and length) vectors in a uniform grid with dimensions $112 \times 112 \times 48$, the blood flow at a certain point and time. The physical dimensions of the volume are $297.32mm \times 297.31mm \times 131.61mm$, resulting in a spatial resolution (voxel size) of $2.65mm \times 2.65mm \times 2.74mm$. The second relevant data sequence contains a binary volume per timestep with the same dimensions, defining a mask for the whole heart ($0 \mapsto$ outside the heart, $1 \mapsto$ inside the heart). In addition, the data set contains mask sequences for every vessel/chamber of the heart. This can be useful for separate rendering of the vessels/chambers (see Fig. 7) or region-based line filtering.

While the temporal resolution is high, spatial data is sparse in comparison to simulation-obtained data and includes noise, especially outside the heart. The noise outside the heart can be eliminated by a voxel-wise multiplication of the velocity volume with the binary mask. For the elimination of noise for measurements inside the heart more advanced techniques need to be utilized.

5. System

Our system is comprised of several components that were combined in order to provide an interactive tool that meets the requirements for 4D blood flow exploration and analysis. These components include customizable visualization for the anatomical context, line illumination that enhances spatial perceptibility of large sets of lines using colour to emphasise additional properties of the blood flow, and finally, we offer a predicate approach to line filtering and clustering for flow exploration and examination. An interaction interface is linked to a rendering window to support the exploration. A central aspect of the entire system is flexibility, it is easily extendable integrating new properties of interest. The single components of the system and the underlying methods are described below.

5.1. Spatial context visualization

We extract the heart's morphology from the binary mask volume using the marching cubes algorithm. The obtained geometry is then used to render the spatial context for the pathlines in form of a contour using illustrative rendering to improve the spatial perception of the shape of heart (see Fig. 1). We have chosen to use a silhouette and contour enhancement. Similar methods have been used e.g. in [DFRS03]:

Consider the view vector $\mathbf{V}$ from the camera towards the point p to be lit and the normal vector $\mathbf{N}$ of the surface at the fragment to be lit, both normalized and in view space. According to Cipolla and Giblin [CG00], the contour of a surface S consists of the set of those points fulfilling

$$\mathbf{N}(p) \cdot \mathbf{V}(p) = 0 \quad (4)$$

with $p \in S$. This method can be extended by calculating the angle $\theta = \arccos(\mathbf{V} \cdot \mathbf{N})$ between the view vector $\mathbf{V}$ and the surface normal $\mathbf{N}$ and rendering the contour as a gradient. The smaller the angle, the higher $\cos\theta$ will be. To render the contour of the mesh as spatial context we apply $1 - \cos\theta$ as alpha value $c_\mathbf{a}$. The contour can then easily be widened or narrowed by applying an adjustable expontent κ to $c_\mathbf{a}$

$$c_\mathbf{a} = (1 - \cos\theta)^\kappa \quad (5)$$

For a more illustrative rendering of the contour we can define an alpha threshold λ for $c_\mathbf{a}$ which will dismiss all values for $c_\mathbf{a}$ below λ and therefore create a binary contour similar to Cipolla's and Giblin's approach with the difference that here

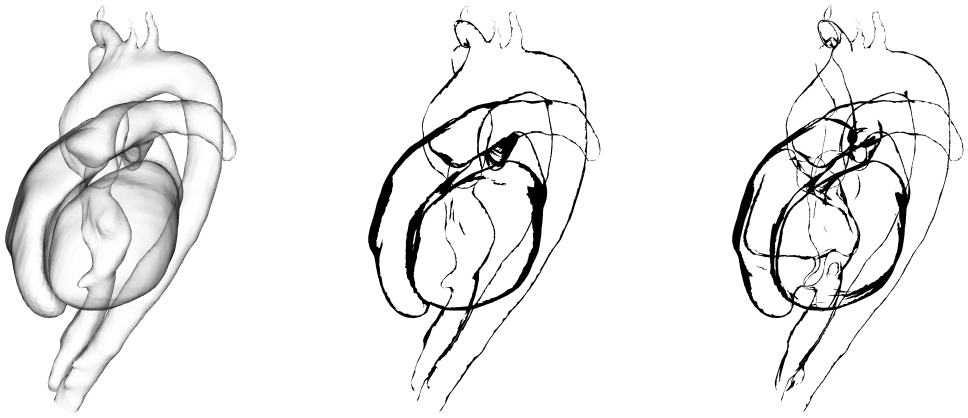


Figure 1: *Different rendering methods for the anatomical context. Left: Contour rendered using the standard method described by Eq. 5 with* $\kappa = 1.0$ *and full opacity. Center: Contour rendered binary according to Eq. 6 with* $\lambda = 0.4$*, full opacity and not displaying hidden structures. Right: Contour rendered binary showing occluded structures with* $\lambda = 0.3$ *and full opacity.*

the thickness of the contour can be controlled through λ:

$$F_{\mathbf{w}}(c_{\mathbf{a}}) = \begin{cases} 1 & \text{if } c_{\mathbf{a}} > \lambda \\ 0 & \text{if } c_{\mathbf{a}} < \lambda \end{cases} \tag{6}$$

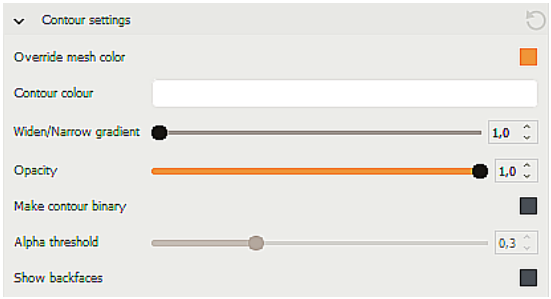


Figure 2: *Settings to control the spatial context rendering: If desired, the user can override the mesh colour. For non-binary contour rendering the exponent* κ *can be adjusted. For binary rendereing the alpha threshold* λ *controls the contour thickness. In order to reveal hidden structures, the user can also select to render hidden frontfaces.*

5.2. Line illumination

When rendering a large amount of lines, perceptability of orientation and arragement in space is often reduced. Applying proper lighting to the lines is used to remedy the problem. As one rendering option we have chosen to use illuminated streamlines as proposed by Banks et al. [Ban94].

With $\mathbf{P}$ as the position of the point to be lit, position $\mathbf{C}$ of the camera, and position $\mathbf{S}$ of the light source, we get the view vector $\mathbf{V} = \mathbf{C} - \mathbf{P}$ and the light vector $\mathbf{L} = \mathbf{S} - \mathbf{P}$. According to the Phong's lighting model [Pho75], light intensity $\mathbf{I}$ can be calculated as follows:

$$\mathbf{I} = \mathbf{I}_a + \mathbf{I}_d + \mathbf{I}_s = k_a + k_d \mathbf{L} \cdot \mathbf{N} + k_s (\mathbf{V} \cdot \mathbf{R})^n \tag{7}$$

with $\mathbf{R}$ as the reflection of $\mathbf{L}$ at $\mathbf{N}$, specular exponent n, and k_a, k_d, and k_s as ambient, diffuse, and specular coefficients.

Since a curve in 3D space does not have a uniquely defined normal at point $\mathbf{P}$ it needs to be calculated from the vectors $\mathbf{V}$, $\mathbf{L}$, and tangent $\mathbf{T}$ at $\mathbf{P}$. As the velocity vector defines the tangent of the integral lines the volume can simply be sampled at a desired position and time step in order to obtain the tangent needed for the aforementioned calculation.

A local coordinate frame $(\mathbf{T}, \mathbf{N}, \mathbf{B})$ is calculated from $\mathbf{V}$ and $\mathbf{T}$ [MPSS05], such that

$$\begin{pmatrix} \mathbf{T} \\ \mathbf{N} \\ \mathbf{B} \end{pmatrix} = \begin{pmatrix} \mathbf{T} \\ \mathbf{B} \times \mathbf{T} \\ \mathbf{T} \times \mathbf{V} / \|\mathbf{T} \times \mathbf{V}\| \end{pmatrix}.$$

The ambiguity in the choice of the normal vector is resolved by treating curves as infinitesimal cylinders and choosing the respective surface normal that maximizes the dot products for the specular and diffuse terms in Eqn. 7 [Ban94].

The diffuse reflection is independent of the viewing direction and according to Phong's model calculated by the dot product of the light vector $\mathbf{L}$ and the facet normal $\mathbf{N}_\theta$ with

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_T \\ \mathbf{L}_N \\ \mathbf{L}_B \end{pmatrix} = \begin{pmatrix} \mathbf{L}_T \\ \sqrt{1 - \mathbf{L}_T^2} \cos\alpha \\ \sqrt{1 - \mathbf{L}_T^2} \sin\alpha \end{pmatrix}, \quad \mathbf{N}_\theta = \begin{pmatrix} 0 \\ \cos\theta \\ \sin\theta \end{pmatrix}$$

in **TNB** space and with α as the angle between $\mathbf{N}$ and the projection of $\mathbf{L}$ onto **NB**. It is $\mathbf{L} \cdot \mathbf{N}_\theta = \sqrt{1 - \mathbf{L}_T^2} \cos(\theta - \alpha)$.

