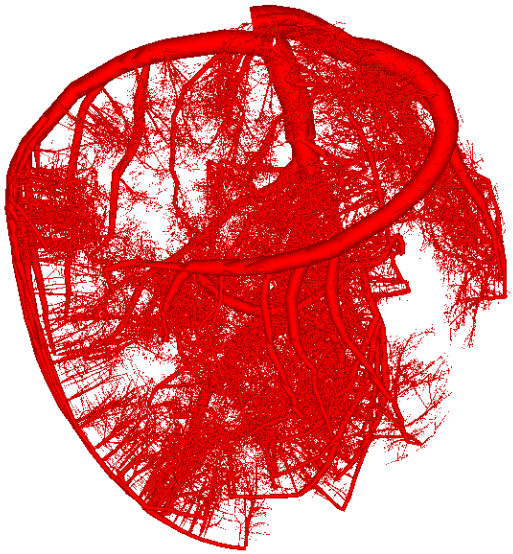
Scientific Visualization: Advanced Concepts

Edited by

Hans Hagen



DAGSTUHL
FOLLOW-UPS

Dagstuhl Follow-Ups – Vol. 1 www.dagstuhl.de/dfu

Editor

Hans Hagen
AG Computergrafik und HCI
University of Kaiserslautern
67653 Kaiserslautern, Germany
hagen@informatik.uni-kl.de

ACM Classification 1998
I.3 Computer Graphics, I.4 Image Processing and Computer Vision, J.2 Physical Sciences and Engineering, J.3 Life and Medical Sciences

ISBN 978-3-939897-19-4

Published online and open access by
Schloss Dagstuhl – Leibniz-Center for Informatics gGmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany.

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Cover graphic: Taken from chapter 14 of Thomas Wischgoll on *"Modeling and Visualization of Cardiovascular Systems"*, pp. 210–226 of this book.

Digital Object Identifier: 10.4230/DFU.SciViz.2010.i

ISBN 978-3-939897-19-4 **ISSN 1868-8977** **http://www.dagstuhl.de/dfu**

DFU – Dagstuhl Follow-Ups

The *DFU – Dagstuhl Follow-Ups* series offers a frame for the publication of peer-reviewed papers based on Dagstuhl Seminars. DFU volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8977

www.dagstuhl.de/lipics

Contents

Scientific Visualization: Advanced Concepts.
Editor: H. Hagen; pp. v–x
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Medical and Life Sciences

Applications

Preface

The Schloss Dagstuhl seminars on Scientific Visualization provide a dynamic setting for ongoing and future research in visualization. Numerous contributions in this active field originated at Schloss Dagstuhl, and were extended to large-scale collaborative research and high-impact works. This volume of the Dagstuhl Follow-Ups series contains the proceedings from two seminars in 2005 and 2007, as well as updated papers on topics related to talks from the 2007 seminar. Many of these works represent updated research and results on the topics that were initiated in the 2005 and 2007 seminars.[1]

Scientific Visualization (SV) is concerned with the use of computer-generated images to aid the understanding, analysis and manipulation of data. Since its beginning in the early 90's, the techniques of SV have aided scientists, engineers, medical practitioneers, and others in the study of a wide variety of data sets including, for example, high performance computing simulations, measured data from scanners (CAT, MR, confocal microscopy), internet traffic, and financial records. Somewhat as a result of these past successes, matters are changing for research in SV. The data sets are becoming massive in size, complex and multi-dimensional in nature and the goals and objectives of the visualization much less precisely defined, but yet the results are needed with higher urgency and importance. The multi-resolution and hierarchical methods of today do not scale to these new data sets. The segmentation and knowledge extraction methods of today need to be completely revamped in order to be useful. Because of the changes that are taking place in SV, it was and is important that a group of senior researchers meet with select junior researchers to map out the future research agenda for this critical area.

One of the important themes being nurtured under the aegis of Scientific Visualization is the utilization of the broad bandwidth of the human sensory system in steering and interpreting complex processes and simulations involving voluminous data sets across diverse scientific disciplines. Since vision dominates our sensory input, strong efforts have been made to bring the mathematical abstraction and modeling to our eyes through the mediation of computer graphics. This interplay between various application areas and their specific problem solving visualization techniques was emphasized in the seminars.

Reflecting the heterogenous structure of Scientific Visualization, the selected papers of this Dagstuhl Follow-Ups volume focus on the following topics:

- **Visual Analytics:** The fields of information analysis and visualization are rapidly merging to create a new approach to extracting meaning from massive, complex, evolving data sources and stream. Visual analytics is the science of analytical reasoning facilitated by interactive, visual interfaces. The goal of visual analytics is to obtain insight into massive, dynamic and often conflicting pieces and formats of information; to detect the expected and to discover the unexpected; and to yield timely assessments with evidence and confidence levels.
- **Quality Measures:** It is vital for the visualization field to establish quality metrics. An intrinsic quality metric will tremendously simplify the development and evaluation of

[1] See `http://www.dagstuhl.de/05231` and `http://www.dagstuhl.de/07291` for details on the corresponding Dagstuhl Seminars.

Scientific Visualization: Advanced Concepts.
Editor: H. Hagen; pp. ix–x
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

various algorithms. The establishment of quality metrics will also advance the acceptance and use of visualization in industrial and medical applications.

- **Ubiquitous Visualization:** As ubiquitous computing is getting increased attention, also visual display of everywhere available data is necessary. Challenges include: heterogeneous output devices, novel interaction metaphors, network bandwidth (availability, reliability), graceful degradation of algorithms with respect to largely varying resources, in vivo visualization (real time, no pre-processing, robust).
- **Multifield and Multiscale Visualization:** The output of the majority of computational science and engineering simulations is typically a combination of fields, so called multifield data, involving a number of scalar fields, vector fields, or tensor fields. Similarly, data collected experimentally is often multifield in nature (and from multiple sources). The ability to effectively visualize multiple fields simultaneously, for both computational and experimental data, can greatly enhance scientific analysis and understanding. Multiscale problems with scale differences of several orders of magnitude in CFD, nanotechnology, biomedical engineering and proteomics pose challenging problems for data analysis. The state of the art in multiscale visualization considerably lags behind that of multiscale simulation. Novel solutions to multiscale and multifield visualization problems have the potential for a large impact on scientific endeavors.
- **Categorical Visualization:** Information and knowledge is extremely difficult to extract from multi-valued, multi-dimensional, multi-modal and multi-layered categorical data. These data sets abound today and the pay-offs for understanding them are substantial. Mathematical techniques based upon functional relationships break down requiring completely new paradigms to visualize these types of data sets.
- **Intelligent/Automatic Visualization:** Ever-increasing data sizes require semi-automatic methods that concentrate on the typically very small portion of the relevant information in the data. Techniques include model- and knowledge-based segmentation, classification in abstract feature spaces, computation of saliency information from derived data characteristics, automatic detection of important isosurfaces, automatic creation of expressive transfer functions, automatic landmark selection and automatic path and navigation guidance.
- **Point-based/Mesh-free Visualization:** A typical strategy to visualize unorganized multidimensional data sets is to transform the data into standard geometric primitives of triangles and triangular mesh surfaces prior to rendering. This intermediate step is time consuming, but necessary to map the data set to standard (hardware and software) graphics primitives. With the recent advances in point-based rendering, new efficient and creative approach for visualizing scattered and unorganized data sets are potentially possible.

Hans Hagen

Generalized Hyper-cylinders: a Mechanism for Modeling and Visualizing N-D Objects

Matthew O. Ward[1] and Zhenyu Guo[1]

1 **Computer Science Department, Worcester Polytechnic Institute**
100 Institute Rd., Worcester, MA 01609 USA
{matt,zyguo}@cs.wpi.edu

Abstract

The display of surfaces and solids has usually been restricted to the domain of scientific visualization; however, little work has been done on the visualization of surfaces and solids of dimensionality higher than three or four. Indeed, most high-dimensional visualization focuses on the display of data points. However, the ability to effectively model and visualize higher dimensional objects such as clusters and patterns would be quite useful in studying their shapes, relationships, and changes over time. In this paper we describe a method for the description, extraction, and visualization of N-dimensional surfaces and solids. The approach is to extend generalized cylinders, an object representation used in geometric modeling and computer vision, to arbitrary dimensionality, resulting in what we term Generalized Hyper-cylinders (GHCs). A basic GHC consists of two N-dimensional hyper-spheres connected by a hyper-cylinder whose shape at any point along the cylinder is determined by interpolating between the endpoint shapes. More complex GHCs involve alternate cross-section shapes and curved spines connecting the ends. Several algorithms for constructing or extracting GHCs from multivariate data sets are proposed. Once extracted, the GHCs can be visualized using a variety of projection techniques and methods to convey cross-section shapes.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases N-Dimensional Visualization, Cluster Visualization

Digital Object Identifier 10.4230/DFU.SciViz.2010.1

1 Introduction

Visualization has been identified as a critical component to the process of interactive exploration and mining of large data repositories. The essential problems that need to be addressed when developing tools for interactive visual data analysis include:

- How to structure and process the data into a format and size that is manageable within the visualization environment, yet retains most, if not all the significant information content of the original data;
- How to best display information on the screen so as to provide users with useful insights into their data given the constraints of visual perception, screen resolution, and processing speed; and
- How to provide users the ability to effectively interact with the visualization to extract meaning from the data.

The field of data visualization can be roughly divided into two distinct subfields. Scientific visualization concentrates predominantly on the display of one, two, or three-dimensional spatial/physical data, while information visualization generally assumes data sets of arbitrary

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 1–10

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

dimensionality, usually without a spatial attribute but often with one or more relations defined between data items.

The display of surfaces and solids has usually been restricted to the domain of scientific visualization; however, little work has been done on the visualization of surfaces and solids of dimensionality higher than three. The primary focus in most high-dimensional visualization has been on the display of data points, rather than surfaces and solids. However, the ability to effectively visualize higher dimensional objects, whether defined analytically or derived from data samples, would be quite useful in studying the shapes and relationships of these so-called hyper-objects. For example, the richness of the description of a cluster of N-dimensional data points could be greatly enhanced beyond commonly used methods, which often just consist of the cluster center along with the hyper-box or hyper-ellipsoid encapsulating the data. Likewise, descriptions of differences or changes in these clusters over time would benefit from richer representations.

The focus of this paper is to describe a novel method for the description, extraction, visualization, and interactive exploration of N-dimensional surfaces and solids. The general concept is to extend generalized cylinders [8, 9], an object representation regularly used in geometric modeling and computer vision, to arbitrary dimensionality, resulting in what we term Generalized Hyper-cylinders (GHCs). In its simplest form, a GHC consists of two N-dimensional hyper-spheres connected by a hyper-cylinder (spine) whose shape at any point along the cylinder is determined by interpolating between the shapes of the endpoints. A broader class of GHCs can be defined by using alternate cross-section shapes as well as curved spines. We describe several algorithms for extracting GHCs from large multivariate data sets, with user-controllable parameters to adjust the accuracy at which the GHCs approximate the real data. Once extracted, the GHCs can be visualized in 2-D or 3-D using a variety of techniques, such as projecting the endpoints into the display space using PCA or MDS. A variety of object types to represent the shape of the GHC are being explored and evaluated. Finally, a suite of tools is being implemented for interactively exploring data displayed with GHCs, including operations for navigation, selection, filtering, and distortion.

This paper is organized as follows. Section 2 describes the work of others in visualizing N-dimensional points and objects. Section 3 defines generalized cylinders and their extension to generalized hyper-cylinders. Section 4 describes methods for visualizing GHCs, while Section 5 focuses on methods to extract GHCs from datasets via manual, semi-automated, and fully automated techniques. We conclude in Section 6 with a summary and a list of potential future research directions.

2 Related Work

Multivariate data can be found in most, if not all, disciplines of study, and a wide range of techniques have been developed for the visualization of such data. Popular techniques include scatterplot matrices, projected point methods [1], parallel coordinates [2, 13], and tabular views [7]. Other techniques that have been proposed include glyphs [12], pixel-oriented techniques [3], and dimensional stacking [5]. While these are useful for examining individual data records, they are less effective at providing a high-level description of the entire dataset or selected subsets of the data.

For example, if one were interested in describing the shape of a cluster in a 5-dimensional dataset, what techniques would be applicable? Most cluster descriptions in common use consist of a small number of attributes, such as the location of the cluster center, its population, and perhaps its dominant axis. Others represent a cluster by a representative

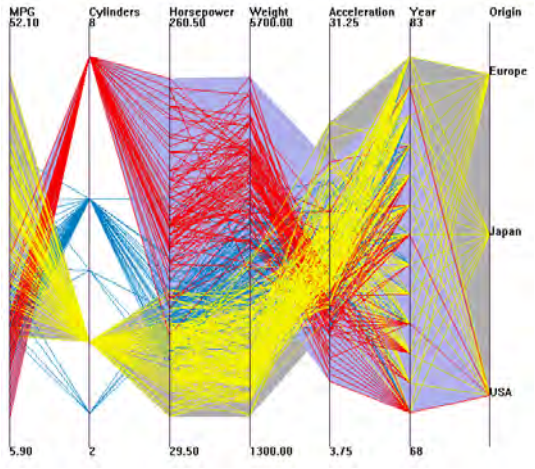


Figure 1 Two hyper-boxes in Parallel Coordinates. Two clusters have been isolated based on the second dimension. The brown and grey regions indicate the surrounding hyper-boxes for the clusters. Image generated with XmdvTool [6, 11].

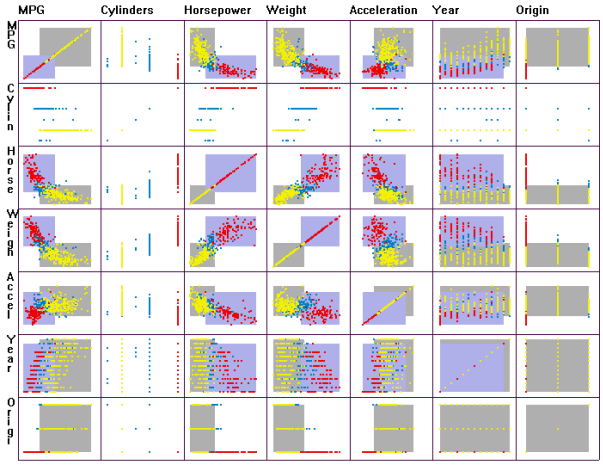


Figure 2 The same clusters in Scatterplot Matrices. Image generated with XmdvTool.

sampling of the data points contained in it. However, this is not a good representation for tasks such as comparing cluster shapes.

A simple approach to representing and visualizing N-dimensional clusters is to use the axis-aligned hyper-box that contains the points of the cluster. For example, in Figures 1 and 2, two clusters of points have been selected in the parallel coordinates and scatterplot matrix displays, respectively [6, 11]. The shaded regions indicate the extents in each dimension that contain the selected points (yellow points in the brown region, red points in the blue region). As can be seen, these regions overlap, so the user can only see the full extents of one of the clusters. Clearly an axis-aligned hyper-box is not a very accurate description of a cluster's shape.