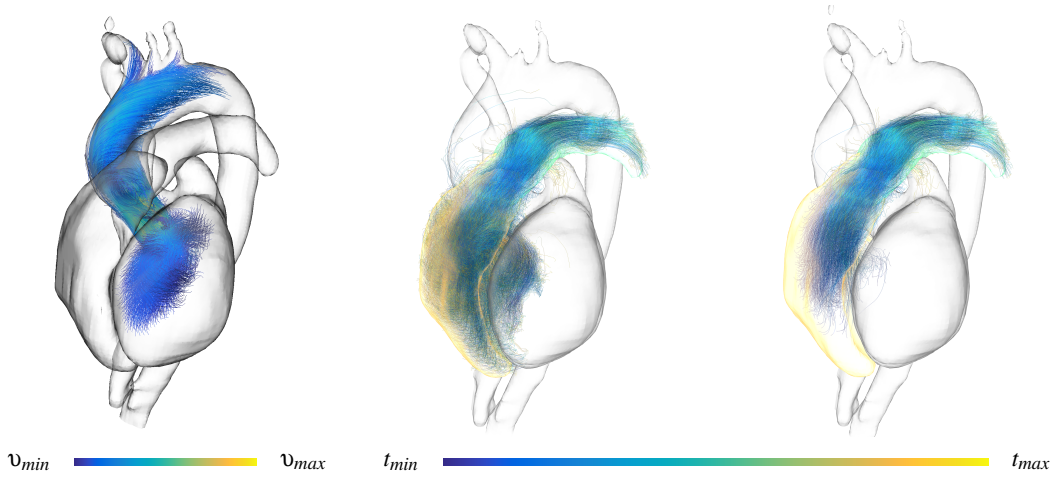


Figure 3: *Left: The lines in this figure start in the left heart chamber and depict the flow durnig systole. Short lines filtered out. Middle: Blood flow over a whole cardiac cycle originating from the right ventricle without applied filtering. The contour has reduced opacity, left ventricle is coloured yellow and pulmonary artery coloured green. Colour-coding by time. Right: Line set from the middle filtered with a ROI predicate, showing only the flow that reaches the pulmonary artery. Same contour and colour-coding.*

With $\alpha = \theta$, at the maximum, the diffuse term becomes:

$$\mathbf{I}_d = k_d\sqrt{1-\mathbf{L}_T^2}.$$

The specular term is given by the dot product of the view vector

$$\mathbf{V} = \begin{pmatrix} \mathbf{V}_T \\ \sqrt{1-\mathbf{V}_T^2} \\ 0 \end{pmatrix}$$

and the reflection vector $\mathbf{R}_\theta$ of $\mathbf{L}$ at $\mathbf{N}_\theta$

$$\begin{aligned} \mathbf{R}_\theta &= -\mathbf{L} + 2(\mathbf{L}\cdot\mathbf{N}_\theta)\mathbf{N}_\theta \\ &= \begin{pmatrix} -\mathbf{L}_T \\ \sqrt{1-\mathbf{L}_T^2}\cos(2\theta-\alpha) \\ \sqrt{1-\mathbf{L}_T^2}\sin(2\theta-\alpha) \end{pmatrix}. \end{aligned}$$

Since the thrid component of $\mathbf{V}$ is zero, the maximum for the specular reflection is reached at $\theta = \alpha/2$ which gives

$$\mathbf{I}_s = k_s\left(-\mathbf{V}_T\mathbf{L}_T + \sqrt{1-\mathbf{V}_T^2}\sqrt{1-\mathbf{L}_T^2}\right)^n$$

for the calculation of the specular term with n as the specular exponent.

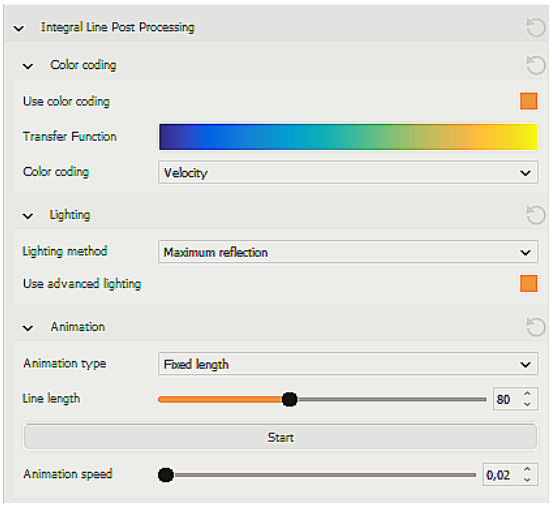


Figure 4: *Setting to control line illumination: If desired, the user can colour-code the pathlines according to an adjustable transfer function applied to selected line properties. Advanced lighting can be enabled/disabled, the lighting model can be chosen and pathlines can be animated, see 5.4.*

5.3. Line predicates based filtering

We use line predicates as central means for the filtering of the pathlines. We have exemplarily implemented some basic predicates. However, this set can be easily extended if required. The implemented predicates depend on line parameters, such as line length, maximum velocity and mean velocity. Vorticity $\omega = \nabla \times \mathbf{v}$ is related to turbulence in a flow field and therefore of special interest for flow analysis. The **R**egion **o**f **I**nterest (ROI) predicate checks if a line passes through a specified region. The presented parameters for am integral line l with velocity $\upsilon_i(l)$ at vertex $x_i \in \{0,\ldots,n\}$ at time t_i are defined below along with their respective predi-

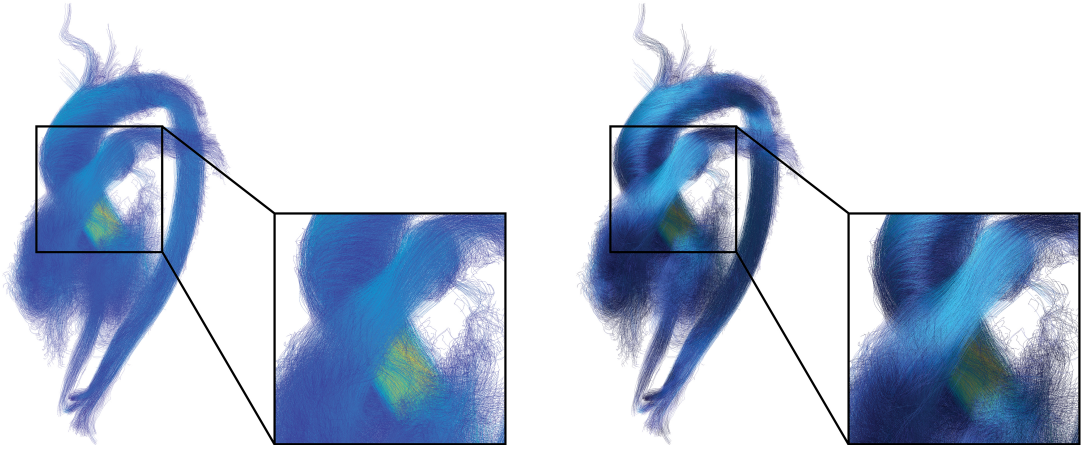


Figure 5: *Comparison of a line set directly rendered as line primitives by OpenGL and therefore without lighting (left) and the same set of lines rendered with applied lighting according to Sec. 5.1 (right)*

cate.

Length. The length of line l during time span $[t_1,t_2]$ is calculated as follows:

$$len(l) \quad = \quad \sum_{i=1}^{n} \|x_i(l) - x_{i-1}(l)\|$$

Mean velocity. The length of a line is direclty related to the mean velocity which is defined as the average velocity of l during time span $[t_1,t_2]$.

$$\upsilon_{mean}(l) \quad = \quad \frac{len(l)}{t_1 - t_2}$$

Maximum velocity. During the time span $[t_0,t_{n-1}]$ the maximum velocity of a line is defined as:

$$\upsilon_{max}(l) \quad = \quad max(\upsilon_0(l,t_0),...,\upsilon_{n-1}(l,t_{n-1}))$$

If we set one of these functions as $f(l)$ we can formulate a general evaluation function, or predicate, $P(l)$

$$P(l) \quad = \quad \begin{cases} true & \text{if } f(l) \in [f_1,f_2] \text{ during } [t_1,t_2] \\ false & \text{otherwise} \end{cases}.$$

Basically, $P(l)$ will evaluate to true if a line meets the user-defined criterion $f(l)$ within $[t_1,t_2]$.

Region of interest. The ROI predicate evaluates if a pathline l runs through a region, or set of points, $R = [x_0,x_n]$ at t_i:

$$P(l) \quad = \quad \begin{cases} true & \text{if } x \in l(t_i) \text{ and } x \in R \\ false & \text{otherwise} \end{cases}.$$

To provide an easy way to define new queries the system in cludes a parser that enables logical combination of predicates in order to create more complex queries. This could be queries like “Which lines reach a high velocity but have a short length?” or “Which lines reach a high vorticity during the systole and either have a low mean velocity or do not reach the aorta?”. The interface for the filtering consists of a pulldown menu for the available predicates and a second pulldown menu indicating the logical combination between two predicates (see Fig. 6). The options `STRONG AND` and `STRONG OR` were added so that a hierachy can be established that would be represented by brackets in a logical equation, for example

$$\mathcal{S} = (P_1 \wedge P_2) \vee P_3 \tag{8}$$

In our application, the logical operator between P_1 and P_2 would be a `STRONG AND` followed by an `OR` operator between P_2 and P_3.

5.4. Implementation Details

The implementation and data flow is designed to find a balance between complexity and interactivity. Therefore, the system can be divided into two parts: Relatively time-consuming pre-calculations performed on the CPU and interactive rendering on the GPU.

The pathline integration is done on the CPU using either the Euler, Heun, or 4th-order Runge-Kutta method. Adjustable parameters available in this process are the temporal step size, the number of steps forward and backwards, and the integration method. Also performed on the CPU is the pathline filtering using line predicates. Once the pathlines have been calculated the user can define the desired predicates and initiate the filtering (see Fig. 6).

Lighting, colour-coding and pathline animation are performed interactively using shaders programmed in GLSL. As shown in Fig. 4, the user can define the transfer function and set the colour-coding to velocity, time, or vorticity. Maximum values for velocity, length and vorticity are available per line on the GPU and used to ensure that the full range of the transfer function is used ideally.

Figure 6: *Settings available for the line predicates. The user can select predicates from a pulldown menu, their bounding values, evaluation time interval, and the logical combination between two predicates from a second pulldown menu. The options* `STRONG AND` *and* `STRONG OR` *were added so that a hierarchy can be established that would be represented by brackets in a logical equation (see Eq. 8).*

Advanced lighting of the pathlines can be turned off if more performance is needed or be set to the maximum reflection principle or the cylinder averaging approach presented in [Ban94]. Animation of pathlines can be done either in an evolving manner, meaning that the pathline will be drawn successively, or as pathlets, meaning short lines with a fixed line length, moving as a kind of wave along the pathlines. Indirectly, the flow can be shown as a particle stream if a very short length is set for the pathlets.

Adjustments to lighting, colour-coding, or animation are directly visible in the rendering at interactive framerates.

6. Results and Conclusion

The presented exploratory system was developed as a module for InViWo using the C++ programming language, supported by programmable shaders implemented in GLSL. The exploratory purpose of the system heavily depends on the accomplished interactivity. The GPU is used for the majority of the visualization methods and related calculations, allowing real-time interaction and customisation.

We have examplarily evaluated calculation times for pathline generation and line filtering for one 4D MRI blood-flow data set. The calculation time for pathline integration depends on the step size, number of steps, integration method and number of seedpoints. The generation of approximately 13,000 pathlines for a full cardiac cycle with a step size of 0,001 (ergo 1000 steps) took about 33.4 seconds using Euler's method and 61.6 seconds using the 4th-order Runge-Kutta integration scheme. We have then applied a combination of two and three predicates to this line set. Filtering took 20.7 seconds and 23.3 seconds, respectively.

Different options for the context rendering can be seen in Fig. 1. Fig. 7 depicts different modes for line rendering (tubes and illuminated lines). The illuminated lines are more appropriate for dense line representation, while the tubes can be used to show flow structures more representatively. An example for filtering using the length predicate can be seen in Fig. 3 as well as different colour-coding examples. Additionally, comparing the middle and left image in Fig. 3 it can be observed that not all blood from the left ventricle reaches the pulmonary artery within one cardiac cycle. This may be indicative of a heart muscle insufficiency.

In conclusion, we present a system that provides interactive 4D blood flow visualization, exploration and analysis. In particular, we offer an application that allows the user to filter out and search for flow structues of interest and helps the user to make sense of the data in a spatio-temporal context through depth-cues, anatomical context and animation.

References

[Ban94] BANKS D. C.: Illumination in Diverse Codimensions. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), SIGGRAPH '94, ACM, pp. 327–334. 2, 4, 7

[BB99] BOGREN H. G., BUONOCORE M. H.: 4D magnetic resonance velocity mapping of blood flow patterns in the aorta in young vs. elderly normal subjects. *Journal of Magnetic Resonance Imaging 10*, 5 (1999), 861–869. 2

[BBLM10] BARKER A., BOCK J., LORENZ R., MARKL M.: 4D flow MR imaging. *MAGNETOM Flash* (2010), 46–52. 2

[BPE*15] BUSTAMANTE M., PETERSSON S., ERIKSSON J., ALEHAGEN U., DYVERFELDT P., CARLHALL C. J., EBBERS T.: Atlas-based analysis of 4D flow CMR: automated vessel segmentation and flow quantification. *J Cardiovasc Magn Reson 17* (2015), 87. 3

[BPMS12] BORN S., PFEIFLE M., MARKL M., SCHEUERMANN G.: Visual 4D MRI blood flow analysis with line predicates. In *Pacific Visualization Symposium (PacificVis), 2012 IEEE* (Feb 2012), pp. 105–112. 1, 2

[Buo98] BUONOCORE M. H.: Visualizing blood flow patterns using streamlines, arrows, and particle paths. *Magn Reson Med 40*, 2 (Aug 1998), 210–226. 2

[CG00] CIPOLLA R., GIBLIN P.: *Visual Motion of Curves and Surfaces*. Cambridge University Press, New York, NY, USA, 2000. 2, 3

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive Contours for Conveying Shape. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22*, 3 (July 2003), 848–855. 3

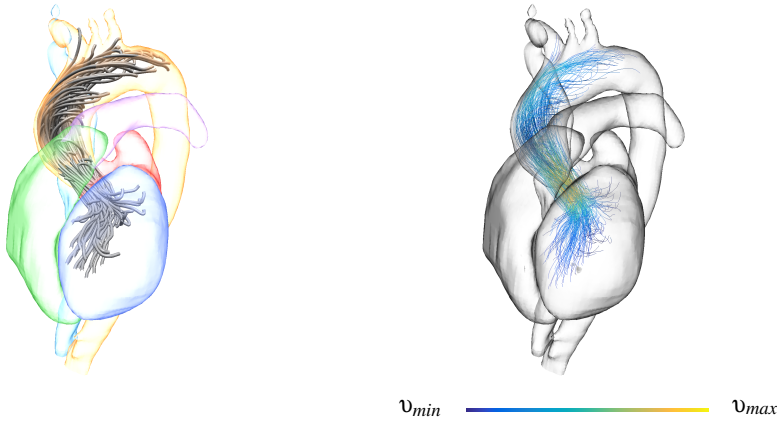


Figure 7: *Examples of different rendering and colouring options for the spatial context as well as the lines. The lines in this figure start in the left heart chamber and depict the flow durnig systole. Left: Each chamber and vessel is assigned a user-defined colour for better distinguishability of the heart structure. Lines are rendered as tubes to represent basic flow structures. Right: Anatomical context rendered rather subtly as black silhouette. Flow is represented by illuminated lines and colour-coded according to velocity.*

[ECD*10] ERIKSSON J., CARLHALL C. J., DYVERFELDT P., ENGVALL J., BOLGER A. F., EBBERS T.: Semi-automatic quantification of 4D left ventricular blood flow. *J Cardiovasc Magn Reson 12* (2010), 9. 2

[GNKP10] GASTEIGER R., NEUGEBAUER M., KUBISCH C., PREIM B.: Adapted Surface Visualization of Cerebral Aneurysms with Embedded Blood Flow Information. In *Eurographics Workshop on Visual Computing for Biology and Medicine* (2010), Bartz D., Botha C., Hornegger J., Machiraju R., Wiebel A., Preim B., (Eds.), The Eurographics Association. 2

[HEWK03] HEIBERG E., EBBERS T., WIGSTROM L., KARLSSON M.: Three-dimensional flow characterization using vector pattern matching. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (July 2003), 313–319. 2

[HSU*10] HEIBERG E., SJOGREN J., UGANDER M., CARLSSON M., ENGBLOM H., ARHEDEN H.: Design and validation of Segment–freely available software for cardiovascular image analysis. *BMC Med Imaging 10* (2010), 1. 2

[InV15] INVIWO: InViWo Overview, 2015. 2

[KGG*12] KRISHNAN H., GARTH C., GUHRING J., GULSUN M. A., GREISER A., JOY K. I.: Analysis of time-dependent flow-sensitive PC-MRI data. *IEEE Trans Vis Comput Graph 18*, 6 (Jun 2012), 966–977. 2

[KYW*00] KILNER P. J., YANG G.-Z., WILKES A. J., MOHIADDIN R. H., FIRMIN D. N., YACOUB M. H.: Asymmetric redirection of flow through the heart. *Nature 404*, 6779 (Apr 2000), 759–761. 1

[MCA*03] MARKL M., CHAN F. P., ALLEY M. T., WEDDING K. L., DRANEY M. T., ELKINS C. J., PARKER D. W., WICKER R., TAYLOR C. A., HERFKENS R. J., PELC N. J.: Time-resolved three-dimensional phase-contrast MRI. *J Magn Reson Imaging 17*, 4 (Apr 2003), 499–506. 2

[MKE11] MARKL M., KILNER P. J., EBBERS T.: Comprehensive 4D velocity mapping of the heart and great vessels by cardiovascular magnetic resonance. *Journal of Cardiovascular Magnetic Resonance 13*, 1 (2011), 1–22. 2

[MPSS05] MALLO O., PEIKERT R., SIGG C., SADLO F.: Illuminated lines revisited. In *Visualization, 2005. VIS 05. IEEE* (Oct 2005), pp. 19–26. 2, 4

[NJB*11] NEUGEBAUER M., JANIGA G., BEUING O., SKALEJ M., PREIM B.: Anatomy-Guided Multi-Level Exploration of Blood Flow in Cerebral Aneurysms. *Computer Graphics Forum 30*, 3 (2011), 1041–1050. 2

[Pho75] PHONG B. T.: Illumination for Computer Generated Pictures. *Commun. ACM 18*, 6 (June 1975), 311–317. 4

[SAG*14] STANKOVIC Z., ALLEN B. D., GARCIA J., JARVIS K. B., MARKL M.: 4D flow imaging with MRI. *Cardiovascular Diagnosis and Therapy 4*, 2 (2014). 1

[SGSM08] SALZBRUNN T., GARTH C., SCHEUERMANN G., MEYER J.: Pathline predicates and unsteady flow structures. *The Visual Computer 24*, 12 (2008), 1039–1051. 1, 2, 3