Another approach is to decompose the N-dimensional space into a (potentially large) number of N-dimensional blocks or subspaces and represent the cluster as the set of subspaces that contain at least one data point. This is akin to the spatial enumeration technique for 3-D solid modeling [8], where a 3-D volume is represented by an array of volume elements

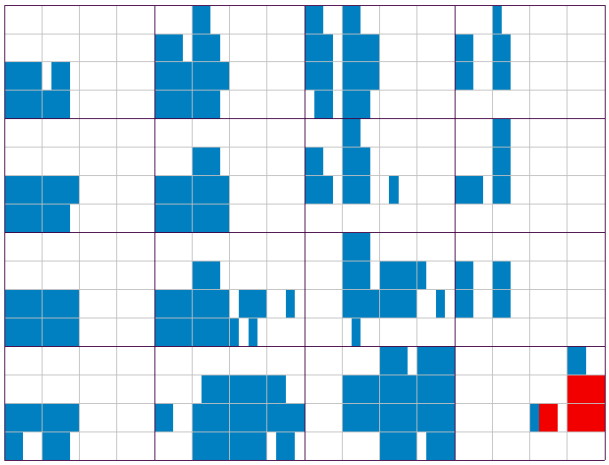


Figure 3 Spatial enumeration with dimensional stacking. Each bin represents a hyper-box in N dimensions. Those in red represent an isolated cluster [5, 4].

(voxels). One way to visualize this hyper-volume is with dimensional stacking [5, 4], where each dimension is divided into a small number of bins and the display space is recursively partitioned using pairs of dimensions. Figure 3 shows an example of a 5-dimensional dataset with one cluster (red) isolated. While separated in display space, the red bins are contiguous in N dimensions. The dark and light grid lines show the first two levels of nesting. Each dimension has four bins. This representation is not very compact, and in order to increase the accuracy of the representation the number of bins per dimension must be high. Also, for high dimensional data the number of occupied bins tends to be very small (the curse of dimensionality).

A method that is more accurate than hyper-boxes and more efficient than dimensional stacking is the H-BLOB method as described by Sprenger et al. [10]. They represent clusters via hierarchically nested hyper-spheres, which are then projected to three dimensions and visualized using implicit surfaces (see Figure 4). This generates a closed, smooth surface around each cluster, and thus provides a rich description of the shape. H-BLOBs have some similarities to GHCs; however, it would take a potentially large number of hyper-spheres to represent a shape that can be captured with a single hyper-cylinder, and we feel there are many shape features that can be derived from GHCs that would be difficult to extract with the H-BLOB representation.

3 Generalized Cylinders and Hyper-cylinders

Hyper-boxes and hyper-spheres are relatively coarse primitives to use in modeling shapes, especially those defined by groups of scattered points. In each case, there can be a significant amount of space within the model where there are no data points. In 3-D, a tapered cylinder can often come closer to encapsulating the points in a cloud, as the endpoints and radius can be adjusted to better fit the data. In geometric modeling and computer vision, this approach is known as *generalized cylinders* (GC) [9], which can be used to model axis-symmetric objects or object parts. A GC consists of two endpoints, a spine (straight or curved), and a cross-section (often a circle or ellipse). Many variants on GCs have been proposed over the years, including the use of non-convex cross-sections and varying the cross-section shape or size as one moves from one endpoint to the other. A wide range of complex object shapes

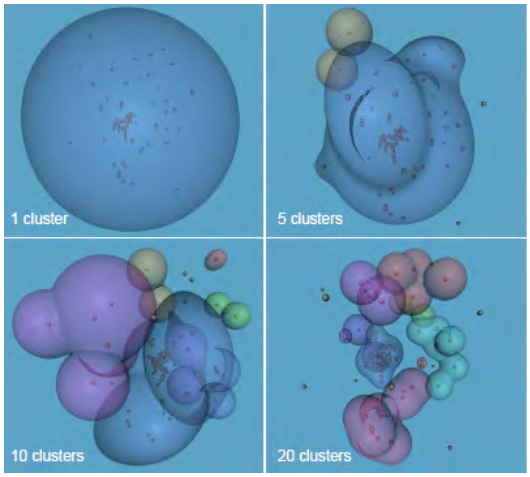


Figure 4 Implicit surfaces generated as hyper-spheres in N dimensions and projected to three dimensions. Image from [10] (used with permission).

can be represented using a small set of GCs. While GCs have been widely used in 2-D and 3-D, to the best of our knowledge, they have not been extended to higher dimensions.

In fact, it is not hard to imagine this extension, which we call a *generalized hyper-cylinder* (GHC). It is clear one can define two N-dimensional endpoints, along with a straight or curved spine connecting them. The shape of the cross-section, however, is not so straightforward. In the simplest form, we can use an (N-1)-dimensional hyper-sphere orthogonal to the spine, with either a constant or variable radius as one moves along the spine. This would result in hyper-planes at the ends of the GHC. Another alternative is to use an N-dimensional hyper-sphere at each end, similar to H-BLOBs, with an interpolated radius along the spine. As with GCs, we can also use ellipses for the cross-section shape. This can allow the GHC to fit a given dataset with increased accuracy. Finally, while it might be possible to use an arbitrary (N-1)-dimensional shape as a cross-section (e.g., represented as a GHC with one less dimension), we feel the resulting complexity would make interpretation, rendering, and analysis difficult.

4 Visualizing GHCs

There are many ways one could consider to render a set of GHCs; indeed. most multivariate data visualization techniques could be extended to convey the endpoints, spines, and cross-sections. As our initial attempt, we focused on GHCs with a straight spine and a hyper-sphere cross-section. For M GHCs we draw M 2-D scatterplots, each aligned with a particular GHC. Each GHC is represented as a trapezoid, where the width of the top and bottom are proportional to the radii at the two endpoints and the length of the trapezoid is proportional to the N-D distance between the endpoints. The endpoints are connected to represent the spine. All other GHCs in a given view are drawn relative to the focus GHC.

In the following, we describe how to generate a 2-D scatterplot to represent and visualize a hyper-cylinder. For each data point, we project it onto a scatterplot view by calculating the x and y coordinates relative to the two endpoints and the spine. Assume the two endpoints are A_1 and A_2, respectively, in N-dimensional space and A_m is the middle point of A_1A_2. For any data point B_i, it can be projected onto the spine A_1A_2. Assume the projection point is B_p, i.e., B_p is on A_1A_2 and B_iB_p is perpendicular to A_1A_2. The point B_p is calculated as

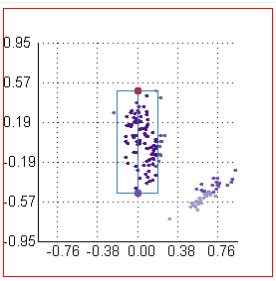


Figure 5 The perspective scatterplot view before rotating.

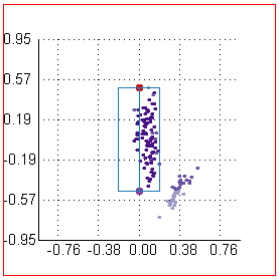


Figure 6 The perspective scatterplot view after rotating $1/3\pi$.

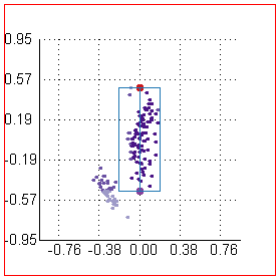


Figure 7 The perspective scatterplot view after rotating $2/3\pi$.

follows:

$$B_p = A_1 + \frac{(B_i - A_1) \cdot (A_2 - A_1)}{||A_2 - A_1||} \frac{A_2 - A_1}{||A_2 - A_1||}$$

The value of the y coordinate is the distance between A_m and B_p. If B_p is nearer A_2, the value is positive; if B_p is nearer A_1, the value is negative. The value of the x coordinate is computed as $||B_i - B_p||cos\theta$, where $||B_i - B_p||$ is the Euclidean distance from the data point and its projection point, and θ is the angle between vector $B_i - B_p$ and any fixed vector that is orthogonal to A_1A_2, say $T - T_p$ (T is any point in N-dimensional space and T_p is the projection point as described before):

$$\theta = arccos(\frac{(O - O_p) \cdot (T - T_p)}{||O - O_p||||T - T_p||})$$

Thus the x coordinate is the rotated distance from a data point to the spine which simulates a perspective view effect. When interactively increasing or decreasing all the angles by an offset, analysts are able to simulate viewing the hyper-cylinder from different orientations, i.e. by rotating the hyper-cylinder around the spine. Figures 5 to 7 show an example of the different perspective views from different orientations when viewing a hyper-cylinder in a three dimensional space. Point T is selected as the first data point of the dataset.

To visualize multiple GHCs in a single scatterplot view, we map the two endpoints of each non-focus GHC and connect them to represent the spine. We draw two perpendicular lines whose lengths are proportional to the two radii and connect the four corners (the end of the two perpendicular lines) to get a trapezoid. We fill the trapezoids with different colors to denote different GHCs.

Figure 8 shows a set of three GHCs for a 4-dimensional dataset. Each view is centered on one GHC (outlined), while the others are shown as filled colored trapezoids. The view with the red boundary indicates the GHC that is currently being edited (the focus GHC). The first GHC contains all of the data points; the amount of empty space indicates that this GHC does not fit the dataset accurately. The second and the third GHCs contain the two clusters, which are more accurate than the first GHC as they have smaller radii and shorter spine lengths. It is very easy to discover how well the data points fit the cluster and examine outliers that do not fit well in either GHC.

This representation is simple, yet flexible. Curved spines can easily be accommodated, as can cross-sections that change in a non-linear way. There are also many possible variations on this simple view, including:

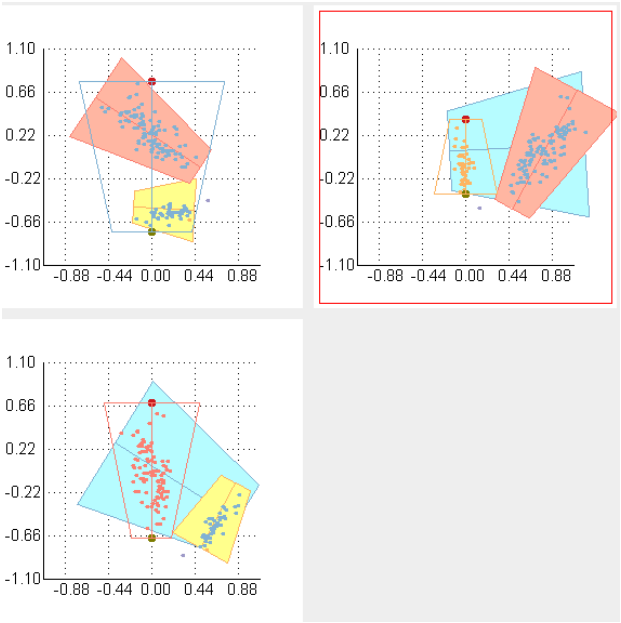


Figure 8 Visual representation of a set of GHCs .

- Placing endpoints, as well as intermediate axis points for curved spines, using dimensionality reduction techniques such as PCA and MDS.
- Rendering in 3-D, which can reduce the amount of occlusion.
- Using colored stripes to represent how each dimension is changing along the length of the spine.
- Extruding a 2-D star glyph along a 3-D spine, where the length of each branch of the star conveys the dimension size (for hyper-ellipsoidal cross-section).

We are experimenting with these and other variations for visualizing GHCs.

5 Extracting GHCs from Data

One of the biggest challenges with GHCs is deriving them from data, as a given dataset could be represented with varying degrees of accuracy, leading to varying numbers and shapes of GHCs. We can categorize potential approaches as either manual, automated, or semi-automated. In the manual case, the user defines the endpoints for each GHC as well as the cross-section size and shape. Data that fall within these specifications are assigned to the GHC being constructed. By coloring the points as they get covered by the GHC, the user can interactively adjust the position, shape, and orientation of the GHC to best fit the data. In this case, it is up to the user to decide when multiple GHCs are needed. Figure 9 shows such an interface for manual GHC specification. A subset of 5 dimensions are selected in the parallel coordinates view, and the endpoints of the GHC are specified in the scatterplot matrix view. In the projected view the user can adjust the radii of the GHC at each end.

In a semi-automatic approach, the user might specify pairs of endpoints and allow the system to compute the best cross-section size and shape to use. It would also be possible to automate the fitting of a curved spine, based on the distribution of points. Again, it would be up to the user to indicate how many GHCs should be used and approximately where they

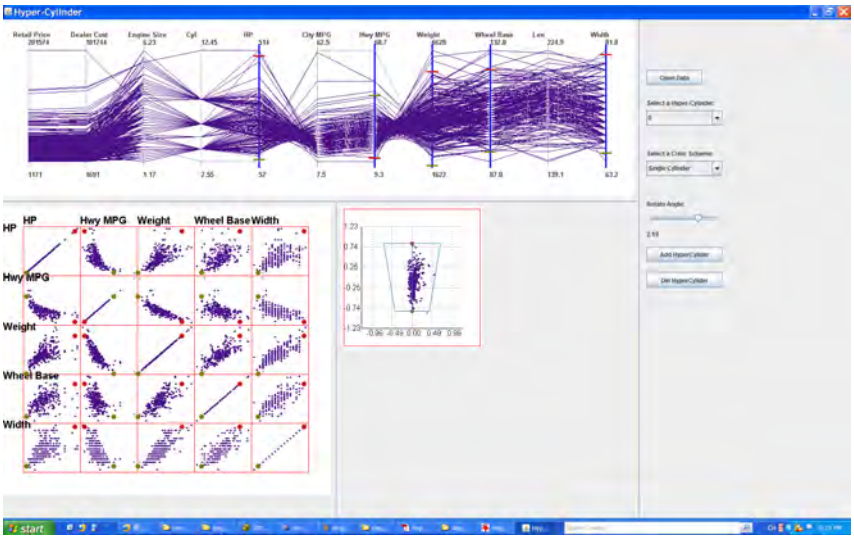


Figure 9 Interactive creation of GHCs to represent a dataset.

are located. Automated techniques could also be used to refine the endpoint location using a localized search.

A fully automatic method could start with a clustering of the data. It would then be assumed that each cluster could be represented by a single curved GHC or a set of connected linear GHCs. Starting with the extreme points of a cluster, the spine could be initialized to the straight line connecting the endpoints. Each point in the cluster would then be projected to this line to ascertain if any gaps exist that should result in the division of the cluster into multiple GHCs. Assuming no gaps exist, the distances and directions to each cluster point from the spine could be used to bend the spine towards the center of the data that project to that neighborhood of the spine. One challenge would be to identify where forks and joins must occur, e.g., when few points are close to the spine and there are two or more groups of points that share an approximate direction from the spine. One problem with this approach is that the initial choice of endpoints is critical; it is important to not choose outliers, and rather choose points that represent the dominant axis of the cluster.

We are studying ways to enhance our manual GHC creation tool as well as exploring algorithmic alternatives to some or all of the stages of extraction and refinement.

6 Summary and Conclusions

In this paper we have introduced a new method for approximately describing objects of dimensionality greater than three. By extending the notion of Generalized Cylinders from 3-D to N-D we can describe clusters and other patterns in multivariate datasets in a compact, yet descriptive form. GHCs can be useful for not only compressing a dataset, but also for comparing multiple datasets for change analysis and in specifying queries on data.

There are many unsolved problems and avenues for research in the definition, extraction, and use of GHCs. While space limitations prohibit us from going into detail here, a partial list of such topics includes:

- What forms of interaction should be available to create and explore GHCs? These might include navigation in data space, feature space, or display space, drill-down and roll-up to

get more or less detail on demand, and distortions such as bending, moving, and scaling to see objects more clearly without losing context.
- Are there other shapes that would capture the shape more accurately or that would be easier to extract with comparable shape accuracy?
- What error measures could be used?
- What object configurations would not be suitable for GHCs?

It is anticipated that a wide range of disciplines will benefit from this research, including bioinformatics, computational modeling, telecommunications, health care, and other scientific and industrial domains where the analysis of high dimensional data and models is important.

Acknowledgements

This work was primarily supported by the National Science Foundation under Award Number IIS-0812027.