[SS06] SALZBRUNN T., SCHEUERMANN G.: Streamline Predicates. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (Nov 2006), 1601–1612. 1, 2

[STH*09] SHI K., THEISEL H., HAUSER H., WEINKAUF T., MATKOVIC K., HEGE H.-C., SEIDEL H.-P.: *Topology-Based Methods in Visualization II*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, ch. Path Line Attributes - an Information Visualization Approach to Analyzing the Dynamic Behavior of 3D Time-Dependent Flow Fields, pp. 75–88. 2

[vPBB*10] VAN PELT R., BESCOS J. O., BREEUWER M., CLOUGH R. E., GROLLER M. E., TER HAAR ROMENIJ B., VILANOVA A.: Exploration of 4D MRI Blood Flow using Stylistic Visualization. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov 2010), 1339–1347. 2

[vPBB*11] VAN PELT R., BESCOS J. O., BREEUWER M., CLOUGH R. E., GROLLER M. E., TER HAAR ROMENIJ B., VILANOVA A.: Interactive Virtual Probing of 4D MRI Blood-Flow. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec 2011), 2153–2162. 2

[Wor15] WORLD HEALTH ORGANIZATION: Global Health Observatory visualizations, Causes of death - Ten leading causes of death, 2012, 2015. 1

SIGRAD 2016
M. Hayashi (Editors)

Automatic CG Talk Show Generation from the Internet Forum

Masaki Hayashi, Steven Bachelder & Masayuki Nakajima

Uppsala University, Campus Gotland, Visby, Sweden

Abstract

We have developed an application to produce Computer Graphics (CG) animations in TV talk show formats automatically from the Internet forum. First, an actual broadcasted talk show is analyzed to obtain data in regards to camera changes, lighting, studio set up, etc. The result of the analysis is then implemented into the application and a CG animation is created using the TV program Making Language (TVML). The application works in the Unity game engine with CG characters speaking with computer generated voices. We have successfully created a CG generated TV talk show which allows users to "watch" the TV show format generated from the text information coming from the forum on the Internet.

1. Introduction

Due to the advancement of Computer Graphics (CG), and wide spread use of the Internet, many consumer services (e.g. Plotagon [Plo16]) have emerged supporting User Generated Content (UGC). The creation of CG animations is part of this development.

To facilitate UGC further, we have been developing a system which is capable of creating TV-show-like CG animations automatically from text based internet sources [HID*14]. Our key concept is to convert various sources on the Internet into text-based scripts. These then are represented by the CG animations created by the scripts processed in the TV program Making Language (TVML)[www.nhk.or.jp/strl/tvml/english/player2/]. The TVML is the base technology that the author uses to facilitate the development of automatic CG content creation from the text sources.

In this paper, we introduce an attempt to generate a TV-show-like CG Talk Show made by the TVML by converting a HTML script taken from the forum on the Internet. The conversion is done fully automatically so that users can actually "watch" the forum by just clicking a button in the application. The animation is made by real-time CG where CG characters with computer generated voices are discussing with appropriate gestures. The use of camera switching is calculated by the system. Our approach to do this is to first analyze a recorded broadcasted talk show on TV, and then to extract rules and statistical behaviors, and to ultimately implement these as an algorithm in the application in order to generate the CG animation which resembles the original format.

2. Analysis and Algorithm

Figure 1: *Reference talk show on TV (broadcasted on Jan. 1989)*

We picked up a series of Japanese talk shows on TV (see Fig. 1) recorded for our analysis. Through the observation of the reference talk show in terms of who is speaking, camera angles, an elapsed time, we formulated the following process (see Fig. 2).

- Phase-1: To determine whether a camera switching occurs when a speaker changes.
- Phase-2: To determine the next camera angle when the camera switching occurs.

Fig.2 shows the algorithm of Phase-1. There are two factors in Phase-1: "probability of camera switching when a speaker changes" and "duration of certain camera angles".

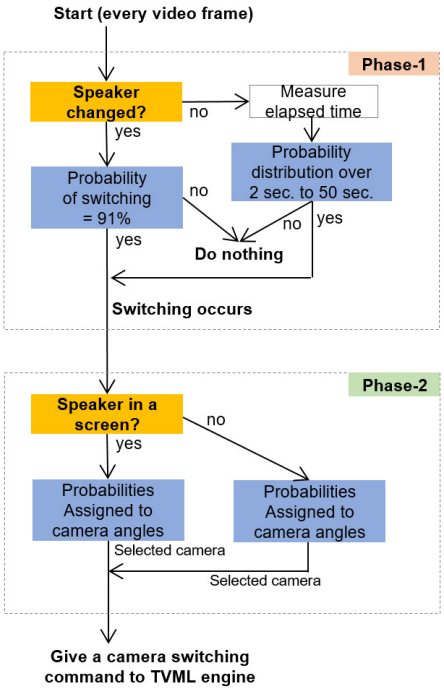


Figure 2: *Algorithm of camera switching.*

When it comes to Phase-2, the camera angles are classified into these six types: "upshot of a speaker", "upshot of a previous speaker", "two-shot of a speaker and a previous speaker", "group shot including a speaker", "shot without including a speaker", "other shot". When the camera switching trigger is sent from Phase-1 to Phase 2, it first checks whether the current speaker is filmed in the current camera angle, then determines the next camera angle based on the possibilities assigned to the next camera angles. We analyzed a video recorded in approximate one hour (number of shot is 264) to determine the factors stated above. The results indicated the probability of switching in Phase-1 is 91The probabilities in Phase-2 are shown in Fig. 3.

	Speaker on a screen	speaker not on a screen
upshot of speaker	51%	73%
upshot of previous speaker	13%	3%
two-shot of speaker and previous speaker	4%	3%
group shot including speaker	4%	6%
shot without including speaker	21%	10%
other shot	7%	5%

Figure 3: *Indicated camera angle possibilities.*

3. Implementation and a Test

The TVML SDK in the Unity3D game engine is used to make the TV-show-like CG animation. It is thus able to create real-time CG animations from a scripts written in TVML. This is done by using CG characters with computer generated voices, superimposing texts, image file displays, movie file playbacks, audio file playbacks, etc.

The algorithm described in Chapter 2 is implemented as a C# program in Unity. Input into the program is the status of "who is speaking" which is obtained from the SDK and an elapsed time sequence measured in the program. The output is a camera switching command given to the TVML engine. The calculation is made in every video frame to make a decision in regards to camera switching by using the probabilities as shown in Fig.3 with random values.

The source for making the talk show animation was the Japanese forum called "2 CHANNERU" [http://www.2ch.sc/]. The reproduced talk show has six CG characters that each post in the forum is assigned to randomly. The necessary camera angles are pre-defined in a TVML script. The camera angles consist of "upshot of each character" (number of the camera is 6), "two-shot of given two characters" (number of the camera is 15) and "other shot" (1 group shot), resulting in a total number of camera angles of 22.

Besides the camera switching described here, character gestures are also added based on a simple heuristic algorithm. The characters change view direction and occasionally nod in real-time to make the animation more realistic.

Fig. 4 shows screenshots of the produced CG talk show. the generated talk show successfully reproduces the desired resemblance to camera switching observed in the original format. The video can be seen at https://youtu.be/twafRtxW1Dk.

Figure 4: *Screenshots of reproduced animation*

4. Conclusion

We have developed a system reproducing a CG talk show with TVML based on the analysis of a real broadcasted TV show. We analyzed the recording of an actual TV talk show

for one hour and extracted the rules and statistical characteristics of camera switching. These were then used to create an algorithm for automatic camera switching. This algorithm was then implemented to TVML SDK in the Unity game engine in order to provide a platform of an automatic talk show generation application. We used the Japanese forum to convert the posts to a TVML script with six CG characters randomly assigned to the different posts. With our implemented algorithm of camera switching and character gestures, the final CG talk show was created.

Our next aim will be to further verify the system, by conducting comprehensive user evaluation test measuring viewer engagement. We plan also to enhance the algorithm for not only camera switching but also camera movement, enabling control of these parameters also in order to further articulate camera movement in TV show formats (e.g. like political debates, stand up comedy, music shows, etc.). We will also focus on the further development of character behavior and movement.

References

[Plo16] Plotagon: http://plotagon.com/ (Accessed Apr. 2016). 1

[HID*14] HAYASHI M., INOUE S., DOUKE M., HAMAGUCHI N., KANEKO H., BACHELDER S., NAKAJIMA M.: T2V: New Technology of Converting Text to CG Animation. *ITE Transactions on Media Technology and Applications 2*, 1 (Dec. 2014), 74–82. 1

SIGRAD 2016, pp. 1–3
M. Hayashi (Editors)

Digital archive of the SHISHIMAI using AR Toolkit

Y.Watanabe[1] and H. Tsujiai[2]

[1]ARCGEO Inc, Japan
[2]University of Toyama, Japan

Abstract

We made a digital archive of SHISHIMAI using an AR Toolkit. We used paper craft, using two markers for an AR toolkit to recreate the appearance of SHISHIMAI. We chose the SHISHIMAI of Toyama to archive digitally. Thus, we proposed a new use of AR technology. In addition, using AR, we created opportunities for the public to easily touch a lion mask used for traditional entertainment.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Graphics systems and interfaces—Mixed / augmented reality

1. Introduction

SHISHIMAI, the Japanese traditional lion dance, is performed by a man wearing a lion mask called a SHISHIGASHIRA. Similar dances are also seen in China and Singapore. We produced 3D data of a lion mask used in SHISHIMAI. Furthermore, we used an AR Toolkit to create a digital archive of the data.

1.1. SHISHIMAI of Toyama

SHISHIMAI of Toyama, Japan, is rich in variation and it still has an inherited style in every region. Almost every Toyama lion mask is made by applying the traditional style [Yas12]. The lion mask handed down from the old days is Toyama prefecture's designated cultural property [Him11,TA90]. We believe that the Toyama SHISHIMAI also has digital-archive value as a cultural art asset.

1.2. New style of digital archive

A digital archive is data material held by a library or museum and used for data preservation; furthermore, it can be accessed through the internet [Tos12]. We used the AR Toolkit to make a digital archive system. Using AR, we can see the size of a digital collection on a human scale. Additionally, we can virtually exhibit a digital collection, such as a mask or accessories.

2. AR SHISHIMAI

This digital archive system uses the AR Toolkit. The interface also uses paper craft, which has two AR markers set as the upper and lower jaws of a lion mask. When the paper craft is placed in front of the camera, a lion mask is shown on the AR marker printed on the paper craft. As a player moves the paper craft, the virtual lion mask moves the same way, so the player can virtually experience the SHISHIMAI.

2.1. The interface of AR SHISHIMAI

We used paper craft that printed two AR Toolkit markers to AR SHISHIMAI. The paper craft's advantage is its low cost and ease of making spares. These features make setting up the digital archive system easy. Therefore, it is found to be a most useful interface [YA12a].

3. 3D scanning of the lion mask

To create 3D data on the digital archive for AR SHISHIMAI, we used a 3D laser scanner, specifically the DAVID LASER SCANNER [Dav14]. It scans shapes with a laser beam, using a camera and a calibration panel, scanning surface texture from the camera.
We scanned four lion masks, two in Habiro, Takaoka city, Toyama; one at the Futagami Imizu shrine, Toyama; and one in Futagami, Takaoka city, Toyama. We almost succeeded in scanning these lion masks' shapes entirely. However, we could not scan three masks perfectly. The lion mask in Habiro was newly printed and therefore glossy; however,

submitted to *SIGRAD (2016)*

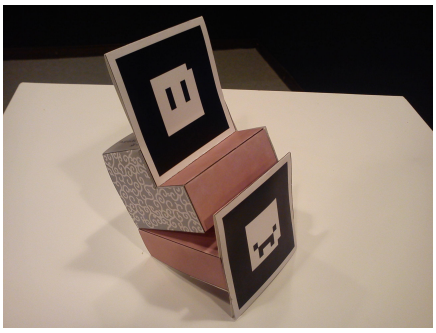


Figure 1: *The interface of AR SHISHIMAI. Paper craft that printed the two AR Toolkit markers.*

the scanning laser did not work well on the glossy surface. Finally, we did scan the lion mask's shape, but the data was of poor quality. The lion mask at the Futagami Imizu shrine has a black surface, which does not allow the laser to reflect well. Thus, it was also difficult to scan. The lion mask in Futagami, has both a black and glossy surface. Therefore, of the four masks the data was of the poorest quality in this mask.

4. Exhibition of AR SHISHIMAI

We exhibited AR SHISHIMAI in the public gallery of Takaoka city, Toyama, and other places. Because visitors can hold a lion mask virtually, they become interested in SHISHIMAI.

Figure 2: *We exhibited the lion mask in Habiro on AR SHISHIMAI. Although we could scan the lion mask' s shape, it was of poor quality.*

5. Conclusion

AR SHISHIMAI is a new application field for AR technology. AR SHISHIMAI can make visitors virtually close to collections, and it is easy to set because we prepare just a PC, a camera, and printed paper. Digital AR archives are not limited to SHISHIMAI, the lion dance, but can be used for other collections like weapons or clothing. However, we know many problems remain in 3D scanning and also in the system.

6. Acknowledgments

We would like to thank Professor Norio Horie and Professor Kazumi Uchida for their advice and guidance. About the study of SHISHIMAI, we thank our practical professor Shimazoe Kimiko, who told us the basics about SHISHIMAI. Moreover, we were taught many kinds of SHISHIMAI in Toyama. About AR technology, we thank Tomohiro Nakayama, the design office art laboratory representative, who told us about the study of AR technology.

Figure 3: *We exhibited AR SHISHIMAI in the public gallery of Takaoka city, Toyama [YH12b, Yuy13a, Yuy13b].*

References

[Yas12] Yasukazu Saeki, The phase of the folk customs of Toyama - Cooking, SHISHIMAI, Article of everyday use, Annual event, Gokayama, And so on : Toyama, Katsura bookshop, 2012(in Japanese) 1

[Him11] Himi City Museum,SHISHIMAI special exhibition Himi -Danced SHISHIMAI and isn't danced SHISHIMAI- :Toyama, Katsura bookshop , 2011(investigated in 1983)(in Japanese) 1

[TA90] Tadashi Nishimura and Akinori Katagishi, SHISHIMAI investigation in Fukumitsu Town: Toyama. Fukumitsu Shinkin Bank, 1990(in Japanese) 1

[Tos12] Toshiaki Baba, Library information outline of resources JLA library and information science text Series III: Toyam, Japan Library Association, 2012(in Japanese) 1

[YA12a] Y. Watanabe and H. Tsujiai, The Interface using two AR Toolkit markers: Chubu University. Joint Conference of Hokuriku Chapters of Electrical Societies (JHES2012). 2012(in Japanese) 1

[Dav14] DAVID 3D Scanner, http://www.david-laserscanner.jp/, (We access on January 15, 2014.) 1

[YH12b] Y.Watanabe and H. Tsujiai, Reappearance of SHISHIMAI using ARToolkit: Kunibiki Messe. Conference Proceedings of the 10th Asia Pacific Conference on Computer Human Interaction(APCHI2012), 719-750, 2012 2

[Yuy13a] Y.Watanabe,AR-ization of SHISHIMAI: Geibungallery, Inseiten(Graduate student exhibition),January 31 - February 12, 2013(in Japanese) 2

[Yuy13b] Y.Watanabe, AR SHISHIMAI:University of Toyama, University of Toyama and PATANASHIN university of arts exchange show,December 4 - 17, 2013 2

SIGRAD 2016
M. Hayashi (Editors)

Towards Full-Scale Ray Tracing in Games

Afshin Ameri E. and Thomas Larsson

Mälardalen University, Sweden

Abstract

We discuss the current status when it comes to real-time ray tracing of games as a full-scale alternative to renderers based on rasterization. Modern games include massive geometry, beautiful graphics, and advanced rendering effects, and still, they run with a high and steady frame rate at high resolutions. We argue that to make ray tracing a viable option, significant further development is required in terms of improved algorithms and software libraries, as well as hardware innovations that will greatly benefit the average gamer's hardware equipment.

1. Introduction

Ever since our PCs began to house parallel features such as instructions pipelining, superscalar execution, SIMD units, and multi-core CPUs, game developers and other programmers have nurtured the idea of having real-time ray tracing at their disposal. Ray tracing is often referred to as being embarrassingly parallel, since each pixel can be generated independently and in parallel with the others by firing independent rays that travels through acceleration data structures, which are either pre-built or reconstructed on the fly. There are ample of opportunities for data-parallel computations that can be distributed over a large number of cores, both during ray traversals and shading calculations.

During the last decade, researchers have tried to realize real-time ray tracing for the masses. Outspoken claims of real-time performance have not been rare (see e.g. [FGD*06, Bik07] and the references therein). It has turned out, though, that despite impressive algorithmic innovations and a substantial improvement of hardware performance, these solutions deliver real-time rates only when applied in certain contexts, such as using low resolutions, simplified lighting models, friendly scene compositions, or high-end hardware.

When will ray tracing become a full-scale alternative to rasterization in game rendering? What kind of hardware is required? What software libraries can make a real difference? These are difficult questions. The fact that rasterization is the predominant rendering technique in commercial games preserves status quo. Games have driven the GPU market to unprecedented levels during the latest 20 years. The introduction of programmable shader stages in the graphics pipeline in 2001 is the single most important step that has kept rasterization at the forefront of game development. Established and standardized APIs that undergo timely and profound modernization further reinforce rasterization as game developer's first choice.

Indeed, the modern GPU is excellent at data-parallel processing of both triangles and pixels, which makes rasterization blazingly fast. With resolution and frame rate standards such as 1920×1080 and 60 fps, it is very challenging to keep up with rasterization for any fundamentally different strategy. Modern rasterizers also use lighting models that incorporate anti-aliasing, shadows, reflections, ambient occlusion, etc. A Whitted-style ray tracer would simply not be enough. As more advanced global illumination effects are supported by game engines, sometimes by using hybrid techniques that include ray tracing ingredients, a competing real-time ray tracer must probably support some type of distribution ray tracing, path tracing, or photon mapping. The extreme workloads this put onto the hardware are well-known in off-line renderers used to produce animated movies.

There are some lights on the horizon, though. First of all, ray tracing naturally supports visibility queries to generate realistic secondary lighting effects: shadows, reflections, and refractions. By hierarchical index methods, the expected time complexity for ray shooting n primitives is $O(\log n)$ for fixed resolution. Furthermore, it is convenient to support a mix of different types of modelling primitives, and advanced effects can be added adaptively by shooting more rays. Rasterization techniques, on the other hand, give cruder approximations of such effects and usually require more parameter tuning to avoid rendering artefacts.