References

1 Patrick Hoffman, Georges Grinstein, and David Pinkney. Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations. In *NPIVM'99: Proceedings of the 1999 Workshop on New paradigms in Information Visualization and Manipulation (in conjunction with the 8th ACM International Conference on Information and Knowledge Management)*, pages 9–16, New York, NY, USA, 1999. ACM.

2 Alfred Inselberg and Bernard Dimsdale. Parallel coordinates: a tool for visualizing multidimensional geometry. In *VIS'90: Proceedings of the 1st Conference on Visualization 1990*, pages 361–378, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

3 Daniel A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000.

4 John T. Langton, Astrid A. Prinz, and Timothy J. Hickey. Neurovis: combining dimensional stacking and pixelization to visually explore, analyze, and mine multidimensional multivariate data. In Robert F. Erbacher, Jonathan C. Roberts, Matti T. Gröhn, and Katy Börner, editors, *Proc. Visualization and Data Analysis*, page 64950H. SPIE, 2007.

5 Jeffrey LeBlanc, Matthew O. Ward, and Norman Wittels. Exploring n-dimensional databases. In *VIS'90: Proceedings of the 1st Conference on Visualization 1990*, pages 230–237, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

6 Allen R. Martin and Matthew O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *IEEE Visualization Conference*, pages 271–278, Los Alamitos, CA, USA, 1995. IEEE Computer Society.

7 Ramana Rao and Stuart K. Card. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *CHI'94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 318–322, New York, NY, USA, 1994. ACM.

8 Aristides G. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, 12(4):437–464, 1980.

9 Steven Shafer and Takeo Kanade. The theory of straight homogeneous generalized cylinders and a taxonomy of generalized cylinders. In *Proceedings of the 1983 DARPA Image Understanding Workshop*, pages 210–218, 1983.

10 T. C. Sprenger, R. Brunella, and M. H. Gross. H-blob: a hierarchical visual clustering method using implicit surfaces. In *VIS'00: Proceedings of the Conference on Visualization 2000*, pages 61–68, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

11 Matthew O. Ward. Creating and manipulating n-dimensional brushes. In *Proceedings of Joint Statistical Meeting*, pages 6–14, 1997.

12 Matthew O. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3/4):194–210, 2002.

13 Edward J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85:664–675, 1990.

Computing an Optimal Layout for Cone Trees

Dirk Zeckzer[1], Fang Chen[1], and Hans Hagen[1]

1 **Department of Computer Science**
University of Kaiserslautern
Kaiserslautern, Germany
{zeckzer,chen,hagen}@informatik.uni-kl.de

Abstract

Many visual representations for trees have been developed in information and software visualization. One of them are cone trees, a well-known three-dimensional representation for trees. This paper is based on an approach for constructing cone trees bottom-up. For this approach, an optimal layout for these trees is given together with a proof that based on the assumptions, there can be no better layouts. This comprises special cases, an optimal constant for the general case, and a post-processing step improving the layout.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Cone Trees, Information Visualization, Tree Layout

Digital Object Identifier 10.4230/DFU.SciViz.2010.11

1 Introduction

Cone trees are a well-known three-dimensional representation for trees. Based on the original work by Robertson et al. in 1991 [4], several different methods for the computation of the layout have been proposed. While the original approach implemented a top-down approach, in [1] a bottom-up approach was chosen to layout the cone tree. Unfortunately, some details of the algorithm were not published. A compensation factor for the computation was motivated and introduced, but its value or computation has not been given explicitely.

In this paper, we build upon this work. We show how special cases can be computed and prove a tight bound for the compensation factor of the general case.

The paper is structured as follows. In Section 2, we review the ideas of [4], [1], and other related work. In Section 3, the general setting and the assumptions for the layout are described. In Section 4, the special cases are introduced, while in Section 5 the general case is described. There, a tight bound for the compensation factor needed is proven. Further, an optimization step is introduced in this Section, too. Finally, in Section 6, open problems are given before we conclude this paper.

2 Related Work

This work is based on [4], [1], and [3]. Cone trees were first introduced in [4] (see Section 2.1). Based on this work, a different approach for constructing cone trees was proposed in [1] (see Section 2.2). In [3], reconfigurable disk trees were proposed as an enhancement to cone trees (see Section 2.3).

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 11–29

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

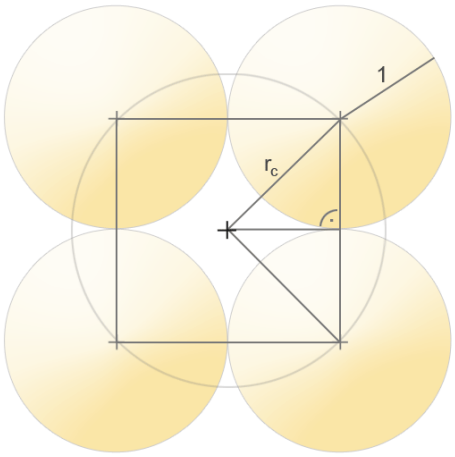


Figure 1 Counterexample.

2.1 Cone Trees

Cone trees were first presented by Robertson et al. in 1991 [4]. They are very well suited to layout trees with a large number of children. Based on the available screen size, the layout is produced in a top-down manner. Details are given in Section 3.

An evaluation of cone trees was performed by Cockburn and McKenzie in 2000 [2]. They found that for certain tasks, the participants were significantly slower than when using a 2D tree interface. Possible reasons for the bad performance of cone trees mentioned in their study comprise less familiarity with cone trees and a comparatively crude 3D interface.

2.2 Beyond Cone Trees

In [1], a different approach was chosen to layout the cone tree. The computation of the respective cones is done bottom-up. Our approach is based on this work. The details are given in Sections 3 and 5.1.

2.3 Reconfigurable Disk Trees

In [3], so-called RDTs (Reconfigurable Disk Trees) are presented. They define a reference point and an apex point together with apex height, reference height, and reference length. These additional parameters allow different cone shapes and layouts. Additionally, they provide an evaluation of the node density.

In the implementation, explicit values are given for the compensation factor introduced in [1]. Unfortunately, there are special cases where this computation will result in overlaps. One of these cases is depicted in Figure 1.

Consider four children with the radius 1. Moving them together as closely as possible leads to a square with edge length 2. Considering the triangle depicted in Figure 1, the radius is the largest edge of a triangle with a right angle and the smaller edges being 1 unit long. Thus, r_c can be computed as

$$r_c = \sqrt{1^2 + 1^2} = \sqrt{2} > 1.414$$

This is the minimal radius.

According to [3], the radius will be computed as follows:

$$\begin{aligned} inner\_sum &= 2 \cdot \sum_k out\_rad_k \\ &= 2 \cdot (1+1+1+1) \\ &= 8 \\ max\_child\_radius &= \max_k \{out\_rad_k\} \\ &= \max(1,1,1,1) \\ &= 1. \end{aligned}$$

It follows that

$$inner\_sum = 8 > 2 \cdot \pi \cdot 1 = 2 \cdot \pi \cdot max\_child\_radius.$$

Thus, the radius will be approximated as

$$rad = \frac{inner\_sum}{2 \cdot \pi} = \frac{8}{2 \cdot \pi} < 1.28.$$

Thus, $rad < r_c$, which leads to overlapping circles for this case.

3 General Setting

Cone trees are essentially built as follows. The root of the tree is placed on top. All children of a node are placed at a fixed distance below this node forming a circle. If the node is projected onto the plane of this circle, it will be projected onto the center of the circle (see also Figure 2). In Figure 3, one cone of a tree is shown. The parent node (yellow) is connected to its children using a transparent cone.

The approach chosen in [4] takes a fixed 3D space and fits the cone tree into this space. The height is divided by the depth of the tree giving the height of each cone. The radius of each circle is progressively reduced from top to bottom. The disadvantage of this approach is that the size of the children will have to be adapted to the size of the circle. For large trees, the circles containing the leaf nodes will be either very small or will not be visible at all.

Another approach was taken in [1]. There, a bottom-up approach was chosen. Each leaf element of the tree has a bounding circle. This circle gives the size of the leaf element. All leaf elements are arranged in a circle such that they do not overlap. The bounding circle of all elements arranged in the bottom circle of the cone gives the size of the cone. The whole cone, leaves plus parent node, is placed in the bottom circle of the next cone one level higher.

Here, we describe the adaptation of the ideas presented in [1]. We describe the construction of one cone. In Figure 2, a 2D view of the circle at the base of the cone is given. The radii of the bounding circles of the children determine the radius of the circle of this cone. In Figure 3, a 3D view is given. The parent node is placed at the top of the cone. All cone elements and variables are described next.

Given are a parent node N_p of the cone and its n child nodes $N_1 \dots N_n$. The size of each node is given by its radius $r_p, r_1, \dots, r_n$. The cone is described by the radius r_c of the circle at its base and its height h_c. Further, the radius r_b of the bounding circle of the children is needed. In the Table 1, these entities are summarized.

Currently, the height of the cone is always set to a predefined constant value $h_c = const$.

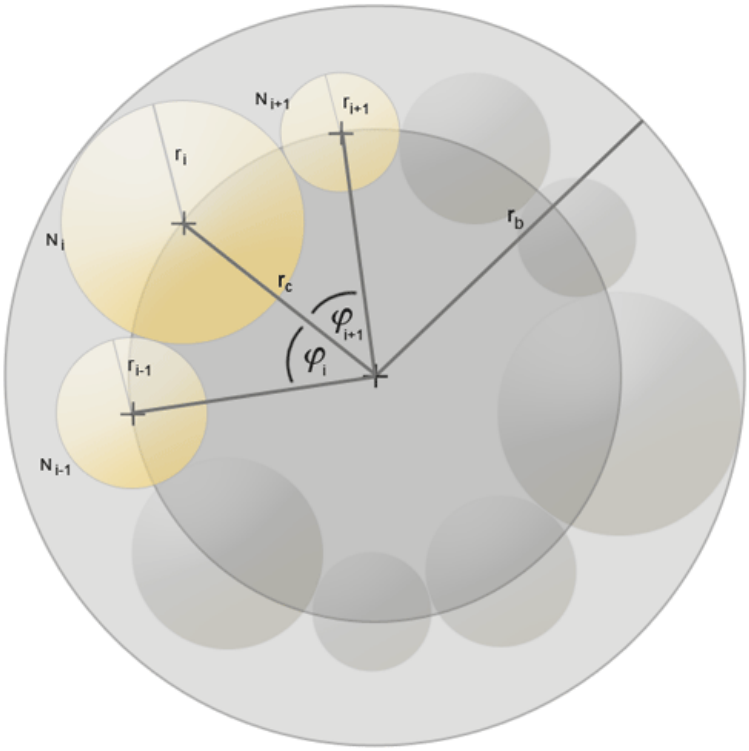


Figure 2 Construction of one cone element: 2D view.

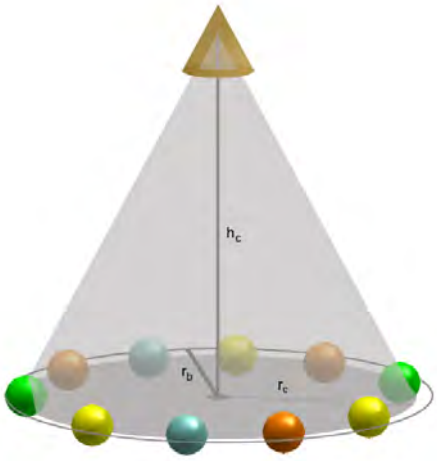


Figure 3 Construction of one cone element: 3D view.

Table 1 Abbreviations.

Abbreviation	Element
N	Node
S	Shape representing a node
R	Radius
H	Height
φ	Angle between two child nodes
C	Circumference
p subscript	Parent node
$1 \dots n$ subscript	Child node number $1 \dots n$
c subscript	Cone
b subscript	Boundary
G subscript	Glyph representing a node

4 Special Cases

4.1 No Children

If there are no children, then the radius and the bounding radius of the cone are set to the bounding radius of the glyph r_G representing the parent node N_p of the cone:

$$r_c = r_b = r_G.$$

4.2 One Child Node

If the number of child nodes is $n = 1$, then the child node is positioned directly below the parent node. Placing a child at a certain position either means placing a leaf at this position or placing the parent node of the cone representing the subtree at this position. See Figure 4 for a depiction of this situation.

The bounding radius of the cone is set to the maximum of the radius of the cone and the bounding radius r_G of the glyph representing the parent node N_p of the cone:

$$r_b = \max(r_G, r_c).$$

4.3 Two Child Nodes

If the number of children is $n = 2$, then an optimal radius and an optimal bounding radius can be computed (see Figure 5).

The cone can be computed as

$$\begin{aligned} r_c &= \frac{r_1 + r_2}{2} \\ r_b &= r_c + \max(r_1, r_2) \\ \varphi_1 &= 0^\circ \\ \varphi_2 &= 180^\circ. \end{aligned}$$

The position of the first child is at an angle of $\varphi_1 = 0^\circ$ and at a distance of r_c from the center of the circle. The position of the second child is at an angle of $\varphi_2 = 180^\circ$ and at a distance of r_c from the center of the circle.

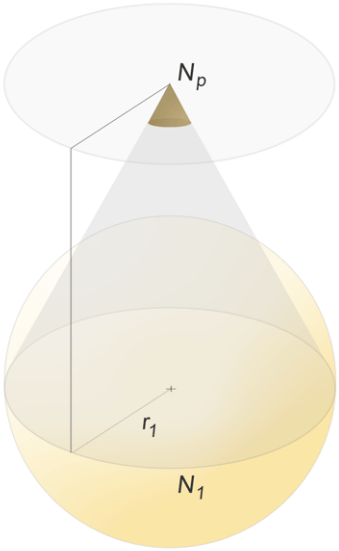


Figure 4 One child node.

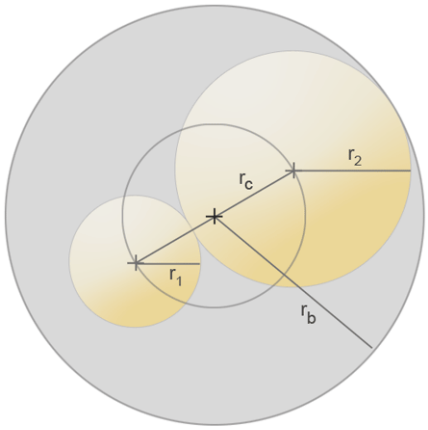


Figure 5 Two child nodes

Remark: If the radii are different, then the bounding circle is not the smallest bounding circle containing both children. Its radius is

$$\frac{|r_1 - r_2|}{2}$$

larger. One possible optimization would be to use the radius

$$r'_b = r_1 + r_2.$$

Then, the parent node has to be put at the position between the two child nodes that is the center of the bounding circle. But then the cone would no longer be a right circular cone but an oblique cone.

4.4 Three Child Nodes

If the number of children $n = 3$, then an optimal radius and an optimal bounding radius can be computed, too. Let r_1, r_2, r_3 be the radii of the children, such that without loss of generality $r_1 \geq r_2 \geq r_3$. Let A, B, C be the corners of a triangle with the edges of that triangle being

$$\begin{aligned} a &= r_1 + r_2 \\ b &= r_1 + r_3 \\ c &= r_2 + r_3. \end{aligned}$$

That is, A is the center of the child having radius r_3, B is the center of the child having radius r_2, and C is the center of the child having radius r_1.