Another noteworthy aspect is that major hardware companies show a lot of interest in ray tracing by developing libraries, e.g., OptiX by Nivida [PBD*10], Embree by In-

Figure 1: *Example scene rendered in full HD by a simple ray tracer that uses the Embree kernel framework.*

tel [WWB*14], and FireRays by AMD. This is probably because graphics and visualization are becoming increasingly important, and ray tracing can also be used to show off the performance of their latest hardware inventions.

2. Experiment

What is the current performance when it comes to using high-performance ray tracing libraries? To get an idea of the rendering speed, we created a simple ray tracer using Embree v.2.9 [WWB*14] and Visual Studio 2013. Embree was initialized to use packets of 8 rays for all ray types. Image tiles (16×16) were rendered in parallel by using OpenMP multithreading with dynamic scheduling. Only a simple Phong illumination model was used. We ran our test on a PC with an Intel Core i7-5960X 3.00 GHz CPU (8 cores with AVX2 support) and 16 GB RAM.

The scene had a total face count of 517 600 and it was composed of a house (created by Herminio Nieves) and a Mercedes-Benz SLS AMG (downloaded from tf3dm.com). There was one directional light source, and some texture maps were used. Figure 1 shows an image of this scene which was rendered at a resolution of 1920×1080. In this particular case, our rendering performance was 7.8 fps, which corresponds to 34.2×10^6 rays/s. When we let the camera orbit around the house, the fps varied from 6.0 to 20.4 with an average of 10.8. We expect that the performance could be improved further by using more aggressively optimizing compilers, e.g., the Intel SPMD Program Compiler (ispc), and by additional code optimizations and parameter tuning, but we would probably not be able to reach 60 fps, assuming the same hardware setup is used.

3. Challenges

There is an interesting old joke that says: "Ray tracing is the technology of the future and it always will be!" To get high-quality images, we need accurate shadows and reflections from multiple light sources, as well as anti-aliasing and some ambient occlusion. Let's assume that a rough estimate of 100 rays/pixel would be enough. Now consider full HD resolution and a target frame rate of 60 fps. Furthermore, assume the computing resources are occupied by other computations concerning game physics, collision detection, artificial intelligence, etc., half of the time. This leaves about 8 ms for rendering each image. Under these circumstances, we need a ray tracing budget of around 25 billion rays/s. If we compare this with the performance reached here, or with the rendering speeds reported elsewhere [WWB*14, PBD*10], there is a substantial difference.

The road ahead requires improved algorithms and more efficient parallel hardware. Algorithmic improvements need to focus on more efficient data structures for handling general scenes with dynamic features and faster ways to reproduce global illumination effects. On the hardware side, strong cooperating heterogeneous chips are needed as well as highly optimized kernel libraries that fully utilizes the architectures, e.g., by online learning and autotuning for increased efficiency. In addition, the solutions have to be applicable on affordable hardware for the average gamer.

For 50 years, we have been able to rely on Moore's law for doubled transistor density leading to an exponential rate of performance improvement. It has been described as a "miraculous development", but the rate of progress has started to decelerate as the transistor size is reaching closer to its physical limit. This may seem like a threat against the high performance we are eagerly seeking, but it may in fact be the opposite; it may trigger innovation and lead to even better hardware designs. Interestingly, Imagination Technologies recently demonstrated that dedicated low-power ray tracing hardware can deliver 300 million rays/s.

Even so, there is still a big performance gap to overcome before ray traced games can take over. In fact, the performance requirement specified above are relatively modest in the sense that it is easy to imagine even higher image resolutions and a wish for more accurate global illumination techniques. Without doubt, full-scale real-time ray tracing will be a main challenge for many years to come.

References

[Bik07] BIKKER J.: Real-time ray tracing through the eyes of a game developer. In *IEEE Symposium on Interactive Ray Tracing* (2007), pp. 1–10. 1

[FGD*06] FRIEDRICH H., GÜNTHER J., DIETRICH A., SCHERBAUM M., SEIDEL H.-P., SLUSALLEK P.: Exploring the use of ray tracing for future games. In *ACM SIGGRAPH Symposium on Videogames* (2006), pp. 41–50. 1

[PBD*10] PARKER, BIGLER, DIETRICH, FRIEDRICH, HOBEROCK, LUEBKE, MCALLISTER, MCGUIRE, MORLEY, ROBISON, STICH: OptiX: A general purpose ray tracing engine. *ACM Trans. Graph. 29*, 4 (2010), 66:1–66:13. 1, 2

[WWB*14] WALD I., WOOP S., BENTHIN C., JOHNSON G. S., ERNST M.: Embree: A kernel framework for efficient CPU ray tracing. *ACM Trans. Graph. 33*, 4 (2014), 143:1–143:8. 2

SIGRAD 2016
M. Hayashi (Editors)

Pattern Generation with Cellular Automata in Hexagonal Modular Spaces

M. Fridenfalk

Department of Game Design, Uppsala University, Sweden

Abstract

This paper presents new methods for the generation of hexagonal patterns, based on cellular automata in small-sized regular hexagonal modular spaces. The patterns are intended to be used for procedural content generation in computer games, but could also be applied for diverse ends, such as logotype design and architecture.

1. Introduction

The concept of cellular automata was introduced by Stanislaw Ulam and John von Neumann in mid 1900s, followed by the popularization of the theory by John Conway in 1970 and Stephen Wolfram in 2002 [Wol02]. Presently, the range of applications is relatively extensive, covering diverse areas, such as from the generation of visual arts, to theory building in physics and computational biology [Gar70, Wol02].

Presently, the most well known cellular automaton is known as Conway's "Game of Life" [Gar70]. The rules of this automaton can be summarized as: a cell (or node) is activated (set to one) if it has three neighbors and deactivated (set to zero) if it has zero, one, or four to eight neighbors, else current value is preserved. Game of Life can be classified as a binary automaton in a Moore neighborhood, where each cell has eight surrounding neighbors in a 2D square lattice.

Although cellular automata are typically implemented as discrete elements within finite computational systems, in practice there are no limitations on the domain, range or the size of a cellular automaton. One of the many options available in the design of cellular automata is the configuration of the neighborhood connectivity map (in this paper, synonymous with the adjacency list), e.g., the application of non-square lattices, such as a regular hexagonal lattice, instead of a square lattice [Ada10, ASM06, GBM08, LBMSTM11].

2. Nonlinear Cellular Automata

This work is part of a systematic study of the implementation and evaluation of small-sized modular cellular automata [Fri13, Fri15], in the general case classified as Nonlinear Cellular Automata (NCA), in the sense that it is the space on which the automaton operates on that is not assumed to be linear or infinite, but modular or even highly irregular, or alternatively, a space defined by nodes on a graph where the nodes are not always connected to their closest neighbors, but where the connectivity list or matrix of the graph defining the space, often has to be generated procedurally.

These new types of cellular automata, as presented in [Fri13, Fri15], are thus typically characterized by a high reproduction rate, in combination with the application of small-sized modular lattices, with application areas such as computer game development, visual arts, logo design, textile pattern design, and architecture. Presently, this method has been used for the generation of chess variants in a computer game [Fri13], and is planned to be used in future game development projects.

3. Proposals

In the method presented in this paper for pattern generation, each cell is connected to its six closest neighbors in a hexagonal grid. To maintain a high reproduction rate, we apply the rule in Volume I in [Fri15], where a cell is activated (set to one) if it has two or three neighbors or else deactivated (set to zero). To initialize a cellular automaton, two kernels are applied, as shown in Figure 8, placed at the center of the hexagonal space.

In the design of regular hexagonal modular spaces, based on hexagonal cells (in this paper represented as circular dots), two models were designed and implemented. The first model is presented in Figure 3, along with a proposal for the neighborhood connectivity map in Figures 5-6. As shown

Figure 1: *Example of hexagonal grids within rhombic modular spaces.*

Figure 2: *The selected model for implementation in this work, based on a regular hexagonal modular space, in this example with a size of $n = 5$ cells. Such a grid may be managed by a standard $n \times n$ matrix through the representation of the rhombic structure by a square-based equivalent.*

here, this model requires a small shift for each modular space, yielding a clockwise or counterclockwise neighborhood connectivity pattern. The second model is based on an aligned pattern (thus eliminating the rotational effect caused by the first model), as presented in Figure 4, along with a proposal for the neighborhood connectivity map in Figure 7. An algorithm was developed for each model for the generation of the connectivity map for an arbitrary size of the modular space, or more specifically for any $s \in \mathbb{N}_2$ (i.e., all integers larger than one) with $n = 2s - 1$, where s denotes the length of the side of the hexagon, and n, the largest width of the hexagon (in this paper referred to as size), both counted in the number of cells.

To extend the algorithm for the generation of integer values instead of Boolean, in this work, as a post-processing effect, each generation is represented as the addition of the last two generations in question, yielding a trinary grid instead of a binary, using the binary values of the last generation as a stencil. One way to convert all grids to trinary ones based on this technique is thus to start with the last generation and iterate backwards to the first.

4. Results

The results of this work are presented in Figures 9-12. Figure 9 shows the progression of kernel 1 (i.e., $k = 1$ in Figure 8), within a hexagonal modular grid of size $n = 17$, from the first to the 59:th generation, based on the aligned model, as schematically defined in Figures 4 and 7. As indicated by Figures 9-12, while smaller hexagonal grids, as a rule, yield a lower number of generations for each kernel before falling into a cyclic loop, making the cellular automaton presented in Figure 9, to fall into a cyclic loop after 53 generations (since generation 12 is identical to 54), while there exist no cyclic loops for the larger cellular automata

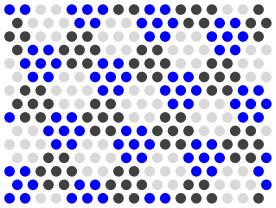


Figure 3: *The primary model designed and implemented in this work. As shown here, a modular connectivity map without any overlapping cells can only be implemented by a clockwise or counterclockwise shift of each hexagonal modular space.*

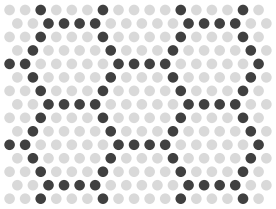


Figure 4: *The aligned model designed and implemented in this work, as a complement to the primary model. In this model, the borderline cells of the modular spaces are evenly shared.*

with $n = 23$, within the presented generation intervals. Figures 10-12 show the progression of kernels 0 and 1 for some selected hexagonal grids, based on the nonaligned model, as schematically defined in Figures 3 and 5-6. As mentioned above, the patterns in these figures are post-processed, yielding trinary results, thereby represented by black, white, and blue dots.

5. Conclusion

The application of modular spaces to hexagonal lattices showed to be more complex compared to square lattices, since instead of a single and relatively straightforward determination of the neighborhood connectivity map, at least two feasible configurations were found, both yielding relatively lengthy algorithms for the generation of each connectivity map. Further on, the outcome of our experiments did not show to qualitatively exceed the ones previously obtained by corresponding methods based on square lattices, other than adding to the variation of previous methods.

6. Future Work

A next step in the progression of this work could consist of the implementation of similar methods, but for irregu-

lar hexagonal modular spaces, where not all sides are of equal length. Another example is the implementation of non-hexagonal modular spaces to hexagonal lattices, such as the example presented in Figure 1.

References

[Ada10] ADAMATZKY A. (Ed.): *Game of life cellular automata*. Springer, 2010. 1

[ASM06] ALONSO-SANZ R., MARTIN M.: A structurally dynamic cellular automaton with memory in the hexagonal tessellation. In *Proc. ACRI 2006, Springer Berlin Heidelberg, Perpignan, France* (2006), pp. 30–40. 1

[Fri13] FRIDENFALK M.: Application of cellular automata for generation of chess variants. In *Proc. 2013 IEEE International Games Innovation Conference (IGIC), Vancouver, Canada* (2013), pp. 57–63. 1

[Fri15] FRIDENFALK M.: *Cellular Automata in Modular Space: Rigid Systems – Volume I – Number I*. Uppsala universitet, 2015. 1

[Gar70] GARDNER M.: Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American 223*, 4 (Oct. 1970), pp. 120–123. 1

[GBM08] GOBRON S., BONAFOS H., MESTRE D.: GPU accelerated computation and visualization of hexagonal cellular automata. In *Proc. ACRI 2008, Springer Berlin Heidelberg, Yokohama, Japan* (2008), pp. 512–521. 1

[LBMSTM11] LEON P. A., BASURTO R., MARTINEZ G. J., SECK-TUOH-MORA J. C.: Complex dynamics in a hexagonal cellular automaton. In *Proc. High Performance Computing and Simulation, HPCS* (2011), pp. 750–756. 1

[Wol02] WOLFRAM S.: *A New Kind of Science*. Wolfram Media, Inc., 2002. 1

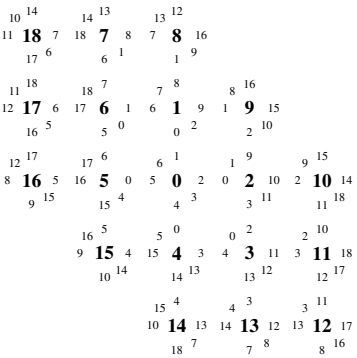


Figure 5: *The nonaligned model, in this example with $n = 5$, for matrix representation by the conversion of the rhombic structure to its square-based equivalent. Each node in this graph is connected to six neighbors.*

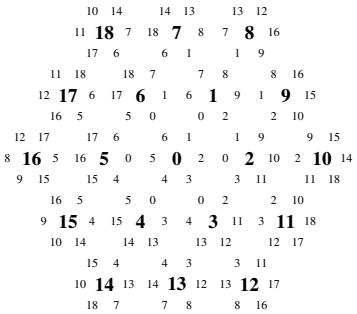


Figure 6: *Same as in previous figure, but with a geometrically correct representation of node locations. In the implemented algorithms for the generation of the connectivity maps, the point of origin is placed at the center of the hexagon, expanding clockwise.*

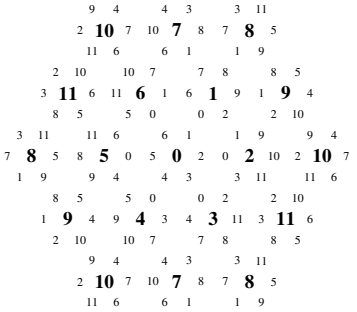


Figure 7: *The aligned model, with borderline nodes evenly shared between hexagonal modular spaces. As shown here, two borderline nodes (8 and 10), are included at three locations, with the remaining borderline nodes, at two.*