A triangle is acute if it satisfies the following set of inequalities:

$$\begin{aligned} a^2 + b^2 &> c^2 \\ b^2 + c^2 &> a^2 \\ c^2 + a^2 &> b^2. \end{aligned}$$

If the triangle is acute, then the circumcenter of the triangle lies inside the triangle. Otherwise, it lies outside. If the circumcenter lies inside the triangle, then the children can be positioned on the circumcircle (see Figure 6). In this case, the order of the radii is not important. Let A denote the first child N_1, B denote the second child N_2, and C denote the third child N_3. The following equations can be used to compute the radius of the circumcircle and the angles of the child positions:

$$\begin{aligned} \beta &= \arccos \frac{a^2 + c^2 - b^2}{2 \cdot a \cdot c} \\ r_c &= \frac{b}{2 \cdot \sin \beta} \\ \varphi_1 &= 0^\circ \\ \varphi_2 &= \arccos \frac{r_c^2 + r_c^2 - b^2}{2 \cdot r_c \cdot r_c} \\ &= \arccos \frac{2 \cdot r_c^2 - b^2}{2 \cdot r_c^2} \\ \varphi_3 &= \arccos \frac{r_c^2 + r_c^2 - c^2}{2 \cdot r_c \cdot r_c} \\ &= \arccos \frac{2 \cdot r_c^2 - c^2}{2 \cdot r_c^2}. \end{aligned}$$

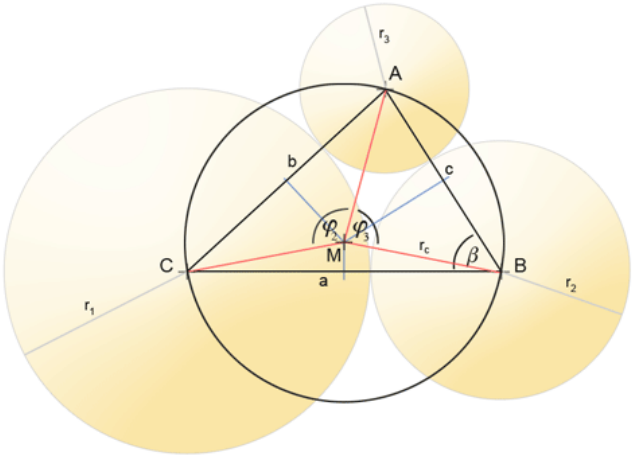


Figure 6 Three child nodes forming an acute triangle: Computation.

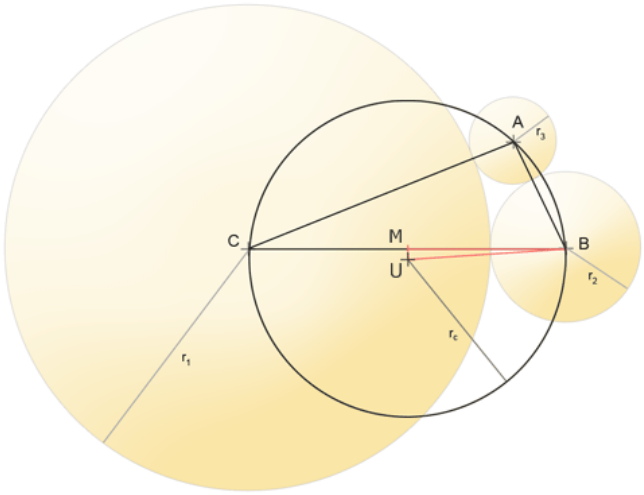


Figure 7 Three child nodes forming a non-acute triangle: Moving the circumcenter on the triangle edge.

If the circumcenter of the triangle lies outside the triangle, we get the situation depicted in Figure 7.

Consider the triangle $\triangle UMB$ where U denotes the circumcenter of the triangle and M the midpoint of the edge $\overline{BC}$. As U lies on the perpendicular bisector of $\overline{BC}$, the triangle has a right angle at the corner M. Therefore, $|\overline{MB}| < |\overline{UB}|$. Thus, using M as center and $|\overline{MB}|$ as the radius of the cone's circle results in a smaller circle. This smallest circle containing all

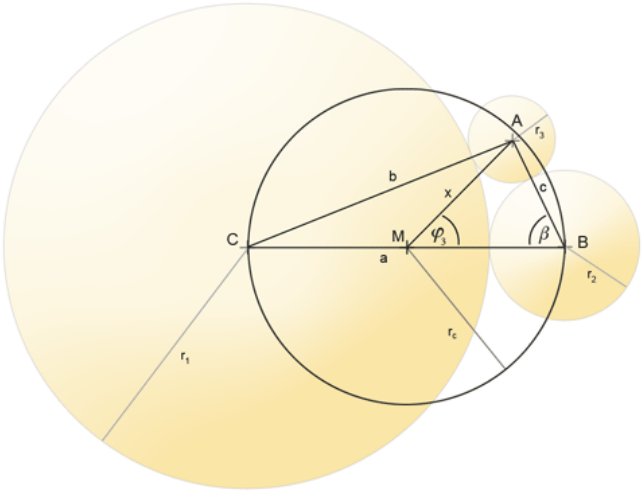


Figure 8 Three child nodes forming a non-acute triangle: Computation.

children can be constructed as follows (see also Figure 8):

$$\begin{aligned} r_c &= \frac{a}{2} \\ r_b &= r_c + r_1 \\ \varphi_1 &= 0° \\ \varphi_2 &= 180° \\ \varphi_3 &= \arccos \frac{x^2 + r_c^2 - c^2}{2 \cdot x \cdot r_c} \end{aligned}$$

with

$$\begin{aligned} \cos\beta &= \frac{a^2 + c^2 - b^2}{2 \cdot a \cdot c} \\ x^2 &= c^2 + r_c^2 - 2 \cdot c \cdot r_c \cdot \cos\beta \\ &= c^2 + r_c^2 - \frac{r_c \cdot (a^2 + c^2 - b^2)}{a}. \end{aligned}$$

In this case, the smallest child with its center A will be moved away from M along a line through M and A (see Figure 9). With respect to the closeness of the children, the circumcircle would be the better solution. But then the radius would be larger.

4.5 Equally Sized Child Nodes

Another special case that can be easily computed is all children having equal size, that is

$$\forall i : r_i = \overline{r}.$$

Then, the children can be placed at the vertices of an equilateral polygon (see Figure 10).

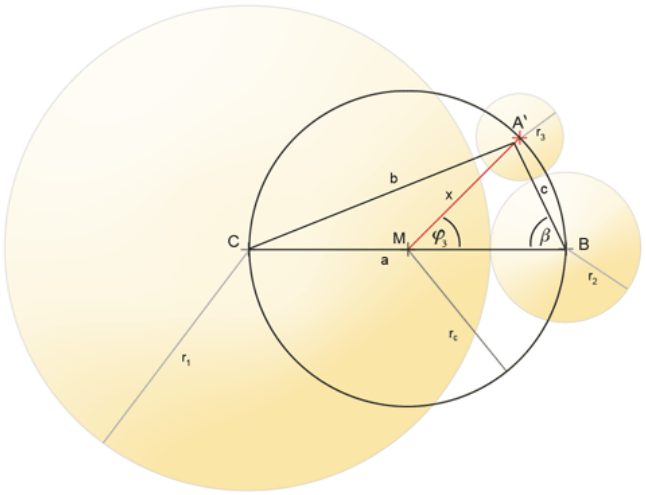


Figure 9 Moving A onto the circle.

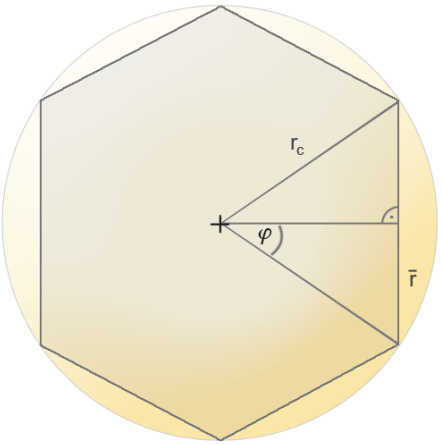


Figure 10 Circumcircle of an equilateral polygon.

The radius of the cone can be computed directly using the following equations:

$$\begin{aligned} \frac{\overline{r}}{\sin\varphi} &= \frac{r_c}{\sin 90^\circ} \\ \varphi &= \frac{2\cdot\pi}{2\cdot n} \\ &= \frac{\pi}{n} \\ &\Downarrow \\ r_c &= \frac{\overline{r}}{\sin\frac{\pi}{n}}. \end{aligned}$$

Remark: A comparison with the approximation given in Section 5.2 yields:

$$\begin{aligned} r'_c &= \frac{2 \cdot \sum_{i=1}^{n} r_i}{2 \cdot \pi} \cdot \frac{\pi}{2} = \frac{n \cdot \overline{r}}{2} \\ r''_c &= \frac{\overline{r}}{\sin \frac{\pi}{n}} \\ &\Downarrow \\ \frac{r'_c}{r''_c} &= \frac{\frac{n \cdot \overline{r}}{2}}{\frac{\overline{r}}{\sin \frac{\pi}{2}}} = \frac{n \cdot \overline{r} \cdot \sin \frac{\pi}{2}}{2 \cdot \overline{r}} = \frac{n \cdot \sin \frac{\pi}{2}}{2} \\ &\Downarrow \\ \lim_{n \to \infty} \frac{r'_c}{r''_c} &= \lim_{n \to \infty} \frac{n \cdot \sin \frac{\pi}{2}}{2} = \frac{\pi}{2}. \end{aligned}$$

This corresponds to the approximation of a circle through an equilateral polygon. Finally, in the limit the circumference of the polygon is equal to the circumference of the circle and thus the error is equal to the compensation factor introduced. On the other hand, for $n = 2$, the quotient is equal to 1 and thus the compensation factor is needed.

5 General Case

The general case allows computing only an approximation of the cone. First, the construction of the circle is given in Section 5.1. In order to compute the radius, a compensation factor is used. A tight bound of this compensation factor is motivated and proven in Section 5.2. Finally, it is possible to improve the construction with a post-processing step. This optimization is presented in Section 5.3.

5.1 Introduction

In case a parent node has more than three children, $n > 3$, an exact computation of the inner circle is no longer possible, except for special cases. One special case would be that all children have the same bounding radius (see Section 4.5). In general, the cone and the positions of the children can be computed as follows (see also Figure 11 and [1]).

The circumference of the cone's base circle can be approximated as

$$\widetilde{c_c} \widetilde{=} 2 \cdot \sum_{i=1}^{n} r_i.$$

An approximation of the radius of the cone's base circle can then be computed as

$$\widetilde{r_c} = \frac{\widetilde{c_c}}{2 \cdot \pi}.$$

The radius used for the cone's base circle is obtained by multiplying the approximated radius by a compensation factor f

$$r_c = \widetilde{r_c} \cdot f.$$

If the number of children is small or if there is a large difference between the radii of the smallest and the largest child, then the circumference will be underestimated. The compensation factor f was motivated and introduced in [1], but no formula for its computation has been given. The special cases for less than four children, $n < 4$, have already been

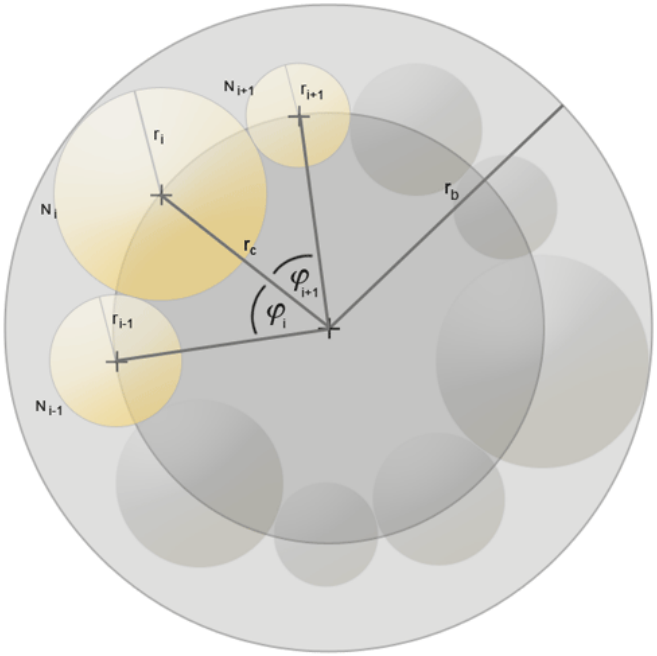


Figure 11 Cone with more than three children.

addressed in Section 4. Here, the maximal error between approximated and minimal circumference needed is computed. It is given by the following equation (see Section 5.2):

$$\epsilon = \frac{c_c}{\widetilde{c_c}} \leq \frac{\pi}{2}.$$

Therefore, choosing

$$f = \frac{\pi}{2}.$$

as compensation factor is sufficient.

The radius of the bounding circle is computed using the following equation:

$$r_b = r_c + \max_{i=1\ldots n} \{r_i\}.$$

Each child is positioned at distance r_c from the center of the cone's circle. The first child is always positioned at $\varphi_1 = 0°$. The angle φ_i between two children N_i and N_{i-1}, $i > 1$ is computed as

$$\varphi_i = \frac{r_{i-1} + r_i}{r_c}.$$

Hence, the children N_i, $i > 1$ are positioned at

$$\overline{\varphi_i} = \sum_{j=1}^{i} \varphi_j.$$

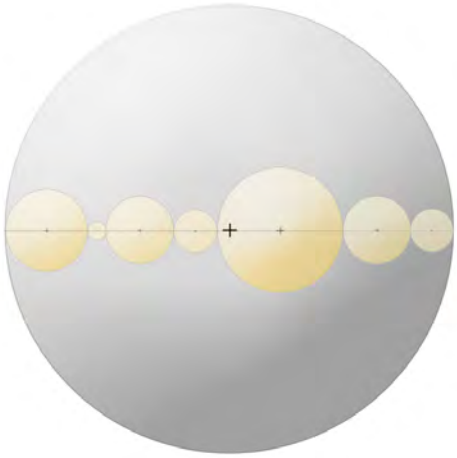


Figure 12 Identification of π as bound of the compensation factor; line-up of the children.

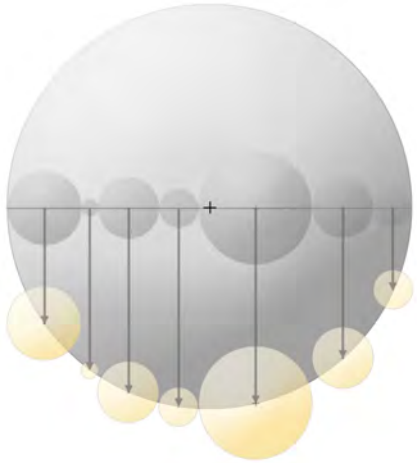


Figure 13 Explanation of π as bound of the compensation factor; moving children onto the boundary.

5.2 Computation of the Compensation Factor

The compensation factor needed in the previous section can be estimated as follows. First of all, we consider the situation depicted in Figure 12. All children are lined up yielding the diameter d of a circle

$$d = 2 \cdot \sum_{i=1}^{n} r_i.$$

Now, the children can be moved perpendicular to the line from the line onto the boundary of the circle (see Figure 13). That is, all children can be placed on a circle having radius r_c

and circumference c_c, such that

$$\begin{aligned} r_c &= \frac{d}{2} = \sum_{i=1}^{n} r_i \\ c_c &= 2 \cdot \pi \cdot r_c = 2 \cdot \pi \cdot \sum_{i=1}^{n} r_i. \end{aligned}$$

The quotient between the chosen circumference c_c and the estimated circumference $\widetilde{c_c}$ is

$$\frac{c_c}{\widetilde{c_c}} = \frac{2 \cdot \pi \cdot \sum_{i=1}^{n} r_i}{2 \cdot \sum_{i=1}^{n} r_i} = \pi.$$

As can be seen from Figure 13, only one half of the circle is used for placing the children. Thus, the compensation factor is much too large. The conjecture is that using

$$f = \frac{\pi}{2}$$

as compensation factor would be sufficient.

Consider the following inequalities:

$$\begin{aligned} \frac{c_c}{\widetilde{c_c}} \leq \frac{\pi}{2} \quad &\Leftrightarrow \quad \frac{2 \cdot \pi \cdot r_c}{2 \cdot \sum_{i=1}^{n} r_i} \leq \frac{\pi}{2} \\ &\Leftrightarrow \quad 2 \cdot \pi \cdot r_c \leq \frac{\pi}{2} \cdot 2 \cdot \sum_{i=1}^{n} r_i \\ &\Leftrightarrow \quad 4 \cdot r_c \leq 2 \cdot \sum_{i=1}^{n} r_i. \end{aligned}$$

In order to show the latter relation, consider the following situations. All children are arranged such that the centers of their bounding circles form a convex polygon. The distance between two children is the sum of their radii. Compute a minimal bounding circle around this convex polygon. Please note that this construction can always be performed. Then, there are two possible situations:

1. Exactly two vertices of the convex polygon are lying on the border of the minimal bounding circle (see Figure 14).
2. Three or more vertices of the convex polygon are lying on the border of the minimal bounding circle (see Figure 15).

If no or only one vertex lies on the border of the circle, then the circle would not be minimal.

Considering the first case, if exactly two vertices of the convex polygon are lying on the border of the bounding circle, then the distance between these vertices is equal to the diameter of the circle. Otherwise, the bounding circle would not be minimal.

Let A and B be the vertices lying on the minimal bounding circle. Then, we get

$$r_c = \frac{|\overline{AB}|}{2}$$

and the above equation is certainly fulfilled:

$$4 \cdot r_c = 4 \cdot \frac{\overline{|AB|}}{2} = 2 \cdot \overline{|AB|} \leq 2 \cdot \sum_{i=1}^{n} r_i.$$

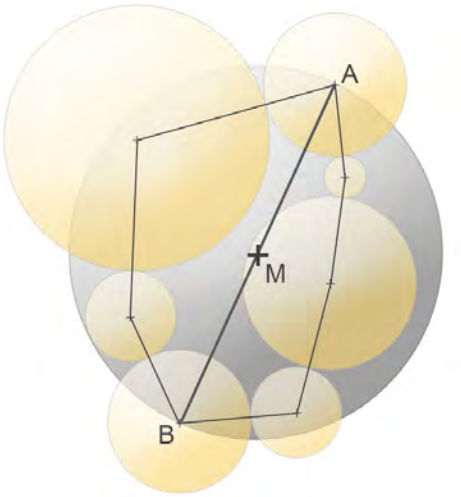


Figure 14 Explanation of the compensation factor: first case.

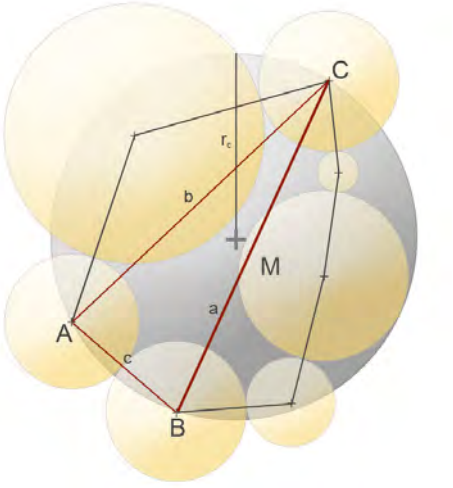


Figure 15 Explanation of the compensation factor: second case.

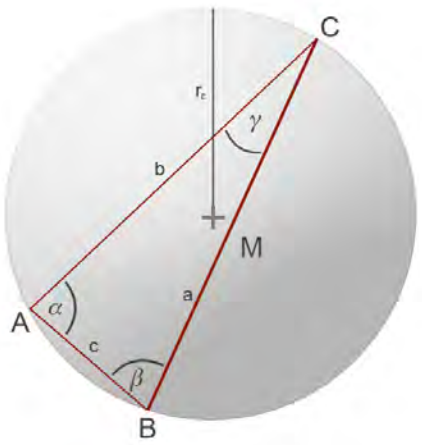


Figure 16 Explanation of the compensation factor: second case.

This holds because the path from A to B using lines of the polygon is certainly longer than the direct path between A and B (triangle inequality).

If three or more vertices are lying on the boundary of the minimal bounding circle, then the result is obtained as follows. Consider any triangle with all vertices lying on the boundary. Then, the bounding circle is also the circumcircle of the triangle. There is at least one triangle with the circumcenter lying inside the triangle or on one of its sides. Otherwise, for each triangle, the circumcenter lies outside the triangle. This is the second situation of the special case described in Section 4.4 and the circle would not be minimal.

Now, choose a triangle such that the circumcenter lies inside the triangle or on one of its sides. If the center of the circumcircle lies on one of the sides of the triangle, then this side is the diameter of the circle and we have the same situation as in the first case. Otherwise, let a, b, c be the sides of the triangle and A, B, C its vertices.

If

$$4 \cdot r_c \leq |a| + |b| + |c|$$

then

$$4 \cdot r_c \leq |a| + |b| + |c| \leq 2 \cdot \sum_{i=1}^{n} r_i.$$

The last inequality holds because a, b, and c are the shortest connections between A, B, and C. Every path on the polygon is longer (triangle inequality). Thus, it is sufficient to show that the first inequality holds.

Consider Figure 16. First of all, the law of sines implies the following equations:

$$\begin{aligned} |a| + |b| + |c| &= 2 \cdot r_c \cdot \sin\alpha + 2 \cdot r_c \cdot \sin\beta + 2 \cdot r_c \cdot \sin\gamma \\ &= 2 \cdot r_c \cdot (\sin\alpha + \sin\beta + \sin\gamma). \end{aligned}$$

Thus, it is sufficient to show

$$\sin\alpha + \sin\beta + \sin\gamma \geq 2$$

because then

$$\begin{aligned} |a| + |b| + |c| &= 2 \cdot r_c \cdot (\sin\alpha + \sin\beta + \sin\gamma) \\ &\geq 2 \cdot r_c \cdot 2 \\ &= 4 \cdot r_c. \end{aligned}$$

In order to show that the sum of the sine of the angles is greater or equal to two, consider the following conditions. As the triangle is acute, we have

$$A, B, C \leq 90°.$$

Therefore, we get

$$A + B \geq 90° \Rightarrow A \geq 90° - B.$$

From this follows

$$\begin{aligned} \sin\alpha &\geq \cos\beta \\ \sin\beta &\geq \cos\alpha. \end{aligned}$$

Using these, we get

$$\begin{aligned} \sin\alpha &+ \sin\beta + \sin\gamma \\ &= \sin\alpha + \sin\beta + \sin(180° - (\alpha + \beta)) \\ &= \sin\alpha + \sin\beta + \sin(\alpha + \beta) \\ &= \sin\alpha + \sin\beta + \sin\alpha \cdot \cos\beta + \cos\alpha \cdot \sin\beta \\ &\geq \sin\alpha + \sin\beta + \cos^2\beta + \cos^2\alpha \\ &= \sin\alpha + \sin\beta + 1 - \sin^2\beta + 1 - \sin^2\alpha \\ &= 2 + \sin\alpha \cdot (1 - \sin\alpha) + \sin\beta \cdot (1 - \sin\beta) \\ &\geq 2. \end{aligned}$$

From this follows the claim about the compensation factor at the beginning of this section.

5.3 Optimization Step

Having computed the layout as described in Section 5.1, an optimization step can be performed (see Figure 17). This is done by computing for each child an optimization factor

$$\begin{aligned} f_i &= \frac{r_i + r_{i+1}}{d(r_i, r_{i+1})} \\ &= \frac{r_i + r_{i+1}}{\sqrt{r_c \cdot r_c + r_c \cdot r_c - 2 \cdot r_c \cdot r_c \cdot \cos\varphi}} \\ &= \frac{r_i + r_{i+1}}{\sqrt{(2 - 2 \cdot \cos\varphi) \cdot r_c^2}}. \end{aligned}$$

where $d(r_i, r_{i+1})$ denotes the distance between r_i and r_{i+1}. Then, an optimized radius can be computed and used for the layout of the children:

$$\hat{r}_c = \max(f_i) \cdot r_c.$$

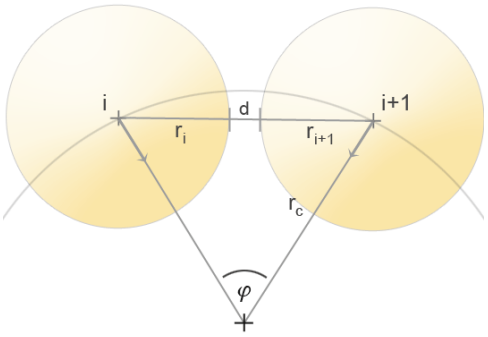


Figure 17 Optimization step.

6 Future research and open problems

Although the layout computed is optimal under the assumptions presented in Section 3 and Section 5.1, there is still space for improvements.

First of all, the proof for the general case is quite long. The question is whether there is a more elegant proof of the bound for this case.

The second question is whether there is an easy way to extend the special cases. Further, only a small number of special cases have been considered, namely, zero to three children and equally sized children. There might be further special cases, leading to simple and efficient solutions.

While this paper focused on the computation of the circle parameters, another parameter is the height of the cone. The question is whether there is an elegant way to compute an optimal height.

Finally, the implications of this work on RDTs can be researched, too.

7 Conclusion

Performing a bottom-up construction of cone trees, formulas for the optimal computation of the cone's base circles have been proposed. This includes special cases for zero to three children and for equally sized children as well as the general case. For the latter, an optimal compensation factor has been motivated and proven. Further, an optimization step has been introduced for this case. This allows implementing cone trees bottom-up in an optimal way.

8 Acknowledgment

Many thanks go to Timo Klein who provided the illustrations of this paper. The authors also wish to thank the participants of the Dagstuhl Seminar 07221 "Information Visualization - Human-Centered Issues in Visual Representation, Interaction, and Evaluation" for valuable comments on a presentation of a preliminary version of this paper. Further, we thank the unknown referees of TVCG of valuable comments on this paper.

References

1 Jeromy Carriere and Rick Kazman. Interacting with huge hierarchies: Beyond cone trees. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'95)*, pages 74–81, Atlanta, 1995.

2 Andy Cockburn and Bruce McKenzie. An evaluation of cone trees. In *People and Computers XV: Proceedings of the British Computer Society Conference on Human Computer Interaction 2000*, pages 425–436. Springer Verlag, 2000.

3 Chang-Sung Jeong and Alex Pang. Reconfigurable disc trees for visualizing large hierarchical information space. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'98)*, pages 19–25, North Carolina, 1998.

4 George G. Robertson, Jock D. Mackinlay, and Stuart Card. Cone trees: Animated 3D visualizations of hierarchical information. In *Proceedings of ACM CHI'91*, pages 189–194, New Orleans, USA, 1991.

Generalized Swap Operation for Tetrahedrizations

Burkhard Lehner[1], Bernd Hamann[2], and Georg Umlauf[3]

1 **Department of Computer Science, University of Kaiserslautern, Germany**
`lehner@cs.uni-kl.de`
2 **Institute for Data Analysis and Visualization (IDAV), Department of Computer Science, University of California, Davis, USA**
`hamann@cs.ucdavis.edu`
3 **Department of Computer Science, HTWG Constance, Germany**
`umlauf@htwg-konstanz.de`

Abstract

Mesh optimization of 2D and 3D triangulations is used in multiple applications extensively. For example, mesh optimization is crucial in the context of adaptively discretizing geometry, typically representing the geometrical boundary conditions of a numerical simulation, or adaptively discretizing the entire space over which various dependent variables of a numerical simulation must be approximated. Together with operations applied to the vertices the so-called edge or face swap operations are the building block of all optimization approaches. To speed up the optimization or to avoid local minima of the function measuring overall mesh quality these swaps are combined to generalized swap operations with a less local impact on the triangulation.

Despite the fact that these swap operations change only the connectivity of a triangulation, it depends on the geometry of the triangulation whether the generalized swap will generate inconsistently oriented or degenerate simplices. Because these are undesirable for numerical reasons, this paper is concerned with geometric criteria that guarantee the generalized swaps for a 3D triangulation to yield only valid, non-degenerate triangulations.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases 3D Triangulation, Geometric Conditions, Swap Operations

Digital Object Identifier 10.4230/DFU.SciViz.2010.30

1 Introduction

Triangulations of points in 2D space for a mesh of triangles or points in 3D space for a mesh of tetrahedra are crucially important for numerous applications encountered in scientific and engineering application, including numerical simulation, shape approximation, or visualization. In scattered data approximation [15, 8, 20] 2D triangulations are used to define a piecewise linear coarse approximation of a dense data set, assigning a "height value" for every vertex. This technique can also be used for image compression [5, 4, 21, 18] and video compression [19, 17]. For reverse engineering [12, 9, 6, 1], the 2-manifold surface to be reconstructed is approximated by a 3D triangulation that contains no tetrahedra. For mechanical engineering and physical simulations [24, 14], 3D triangulations are used as meshes for finite element methods.

For all of these applications the triangulation needs to be optimized with respect to an application-dependent cost function measuring mesh quality based on a multitude of proper mesh quality variables, including, for example, point distribution, approximation error [7, 18], triangle shape [10], dihedral angles [14], etc. The optimization process is usually based on simple, local changes in the triangulations such as repositioning of vertices [15],

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 30–44

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

insertions and removal of vertices [7, 11] and edge and face swaps [22]. While the first of these operations change geometry and connectivity of the triangulation the swaps change only the connectivity of a triangulation. To speed up the optimization or to avoid local minima during mesh optimization multiple edge and face swaps are combined to generalized swap operations that change the connectivity of more than three tetrahedra of the triangulation [13, 25, 23, 17], see Section 2.

However, it depends on the geometry of the triangulation if a generalized swap will generate flipped or degenerate simplices. We present in this paper geometric criteria that guarantee that a generalized swap operation in a 3D triangulation will generate only valid, non-degenerate triangulations.

2 Related Work

In general, a *swap operation* replaces d-dimensional simplices of a triangulation ($d \geq 1$) by other simplices. It usually affects only a local area of the triangulation, and changes the connectivity of the triangulation without changing the number or position of the vertices.

Lawson [16] was was among the first scientists studying and publishing swap operations systematically. He showed that $d+2$ points in d dimensions, which do not all lie in a hyper-plane, have either one unique triangulation $\mathcal{T}$ or two possible triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$. Which case happens depends on the vertex positions, see Figure 1 for the 2D case. In the latter case, $\mathcal{T}_1$ and $\mathcal{T}_2$ differ only in connectivity and the transformation from $\mathcal{T}_1$ to $\mathcal{T}_2$ is called swap operation $s_{1\to2}(\mathcal{T}_1) = \mathcal{T}_2$. The opposite transformation is $s_{2\to1}(\mathcal{T}_2) = \mathcal{T}_1$. Because $s_{1\to2} \circ s_{2\to1} = s_{2\to1} \circ s_{1\to2} = \mathrm{id}$, $s_{1\to2}$ and $s_{2\to1}$ are inverse operations.