Figure 8: *Implemented kernels, $k = 0$ (left) and $k = 1$ (right).*

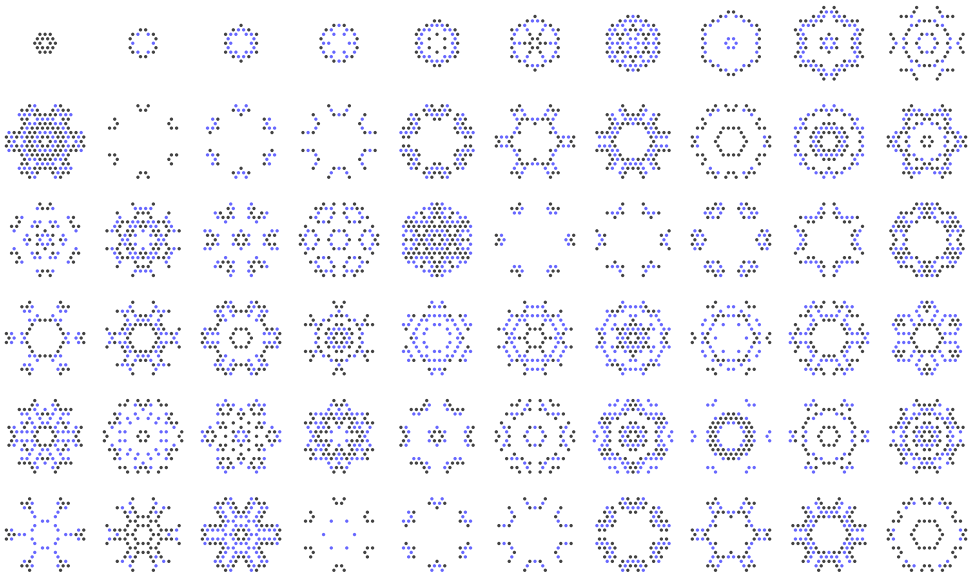


Figure 9: *Aligned model, $n = 17$, $k = 1$, generations 0 to 59.*

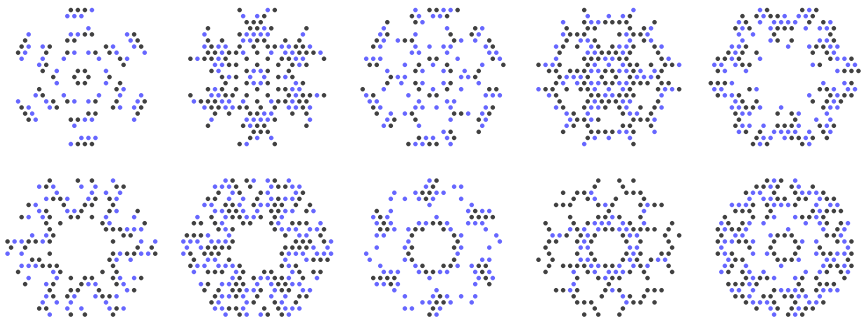


Figure 10: *Primary model, $n = 23$, $k = 1$, generations 1000 to 1009.*

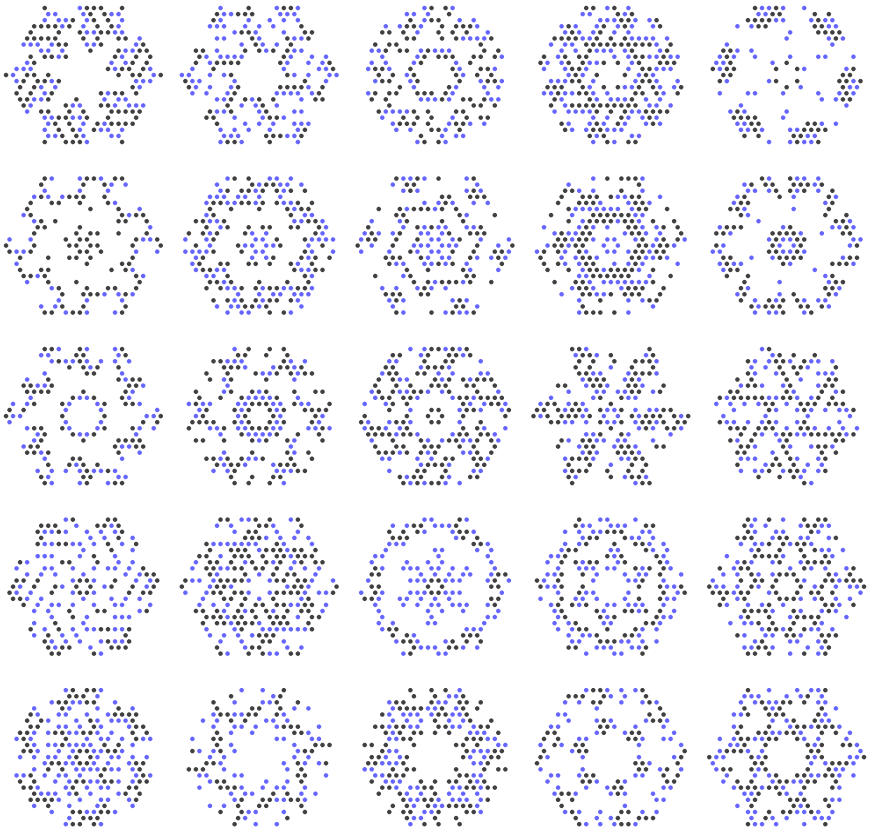


Figure 11: *Primary model, $n = 23$, $k = 0$, generations 100 to 124.*

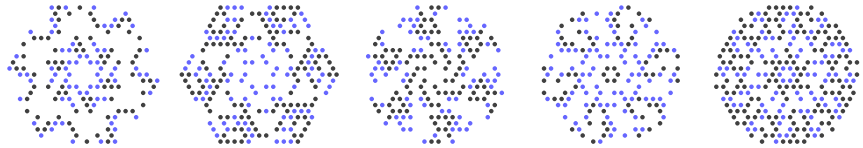


Figure 12: *Primary model, $n = 23$, $k = 0$, generations 1015 to 1019.*

SIGRAD 2016
M. Hayashi (Editors)

Low Quality Mobile Image Data Processing Under Uneven Shading

- Separating and Cleaning Text Lines and Graphic Regions in mobile Color Document Image -

Xiaohua Zhang[1], Ning Xie[2], Masayuki Nakajima[3,4], Masaki Hayashi[4] and Steven Bachelder[4]

[1]Hiroshima Institute of Technology, Hiroshima, Japan
[2]Tongji University, Shanghai, China
[3]Kanagawa Insitute of Technology, Atsuki, Japan
[4]Uppsala University, Visby, Gotland, Sweden

Abstract

This paper proposes a simple approach for extracting texts from graphic regions in low quality color document images taken by smart phones or other mobile devices with cameras. An algorithm first computes an edge map by the Canny edge detector. All textual and non-textual regions are then analyzed heuristically based on their connected components(CC). A 2D histogram is calculated to estimate the frequent width and height of connected components. After grouping the CCs according to association rules, the CCs in which the width or height levels are then measured as extremely large or small are assigned as non-textual regions. The remaining CCs are then extracted as text regions. The results of our experimentations demonstrate that the proposed approach performs with plausible consistency.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [IMAGE PROCESSING AND COMPUTER VISION]: Scene Analysis—Shading

1. Introduction

Smart phones and other mobile devices equipped with camera are becoming increasingly ubiquitous with widespread use throughout most parts of the world. An effective way of taking a memo of text segments found in magazines, advertisements or other texts is to simply take a photo of the text rather than resorting to pen and paper. To increase automatisation, the text in the document image should be formatted in order to facilitate text recognition [ZENM13]. In this paper, we propose a simple approach for extracting texts from graphic regions in low quality color images. The algorithm computes an edge map using the Canny edge detector. By analyzing connected components heuristically, the textual and non-textual regions are separated making text extraction possible.

2. Extracting text regions

In a document image such as shown in Fig. 1(a) which is taken by a mobile phone under poor lighting conditions, the image contains textual and non-textual regions and is covered with uneven shading. We assume that the given document image has little distortion and that the text lines are nearly aligned in horizontal.

The proposed algorithm is applied to the gray channel. Once the uneven shading is removed [ZXHX16] based on retinex theory [LM71], an edge map is constructed using the Canny algorithm. Note that since the background in the image is complicated, the binarized is useless for text extraction. The detected edges are very important and are used for extracting text lines from graphic regions.

To separate textual and non-textual regions, all pixels on edges are labeled. After labeling, each connected component (CC) is embraced as a quadrangle called a bounding box as shown in Fig. 1(b). A CC is featured with several attributes such as position, width, height, area, density, aspect ratio etc. If a CC has a large width or height, such as those on the left and right in Fig. 1(b), it is obviously a non-textual region and should be deleted. The CCs with small area, density or aspect ratio are considered as noise and are deemed non-textual regions. These CCs are also deleted.

A text line consists of several CCs distributed on the same line. To group these CCs, the frequent width and height of

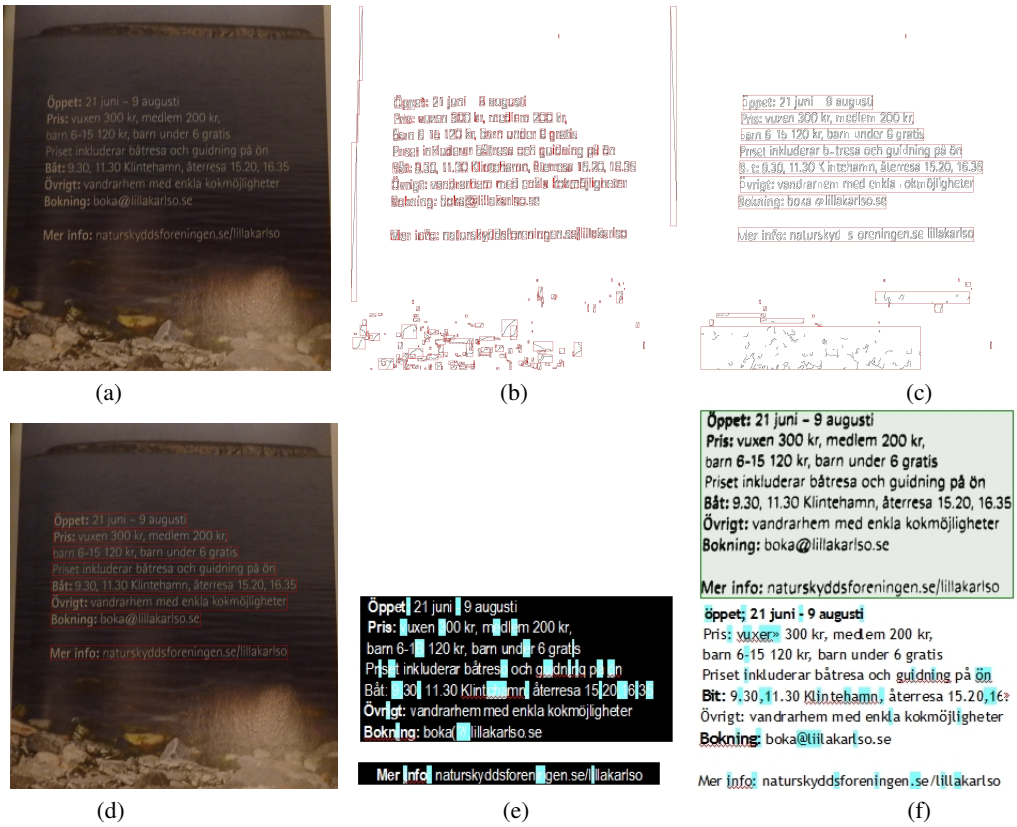


Figure 1: *Original low quality color document image (a); initial connected components after the Canny edge detection (b); connected components after grouping (c); extracted text lines (d); OCR result of text regions (e)*

CCs are required. To this end, a 2D histogram of width and height is computed. The location of the highest peak indicates the mode of width and height. Compared with the frequent height, if a CC has a larger height, it is assumed as an image region. All inner CCs are combined with its mother CC by detecting the inclusion relations. Later, all CCs are grouped according to their attributes to extract text lines as demonstrated in Fig. 1 (c). After grouping, connected components with extremely large or small width and height as those at the bottom in Fig. 1 (c) are deleted.

All remaining connected components are the extracted text lines as shown in Fig. 1 (d). It can be clearly observed that the text lines are well extracted. We binarized [SP00] the extracted text regions as shown at the top in Fig. 1 (f) and proceeded to feed it to OCR software, the recognized result is at the bottom in Fig. 1 (f). The recognition rate is very high. Note that the recognition rate, as shown in Fig. 1 (f), is very low when feeding original image directly to the OCR software.

3. Conclusions

We proposed an approach for extracting text regions from graphic regions in low quality document images taken by smart phones and other mobile devices equipped with cameras. Once the uneven shading was removed, the text lines were collected by heuristically analyzing the connected components computed from the detected edges. The text regions were segmented and defined according to the frequent size, positions, densities from all the CCs. To verify the accuracy of extractions, the text-only regions are binarized, then fed to OCR software. The experimental results demonstrate that the recognition rate is higher than images tested without text region extraction.

References

[LM71] LAND E., MCCANN J.: Lightness and retinex theory,. *Journal of the Optical Society of America 61* (1971), 1–11. 1

[SP00] SAUVOLA J., PIETIKAINEN M.: Adaptive document image binarization. *Pattern Recognition 33*, 2 (Feb. 2000), 225–236. 2

[ZENM13] ZIRARI F., ENNAJI A., NICOLAS S., MAMMASS D.: A document image segmentation system using analysis of connected components. In *Proc. of* 12th *ICDAR* (2013). 1

[ZXHX16] ZHANG X., XIE N., HUANG H., XIN Y.: Fast restoration of text strokes from a document image with uneven shading. In *Proc. of* 19th *IWAIT* (2016). 1