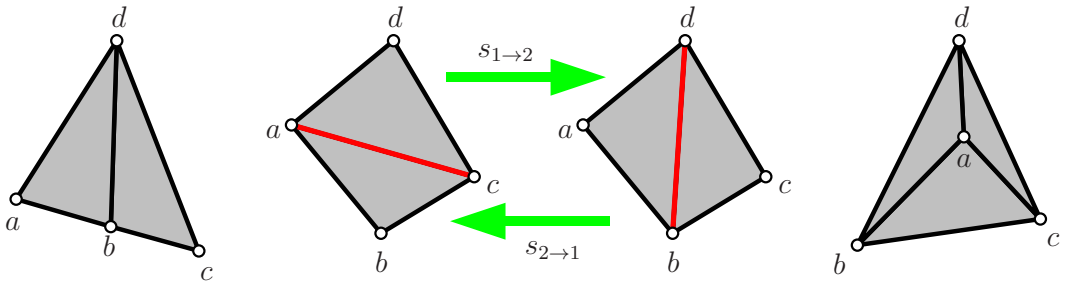


Figure 1 Triangulations of four points in the 2D case.

If $\mathcal{T}_1$ is a subset of a larger triangulation $\mathcal{T}$, the swap operation can be applied by replacing only the simplices of $\mathcal{T}_1$ with those of $\mathcal{T}_2$, and leaving all simplices of $\mathcal{T}$ unchanged, i.e., $\mathcal{T}' = (\mathcal{T} \setminus \mathcal{T}_1) \cup \mathcal{T}_2$. Note that the subset $\mathcal{T}_1$ has to be a triangulation, i.e. it has to fill the convex hull of its vertices, and must be convex.

Additionally to these basic swaps, one can construct generalized swap operations that replace a set of simplices C of the triangulation by a different set of simplices C'. Thus, C and C' are not required to cover the convex hull of their vertices. Since the generalized swaps are usually more powerful, they can lead to a good triangulation with less swap operations, but are often less efficient.

One way to construct a generalized swap operation is to combine a sequence of basic swap operations to a so-called *composed swap operation*. For the 2D case Yu et al. [25] use a combination of two edge swap operations. If a simple edge swap does not reduce the cost function, they swap the edge and one of its adjacent edges. Thus, the affected faces do not

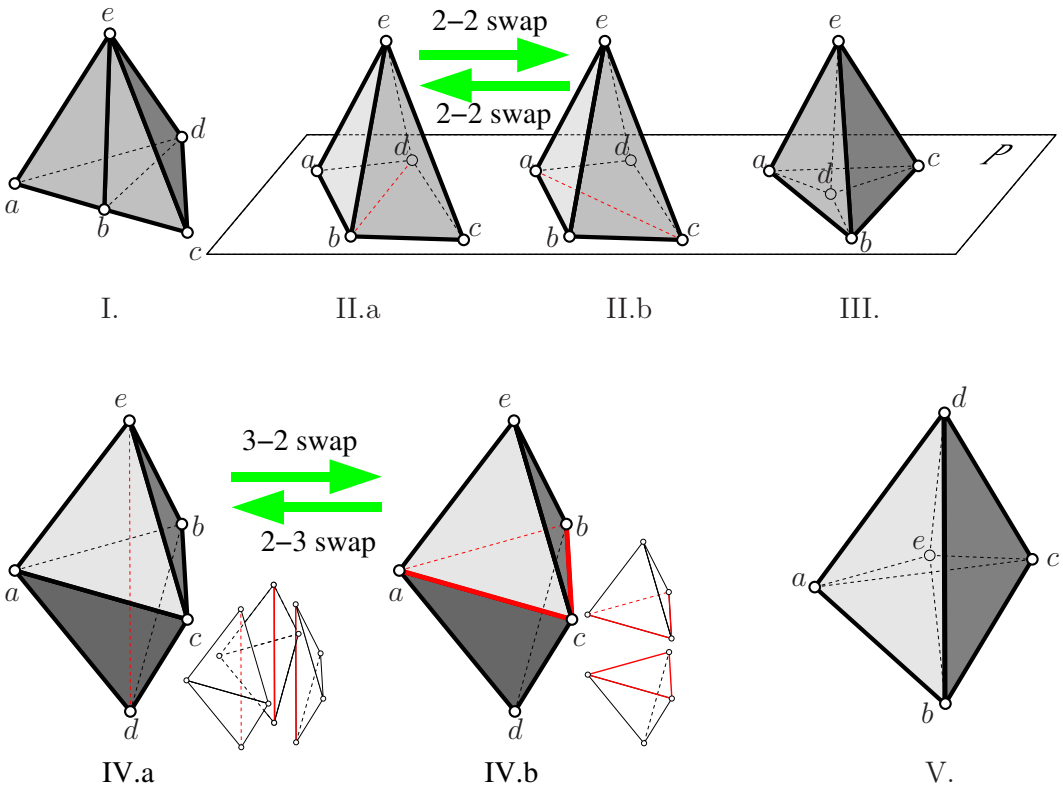


Figure 2 The different settings of five points in the 3D case.

need to form a convex polygon for the composed swap operations. Using the composed swap operations can improve the optimization results significantly.

Concerning the 3D case, the set of swap operations is larger and more varied than in the 2D case. Again, we can categorize them into basic swap operations and composed swap operations. According to Lawson [16], there are five different settings of five points a, b, c, d and e in 3D space, only two of which have two different triangulations and therefore provide swap operations, see Figure 2. If three points are collinear, or four points a, b, c, d are coplanar with $d \in \text{conv}(a, b, c)$, or $e \in \text{conv}(a, b, c, d)$ there is only one possible triangulation, see Figures 2 I., III., and V. If exactly four points are coplanar and form a convex quadrilateral q there are two possible triangulations with flipped diagonals of q, see Figure 2 II. Because the triangulation consists of two cells before and after the swap, the swap is called a *2-2 swap*. For the most general case in which all five points are corners of $\text{conv}(a, b, c, d, e)$ there are also two possible triangulations, see Figure 2 IV. Because this swap replaces three cells by two and vice versa, it is called a *3-2 swap* or *2-3 swap*, respectively.

When applied to a subset of a triangulation $\mathcal{T}$, the 2-2 swap is only possible if the two faces $\{a, b, d\}$ and $\{b, c, d\}$ are border faces of $\mathcal{T}$. If they are interior faces, the incident two cells also have to be swapped, see Figure 3. This leads to the 4-4 swap, which replaces four cells with four other cells.

In 3D also a combination of basic swap operations can be more powerful. Joe [13] systematically analyzed the possible settings. Every face of a triangulation is assigned to nine different categories, describing their local setting and their status of being transformable by a basic swap operation. He proposes a set of composed swap operations to transform faces that

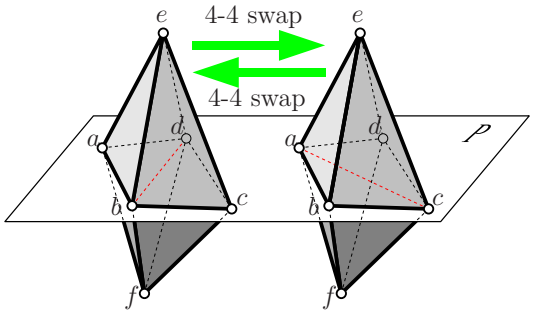


Figure 3 The 4-4 swap is used if the faces of the 2-2 swap are no border faces.

are initially not transformable, by first swapping adjacent faces. For every composed swap operation, he lists the cells that are removed and created. From this list, he provides criteria in [13] to compute the change of a cost function c resulting from each of the operations, if c is the minimum of the costs of the individual cells.

Another class of composed swap operations is the class defined by the generalizations of the 3-2 and 2-3 swaps, see [24, 3].

Generalized 3-2 swap ($\mathcal{G}_{32}$) A *generalized 3-2 swap ($\mathcal{G}_{32}$)* can be applied to an edge $e = \{a, b\}$ with $n \geq 3$ incident cells $C = \{c_1, \ldots, c_n\}$, with $c_i = \{a, b, v_i, v_{i+1}\}$ and $v_{n+1} \equiv v_1$, see Figure 4 (left). The loop $(v_1, \ldots, v_n)$ is split into a set of $n-2$ connected faces $F = \{f_1, \ldots, f_{n-2}\}$. Note that the choice of F is not unique. $\mathcal{G}_{32}$ replaces the edge e with the faces F, where the n cells C are replaced by the $2(n-2)$ cells $C' = \{c'_{a,1}, c'_{b,1}, \ldots, c'_{a,n-2}, c'_{b,n-2}\}$ with $c'_{a,i} = f_i \cup \{a\}$ and $c'_{b,i} = f_i \cup \{b\}$.

Generalized 2-3 swap ($\mathcal{G}_{23}$) We say a face $f = \{v_1, v_2, v_3\}$ is *sandwiched* between vertices a and b, if the two cells incident to f are $c_1 = \{a, v_1, v_2, v_3\}$ and $c_2 = \{b, v_1, v_2, v_3\}$. A *generalized 2-3 swap ($\mathcal{G}_{23}$)* is applied to a set $F = \{f_1, \ldots, f_{n-2}\}$ of faces, which are sandwiched between two points a and b, see Figure 4 (right). A new edge $e = \{a, b\}$ is inserted into the triangulation, and the border edges of F are connected to the new edge e to form the new cells. Let $C' = \{c'_{a,1}, c'_{b,1}, \ldots, c'_{a,n-2}, c'_{b,n-2}\}$ be the set of cells incident to the faces $c'_{a,i} = f_i \cup \{a\}$ and $c'_{b,i} = f_i \cup \{b\}$ of F, and $(v_1, \ldots, v_n)$ be the loop of vertices defined by the border edges of F. $\mathcal{G}_{23}$ replaces the faces of F by the edge $\mathbf{e} = \sigma_{\{a,b\}}$, and the $2(n-2)$ cells of C' are replaced by the n cells $C = \{c_1, \ldots, c_n\}$, with $c_i = \{a, b, v_i, v_{i+1}\}$, and $v_{n+1} \equiv v_1$.

$\mathcal{G}_{23}$ is the inverse of $\mathcal{G}_{32}$. Since the choice of faces is not unique in either direction, applying the one swap operation after the other leads to the start triangulation only if for both swaps the same faces are chosen. Also note that the 2-3 swap is a special case of $\mathcal{G}_{23}$, the 3-2 swap of $\mathcal{G}_{32}$, and the 4-4 swap a special case of $\mathcal{G}_{23}$ and $\mathcal{G}_{32}$.

The execution of $\mathcal{G}_{32}$ and $\mathcal{G}_{23}$ can result in invalid triangulations. In Sections 4 and 5 we discuss necessary and sufficient geometric conditions to ensure the validity of the resulting triangulation. Shewchuk [23] notes that these swaps can be replaced by a series of 2-3 and 3-2 swaps, where the intermediate triangulations are topologically correct, but may contain degenerate or inverted cells. In Section 6 we show that there is always a sequence of 2-3, 3-2, and 4-4 swaps to replace a $\mathcal{G}_{23}$ or $\mathcal{G}_{32}$ swap without degenerate or inverted cells.

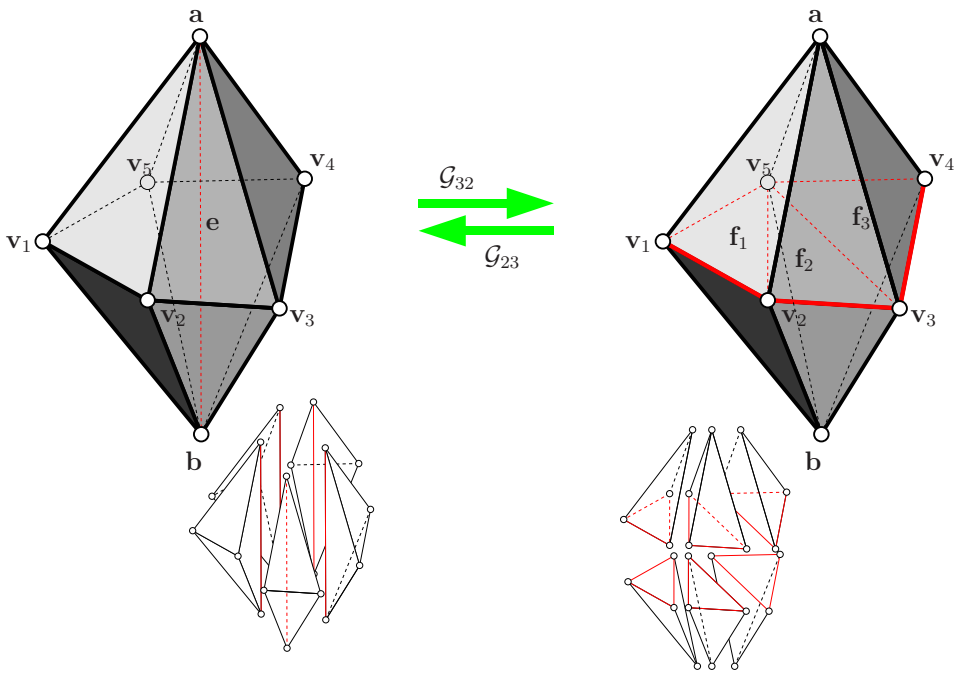


Figure 4 The generalized 3-2 and 2-3 swaps.

3 Notation

In order to define the generalized swap operation in terms of connectivity changes and associated geometric conditions, we first adjust our notation properly.

A *3D triangulation* $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{C})$ (tetrahedrization) consists of a set of *vertices* $\mathcal{V}$, edges $\mathcal{E} \subset \mathcal{V}^2$, faces $\mathcal{F} \subset \mathcal{V}^3$ (triangles), and cells $\mathcal{C} \subset \mathcal{V}^4$ (tetrahedra). Thus, an edge is a pair of vertices, a face a triple of vertices, and a cell a quadruple of vertices. All these entities are ordered such that $\mathcal{T}$ is an oriented simplicial 3-complex, where the edges of adjacent faces and the faces of adjacent cells are order reversely. In this case, we call $\mathcal{T}$ a *valid* triangulation. We will use set operations to define new faces and cells, i.e., for $v_1 \in \mathcal{V}$, $e = (v_2, v_3) \in \mathcal{E}$, $f = (v_2, v_3, v_4) \in \mathcal{F}$ and $c \in \mathcal{C}$ we define

$$\begin{aligned} e \cup \{v_1\} &= (v_1, v_2, v_3) \in \mathcal{F}, \\ f \cup \{v_1\} &= (v_1, v_2, v_3, v_4) \in \mathcal{C} \end{aligned}$$

and

$$\begin{aligned} e \in f &\iff (v_2, v_3) \text{ is a sub-tuple of } f, \\ f \in c &\iff (v_2, v_3, v_4) \text{ is a sub-tuple of } c. \end{aligned}$$

While $\mathcal{V}$, $\mathcal{E}$, $\mathcal{F}$, and $\mathcal{C}$ describe only the connectivity of the triangulation, a geometric realization of $\mathcal{T}$ is defined by associating a point $\mathbf{v} \in \mathbb{R}^3$ to every vertex $v \in \mathcal{V}$. The geometric realizations of an edge $e \in \mathcal{V}^2$, a face $f \in \mathcal{V}^3$, or a cell $c \in \mathcal{V}^4$ are then defined as the convex hull of the geometric realizations of their vertices, and are also denoted in boldface letters $\mathbf{e}$, $\mathbf{f}$, and $\mathbf{c}$, respectively. Furthermore, for a set M of edges, faces, or cells, we denote by $\mathbf{M}$ the union of the geometric realizations of the elements of M. Throughout this paper, geometric realizations of elements of a triangulation are denoted by boldface letters.

We say a valid triangulation $\mathcal{T}$ is *consistently oriented*, when the geometric realizations of all cells have the same geometric orientation. The orientation of a cell induces a notion of orientation on all of its contained k-sub-simplices for $k = 1, 2$. A k-sub-simplex is called positively oriented if it is positively oriented in the k-dimensional hyperplane bounding the enclosing $(k+1)$-sub-simplex with outward pointing normal. This means in particular, that all faces of a cell are positively oriented with respect to the half-plane bounding the cell with a normal pointing to the outside of the cell. If the vertices of a cell are not affinely independent, it is called *degenerate*, and if a cell or any of its k-sub-simplices are not positively oriented, we call it *inconsistently oriented*.

Border faces are faces of a triangulation $\mathcal{T}$ that are incident to only one cell in $\mathcal{T}$, all other faces are called *inner faces*. Analogously, *border edges* are incident to only one inner face, all other edges are called *inner edges* The *border of a triangulation* $\mathcal{T}$ is the set of all its border faces. If $\mathcal{T}$ is a valid, consistently oriented triangulation, the geometric realization of its border is a 2-manifold.

The *boundary* ∂S of a subset S of a manifold M are the points in S for which every ε-ball in M contains points in $M \setminus S$. Note that the term *border* is an attribute of the connectivity of a triangulation, whereas *boundary* is a property of its geometric realization.

We need to provide some definitions concerning spherical projections, which we will use to establish geometric conditions for allowable swap operations.

▶ **Definition 1.** *The* spherical projection *of a point* $\mathbf{p} \in \mathbb{R}^3$ *onto the sphere* $S^{\mathbf{q}}$ *with center* $\mathbf{q} \in \mathbb{R}^3$ *and radius* r *is defined as*

$$\Pi^{\mathbf{q}}(\mathbf{p}) = \mathbf{q} + r(\mathbf{p} - \mathbf{q})/\|\mathbf{p} - \mathbf{q}\|_2, \quad \mathbf{p} \neq \mathbf{q}.$$

A projection of a set of points $P \subset \mathbb{R}^3 \setminus \{\mathbf{q}\}$ *is the set of the projected points,*

$$\Pi^{\mathbf{q}}(P) = \{\Pi^{\mathbf{q}}(\mathbf{p}) | \mathbf{p} \in P\}.$$

Some properties of the spherical projection (without proof) are:

- If P is a line, $\Pi^{\mathbf{q}}(P)$ is either two antipodal points (for $\mathbf{q} \in P$), or a half great circle (for $\mathbf{q} \notin P$) of $S^{\mathbf{q}}$.
- If P is a plane, $\Pi^{\mathbf{q}}(P)$ is either a great circle (for $\mathbf{q} \in P$), or an open half sphere (for $\mathbf{q} \notin P$) of $S^{\mathbf{q}}$.
- If $P = \text{conv}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ is a triangle and the plane defined by P does not contain $\mathbf{q}$, $\Pi^{\mathbf{q}}(P)$ is a spherical triangle, bounded by the projection of the edges $\Pi^{\mathbf{q}}(\text{conv}(\mathbf{p}_1, \mathbf{p}_2))$, $\Pi^{\mathbf{q}}(\text{conv}(\mathbf{p}_2, \mathbf{p}_3))$, $\Pi^{\mathbf{q}}(\text{conv}(\mathbf{p}_3, \mathbf{p}_1))$, which are segments of great circles of $S^{\mathbf{q}}$.

4 Geometric Conditions for $\mathcal{G}_{32}$

For the geometric conditions to be satisfied for a $\mathcal{G}_{32}$-swap as defined in Section 2 we have an edge $e = (a, b)$ with n incident cells that is swapped. The triangulation before and after the $\mathcal{G}_{32}$-swap is denoted by $\mathcal{T}$ and $\mathcal{T}'$.

▶ Condition 1. The triangulation $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{C})$ is valid, and all cells of $\mathcal{T}$ have positive orientation.

▶ Condition 2. The edge e is an inner edge of $\mathcal{T}$, i.e., every face f incident to e is incident to exactly two cells $c_{f,1} \neq c_{f,2}$.

Note that the last condition implies that e is not on the border of T. Furthermore, these conditions induce an order of the faces incident to e.

▶ **Lemma 2.** *All faces containing e can be ordered to form a cyclic sequence $G = (g_1, \ldots, g_n)$, i.e., the index $i = 1, \ldots, n$ of g_i is understood modulo n. Furthermore, the dihedral angles θ_i between $\mathbf{g}_i$ and $\mathbf{g}_{i+1}$ (in the direction a to b) are in the interval $(0, \pi)$, and sum to 2π.*

Proof. Due to Condition 2, a face $g = (a, b, v)$ incident to e is incident to two cells $c_{g,1}, c_{g,2}$. Both have two faces incident to e, one of the two is g, the other ones are g'_1 and g'_2, respectively. The successor of g is the face g'_k of cell $c_{g,k}$ on the positive side of g (in the direction a to b), $k = 1, 2$. The predecessor of g is the other face. Due to Condition 2 this relation determines a cyclic successor-graph without branches.

The dihedral angle θ_i between a face $\mathbf{g}_i$ and its successor $\mathbf{g}_{i+1}$ is the dihedral angle at e of the cell that contains both faces. Therefore, $0 < \theta < \pi$, because otherwise the cell would be inverted or degenerate, contradicting Condition 1.

Since the sequence of faces is cyclic, it surrounds $\mathbf{e}$. It can only cycle exactly once around $\mathbf{e}$, because otherwise cells between the faces would intersect in their interior, which contradicts Condition 1. The sum of the dihedral angles between the faces is therefore 2π. ◀

We denote the cell between g_i and g_{i+1} as c_i, and the third vertex of g_i as v_i. Thus, Lemma 2 induces also a cyclic order on the cells $C = (c_1, \ldots, c_n)$ and vertices $V = (v_1, \ldots, v_n)$ around e. Because $\mathcal{G}_{32}$ replaces the cells of C by other cells, we call $\mathbf{C}$ the *affected region*, and the border faces of it are given by

$$\partial C := \{(a, v_2, v_1), (b, v_1, v_2), \ldots, (a, v_1, v_n), (b, v_n, v_1)\},$$

i.e., $\partial\mathbf{C} = \bigcup_{f \in \partial C} \mathbf{f}$. The line through a and b is denoted by

$$\mathbf{l} = \{\mathbf{a} + \lambda(\mathbf{b} - \mathbf{a}) | \lambda \in \mathbb{R}\}. \tag{1}$$

▶ **Lemma 3.** *There is a closed loop of edges $B = \{b_1, \ldots, b_n\}$ that winds around $\mathbf{l}$ exactly once.*

Proof. This follows from Lemma 2, where b_i is the edge of c_i opposite to e. ◀

For the sphere $S^{\mathbf{a}}$ around $\mathbf{a}$ contained in the convex hull of all cells containing $\mathbf{a}$ we set $t^{\mathbf{a}}_{\mathrm{I}} = \Pi^{\mathbf{a}}(\mathbf{b})$ and denote by $t^{\mathbf{a}}_{\mathrm{O}}$ the antipodal point of $t^{\mathbf{a}}_{\mathrm{I}}$. Let $\mathbf{B}^{\mathbf{a}} = \Pi^{\mathbf{a}}(\mathbf{B})$ the spherical projection of $\mathbf{B}$ onto $S^{\mathbf{a}}$. Since $\mathbf{B}^{\mathbf{a}}$ is a closed loop on $S^{\mathbf{a}}$, it splits $S^{\mathbf{a}}$ into two parts $S^{\mathbf{a}}_{\mathrm{I}}$ and $S^{\mathbf{a}}_{\mathrm{O}}$, which are characterized by $t^{\mathbf{a}}_{\mathrm{I}} \in S^{\mathbf{a}}_{\mathrm{I}}$ and $t^{\mathbf{a}}_{\mathrm{O}} \in S^{\mathbf{a}}_{\mathrm{O}}$, see Figure 5. Analogously, $S^{\mathbf{b}}$, $t^{\mathbf{b}}_{\mathrm{I}}$, $t^{\mathbf{b}}_{\mathrm{O}}$, $\mathbf{B}^{\mathbf{b}}$, $S^{\mathbf{b}}_{\mathrm{I}}$, and $S^{\mathbf{b}}_{\mathrm{O}}$ are defined.

▶ **Definition 4.** *A* partition *of B is a set $F = \{f_1, \ldots, f_m\}$ of faces $f_i \notin \mathcal{F}$, where*

1. *all vertices of f_i belong to edges of B, i.e., $f_i \subset V$,*
2. *all edges of f_i are either edges of B or inner edges I, and*
3. **a.** *every edge of B is incident to exactly one face of F,*
   b. *every edge of I is incident to exactly two faces of F.*

▶ **Lemma 5.** *Every partition F of B has $n - 3$ inner edges and $m = n - 2$ faces.*

Proof. As a consequence of Lemma 2 partitioning B is equivalent to a triangulation of a simple polygon $\mathbf{B}'$ in a plane perpendicular to $\mathbf{l}$ without introducing new vertices. This polygon is the orthogonal projection of $\mathbf{B}$ along direction $\mathbf{l}$. Since every simple polygon with n vertices can be triangulated with $n - 2$ triangles (see [2]), i.e., $n - 3$ inner edges, the claim follows. ◀

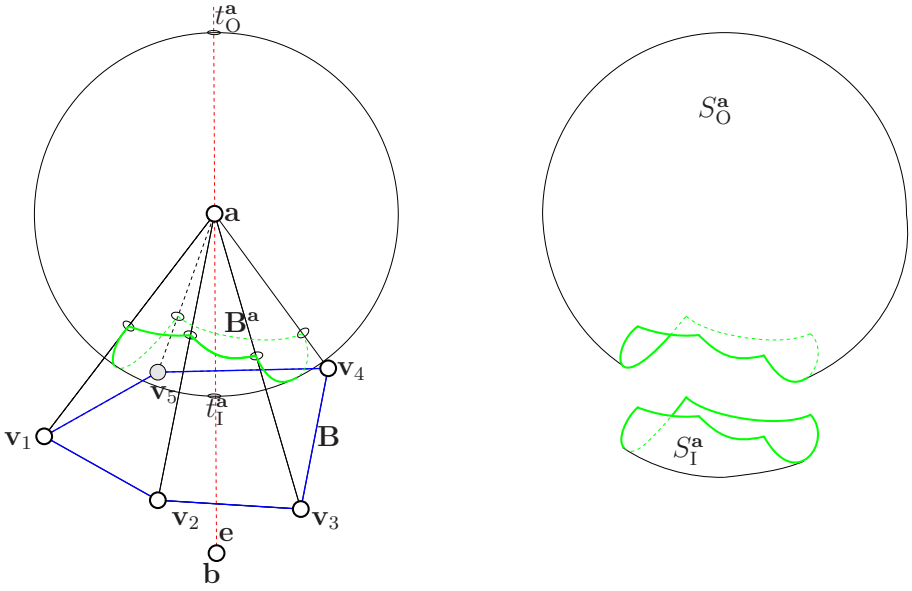


Figure 5 Terms used in spherical projection with $\mathbf{B}$ in blue and $\mathbf{B}^{\mathbf{a}}$ in green.

The partition F of B defines the cells that are created by the $\mathcal{G}_{32}$-swap. Every face of the partition is connected to a and b to form two new cells. The set of new cells is $C' = \{c'_{a,1}, c'_{b,1}, \ldots, c'_{a,m}, c'_{b,m}\}$ with $c'_{a,j} = f_j \cup \{a\}$ and $c'_{b,j} = \overline{f_j} \cup \{b\}$ for $f_j \in F$. Note that for $n > 3$ the partitions and also the $\mathcal{G}_{32}$-swap is not unique.

It can happen that C' contains inconsistently oriented or degenerate cells. Therefore, the $\mathcal{G}_{32}$-swap would result in an invalid triangulation and must not be applied. Whether this is happens depends on e and B and also on the choice of F. We call F a *valid partition* if all cells in C' are valid.

Depending on the geometry, there are three different cases. For every case we present an example for $n = 4$, so that two different partitions exist: $F_1 = \{(v_1, v_2, v_3), (v_1, v_3, v_4)\}$ and $F_2 = \{(v_1, v_2, v_4), (v_2, v_3, v_4)\}$. For every example, $\mathbf{a} = (0, 0, 1)$ and $\mathbf{b} = (0, 0, -1)$. Furthermore, the x and y coordinates of $\mathbf{v}_1$ to $\mathbf{v}_4$ are $(-0.3, -0.3)$, $(0.7, -1.3)$, $(1.7, -0.3)$, and $(0.7, 0.7)$, respectively.

Every partition is valid For every partition F, all cells in C' are valid. For our example, we choose the z coordinates to be $z_1 = z_2 = z_3 = z_4 = 0$. Both partitions F_1 and F_2 are valid in this case. Note, that every partition is valid as long as the affected region $\mathbf{C}$ is convex. which is only the case if (as in this example) all $\mathbf{v}_i$ are coplanar. But also for a non-convex affected region all partitions can be valid.

Some partitions are invalid For some partitions, there are cells in C' that are inverted or degenerate. But other partitions are valid. For a concrete example, set the z coordinates to $z_1 = z_2 = z_3 = 0.8$ and $z_4 = -0.8$. Here, F_1 is an invalid partition, because the cell $(\mathbf{a}, \mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_4)$ is inverted, while partition F_2 is valid.

All partitions are invalid It can also happen that no valid partition exists at all. In this case, $\mathcal{G}_{32}$ cannot be applied to e. An example for this case is $z_1 = z_3 = 0.8$ and $z_2 = z_4 = -0.8$. Here, F_1 is invalid because of the inverted cell $(\mathbf{b}, \mathbf{v}_2, \mathbf{v}_4, \mathbf{v}_3)$, F_2 is invalid because of the inverted cell $(\mathbf{a}, \mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_4)$.

These examples show that we need another condition that ensures that F is a valid partition. Under the assumption that Conditions 1 and 2 are satisfied, we found four equivalent formulations 3.2., 3.1., 3.3., and 3.4. for the missing condition. Will prove their equivalence later in Theorem 11. Before we describe the missing condition in detail we need to define the *supporting plane* $\mathrm{pl}(t)$ of a triangle t as the affine hull of its vertices.

▶ Condition 3.

3.1. All cells $\mathbf{c}'_{a,j}$ and $\mathbf{c}'_{b,j}$ have positive orientation.

3.2. Every $\mathbf{f}_i$ has $\mathbf{a}$ on its positive side, and $\mathbf{b}$ on its negative side.

3.3. The spherical projection of the faces $\mathbf{f}_i$ onto $S^{\mathbf{a}}$ is contained in $S^{\mathbf{a}}_{\mathrm{I}} \cup \mathbf{B}^{\mathbf{a}}$, and the interior of the inner edges is projected into $S^{\mathbf{a}}_{\mathrm{I}}$ (for $S^{\mathbf{b}}$ analogously),

$$\Pi^{\mathbf{p}}(\mathbf{f}_i) \subset S^{\mathbf{p}}_{\mathrm{I}} \cup \mathbf{B}^{\mathbf{p}}, \qquad \text{for all } i = 1, \dots, n, \qquad \text{for } \mathbf{p} \in \{\mathbf{a}, \mathbf{b}\}. \tag{2}$$

$$\Pi^{\mathbf{p}}(\mathring{\boldsymbol{d}}) \subset S^{\mathbf{p}}_{\mathrm{I}}, \qquad \text{for all } d \in I, \tag{3}$$

3.4. The interior of the inner edges is a subset of the interior of the affected region, and the supporting planes of all faces f_i intersects the line $\mathbf{l}$ in the interior of $\mathbf{e}$,

$$\mathring{\boldsymbol{d}} \subset \mathbf{C} \setminus \partial\mathbf{C}, \qquad \text{for all } d \in I, \tag{4}$$

$$pl(\mathbf{f}_i) \cap \mathbf{l} \in \mathring{\mathbf{e}}. \tag{5}$$

▶ **Theorem 6.** *If Conditions 1, 2, and 3 are met, the triangulation $\mathcal{T}' = (\mathcal{V}, \mathcal{E}', \mathcal{F}', \mathcal{C}')$ with $\mathcal{C}' = (\mathcal{C} \setminus C) \cup C'$ (and $\mathcal{E}'$ and $\mathcal{F}'$ accordingly) is valid.*

Proof. Due to Conditions 1 and 3.1., all cells of $\mathcal{C}'$ have positive orientation. To prove that there are no holes in C', we check for border faces of the cells of C':

- The faces $b_i \cup \{p\}$ for $p \in \{a, b\}$ are border faces of both C and C'.
- The faces f_j are incident to $c'_{a,j}$ and $c'_{b,j}$, i.e., f_j is not on the border of C'.
- For the faces $f = d \cup \{p\}, d \in I, p \in \{a, b\}$, the edge d is incident to two f aces f_j and f_k, i.e., f is incident to $c'_{p,j}$ and $c'_{p,k}$. So, f is not on the border of C'.

Thus, there are no new border faces, i.e., there are no holes in C'. ◀

▶ **Lemma 7.** *Condition 3.1. and Condition 3.2. are equivalent.*

Proof. By definition, $\mathbf{a}$ is on the positive side of $\mathbf{f}_i$ if and only if the cell $\mathbf{c}'_{a,i}$ has positive orientation. Furthermore, $\mathbf{b}$ is on the negative side of $\mathbf{f}_i$ if and only if the cell $\mathbf{c}'_{b,i}$ has positive orientation. ◀

▶ **Lemma 8.** *Conditions 3.1. and 3.2. imply Condition 3.3.*

Proof. To prove (2) we first show that $\Pi^a(\mathbf{F})$ is a connected region on $S^{\mathbf{a}}$ that is bounded by $\mathbf{B}^{\mathbf{a}}$. Then we show that $t^{\mathbf{a}}_{\mathrm{I}} \in \Pi^{\mathbf{a}}(\mathbf{F})$.

Due to Condition 3.1. $\mathbf{a}$ is not in $\mathbf{F}$, since this would cause degenerate cells, and $\Pi^{\mathbf{a}}(\mathbf{F})$ is a connected region on $S^{\mathbf{a}}$. For $f_i, f_j \in F$ with common edge $d \in I$, the spherical triangles $\Pi^{\mathbf{a}}(\mathbf{f}_i)$ and $\Pi^{\mathbf{a}}(\mathbf{f}_j)$ share the spherical edge $\Pi^{\mathbf{a}}(\boldsymbol{d})$. Due to Condition 3.1. the both cells $\mathbf{c}'_{a,i}$ and $\mathbf{c}'_{a,j}$ have positive orientation, so they are on opposite sides of the plane P through $\boldsymbol{d}$ and $\mathbf{a}$. Therefore, $\Pi^{\mathbf{a}}(\mathbf{f}_i)$ and $\Pi^{\mathbf{a}}(\mathbf{f}_j)$ are also on opposite sides of $\Pi^{\mathbf{a}}(\boldsymbol{d})$, see Figure 6. This implies that the interior of all inner edges of I is not projected to the boundary of $\Pi^{\mathbf{a}}(\mathbf{F})$. The same holds true for all interior points of $\mathbf{F}$. Thus, the boundary of $\Pi^{\mathbf{a}}(\mathbf{F})$ consists of projections of the border edges of B. Consequently, the interior of $\Pi^{\mathbf{a}}(\mathbf{F})$ is not intersected by $\mathbf{B}^{\mathbf{a}}$, so $\Pi^{\mathbf{a}}(\mathbf{F})$ is either completely in $S^{\mathbf{a}}_{\mathrm{I}} \cup \mathbf{B}^{\mathbf{a}}$, or in $S^{\mathbf{a}}_{\mathrm{O}} \cup \mathbf{B}^{\mathbf{a}}$.

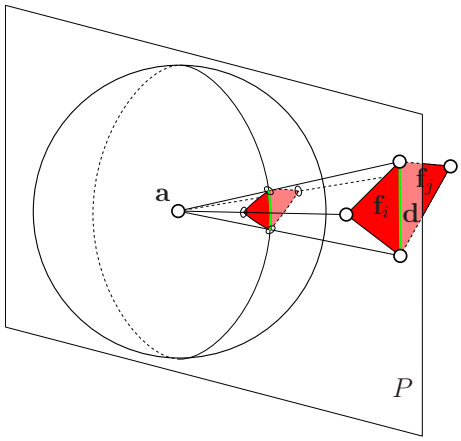


Figure 6 The projections of $\mathbf{f}_i$ and $\mathbf{f}_j$ are on opposite sides of the projection of the edge $\mathbf{d}$.

Since $\mathbf{B}$ winds around $\mathbf{l}$ once, the $\mathbf{l}$ line intersects $\mathbf{F}$ in at least one face $\mathbf{f}_i$. Let $\mathbf{p} = \mathbf{l} \cap \mathbf{f}_i$. Because $\mathbf{a}$ is on the positive side of $\mathbf{f}_i$ (Condition 3.2.), $\Pi^{\mathbf{a}}(\mathbf{p}) = t_{\mathrm{I}}^{\mathbf{a}}$. Therefore, $\Pi^{\mathbf{a}}(\mathbf{F}) \subset S_{\mathrm{I}}^{\mathbf{a}} \cup \mathbf{B}^{\mathbf{a}}$. Analogously, one can show $\Pi^{\mathbf{b}}(\mathbf{F}) \subset S_{\mathrm{I}}^{\mathbf{b}} \cup \mathbf{B}^{\mathbf{b}}$.

Especially, the interior of $\Pi^{\mathbf{a}}(\mathring{\boldsymbol{d}})$ does not intersect $\mathbf{B}^{\mathbf{a}}$, which implies (3). ◀

▶ **Lemma 9.** *Condition 3.3. implies Condition 3.4.*

Proof. Let $d \in I$ be an inner edge of F, and $\mathbf{p} \in \mathring{\boldsymbol{d}}$ be an interior point of $\boldsymbol{d}$. Due to Condition 3.3., $\mathbf{p}^{\mathbf{a}} = \Pi^{\mathbf{a}}(\mathbf{p}) \in S_{\mathrm{I}}^{\mathbf{a}}$. We split $S_{\mathrm{I}}^{\mathbf{a}}$ into spherical triangles by adding edges from $\Pi^{\mathbf{a}}(\mathbf{v}_i)$ to $t_{\mathrm{I}}^{\mathbf{a}}$. At least one of these triangles contains $\mathbf{p}^{\mathbf{a}}$. Let this triangle be $t = (t_{\mathrm{I}}^{\mathbf{a}}, \Pi^{\mathbf{a}}(\mathbf{v}_l), \Pi^{\mathbf{a}}(\mathbf{v}_{l+1}))$, see Figure 7. The boundary $\mathbf{B}^{\mathbf{a}}$ (green) is partitioned into spherical triangles (red lines), $\boldsymbol{d}$ is $(\mathbf{v}_1, \mathbf{v}_4)$ (blue line) and $\mathbf{p} \in \mathring{\boldsymbol{d}}$. In this case, $\Pi^{\mathbf{a}}(\mathbf{p})$ is within the spherical triangle $t = (t_{\mathrm{I}}^{\mathbf{a}}, \Pi^{\mathbf{a}}(\mathbf{v}_4), \Pi^{\mathbf{a}}(\mathbf{v}_5))$.

The set of points that are projected into t is defined as the intersection of the half spaces defined by the planes spanned by $\mathbf{a}$ and one of the edges of t, i.e., $g_1 = (a, b, v_l)$, $g_2 = (a, v_{l+1}, b)$, and $g_3 = (a, v_l, v_{l+1})$, which contains the fourth point $\{a, b, v_l, v_{l+1}\} \setminus g_i$. The point $\mathbf{p}$ cannot be on the negative side of $\mathbf{g}_1$, $\mathbf{g}_2$ or $\mathbf{g}_3$, as this would mean that its image is not in t. Also, it cannot be in the plane defined by $\mathbf{a}$ and $\mathbf{g}_3$, as this would mean that it is projected to $\mathbf{B}^{\mathbf{a}}$.

With the same argument for $\Pi^{\mathbf{b}}$, we obtain the faces $g_4 = (b, v_l, a)$, $g_5 = (b, v_{l+1}, a)$, and $g_6 = (b, v_{l+1}, v_l)$. Removing the redundant faces $g_4 \equiv g_1$ and $g_5 \equiv g_2$, we can conclude that $\mathbf{p}$ is not on the negative side of $\mathbf{g}_1$ and $\mathbf{g}_2$, and it is on the positive side of $\mathbf{g}_3$ and $\mathbf{g}_6$. These four faces define the cell c_i. Thus, $\mathbf{p} \in \mathbf{C}$, and $\mathbf{p} \notin \partial\mathbf{C}$, proving (4).

We still have to prove (5). Assume there exists a face f in F with $\{\mathbf{q}\} = \mathrm{pl}(\mathbf{f}) \cap \mathbf{l} \notin \mathring{\mathbf{e}}$ and, without loss of generality, $\lambda \leq 0$. This face has at least one interior edge $d \in I$ and we chose an arbitrary point $\mathbf{p} \in \mathring{\mathbf{d}}$. Now, $\mathbf{p}$ is projected to $\mathbf{p}^{\mathbf{a}}$ which lies outside of $S_{\mathrm{I}}^{\mathbf{a}}$. This contradicts (3) and, thus, proves (5). ◀

▶ **Lemma 10.** *If Conditions 1 and 2 are satisfied, Condition 3.4. implies Condition 3.2.*

Proof. For $n = 3$ we have $I = \emptyset$ and $F = \{f_1\}$. Since B circles around $\mathbf{l}$, there must be an intersection of $\mathbf{l}$ and $\mathbf{f}_1$. Due to Condition 3.4., this is between $\mathbf{a}$ and $\mathbf{b}$, and because of the order of the vertices of f_1 as induced by Lemma 2, $\mathbf{a}$ is on the positive and $\mathbf{b}$ on the negative side of $\mathbf{f}_1$, and Condition 3.2. is satisfied.

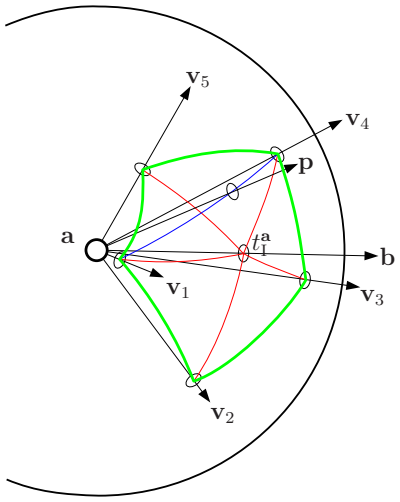


Figure 7 $S_{\mathrm{I}}^{\mathbf{a}}$ is divided into spherical triangles (red lines), one of which contains $\Pi^{\mathbf{a}}(\mathbf{p})$.

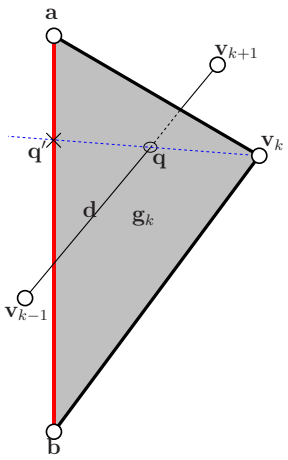


Figure 8 The intersection of the extension of segment $\mathbf{v}_k$ to $\mathbf{q}$ with $\mathbf{l}$ is between $\mathbf{a}$ and $\mathbf{b}$.

We now consider $n > 3$. The partition F contains $n-2$ faces, the border B has n edges (see Lemma 3). If every face of F had at most one edge of B, there would be at least two edges in B left. Since no face of F can have three edges of B (otherwise B would have a sub-cycle of three edges), at least two faces of F must have two edges of B. Let $\hat{F} \subset F$ be the set of faces with two edges in B.

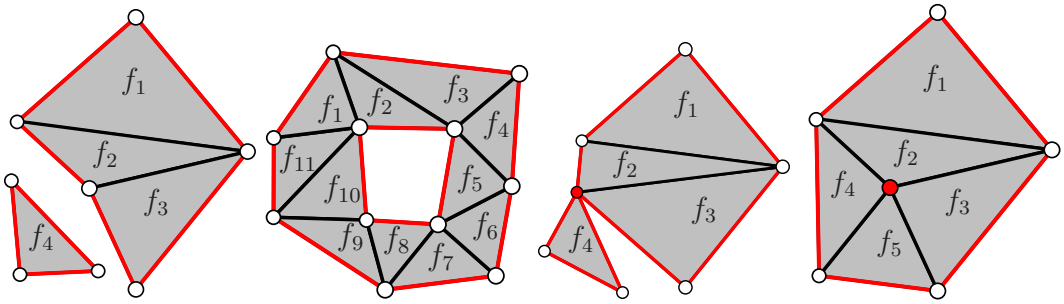
The line $\mathbf{l}$ intersects either one face of F in its interior, or it intersects an inner edge of I and therefore two faces of F on their border.

In the case that $\mathbf{l}$ intersects an inner edge, and the adjacent faces of F are the only two faces in $\hat{F}$, there can be no other faces in F, due to the following: if two faces with each two edges in B and both sharing a common inner edge, their edges in B already define a cycle. Since B does not contain any sub-cycles, there can be no further edges in B. In this case we have $n = 4$. Since the intersection of $\mathbf{l}$ with $\mathbf{f}_1$ and $\mathbf{f}_2$ is between $\mathbf{a}$ and $\mathbf{b}$ (Condition 3.4.), and because of the order of the vertices of f_1 and f_2, $\mathbf{a}$ is on the positive side of $\mathbf{f}_1$ and $\mathbf{f}_2$, and $\mathbf{b}$ is on the negative side. Thus, in this case Condition 3.2. is satisfied.

For the remaining case there is at least one face f in $\hat{F}$ that has no intersection with $\mathbf{l}$, because otherwise Conditions 1 and 2 were violated. Let $f = (v_{k-1}, v_k, v_{k+1})$. Because f does not intersect $\mathbf{l}$, $\theta_{k-1} + \theta_k < \pi$. Thus, the inner edge $d = (v_{k-1}, v_{k+1})$ cannot cross any other cell besides c_{k-1} and c_k. Due to Condition 3.4., $\mathring{\boldsymbol{d}} \subset c_{k-1} \cup c_k$. With $g_k = (a, b, v_k)$, the intersection $\boldsymbol{d} \cap \mathbf{g}_k = \{\mathbf{q}\}$, with $\mathbf{q}$ in $\mathring{\mathbf{g}}_k$. When extending the line segment from $\mathbf{v}_k$ to $\mathbf{q}$, it intersects the segment $\mathbf{e}$ in its interior in point $\mathbf{q}'$, because of (5) (see Figure 8). Since $\mathbf{v}_k$ and $\mathbf{q}$ are points in $\mathbf{f}$, the line through $\mathbf{v}_k$ and $\mathbf{q}$ is also in the plane of $\mathbf{f}$, and so is $\mathbf{q}'$. From these considerations and the vertex order of f, it follows that $\mathbf{a}$ is on the positive and $\mathbf{b}$ is on the negative side of $\mathbf{f}$. Thus, $\mathbf{f}$ fulfills Condition 3.2..

Now we remove f from F, i.e., F becomes $F \setminus \{f\}$, B becomes $(B \setminus \{b_{k-1}, b_k)\}) \cup \{d\}$, and I becomes $I \setminus \{d\}$. This new edge cycle B still satisfies Conditions 1 and 2, but has one edge less. This procedure can be repeated until $n = 3$, or $n = 4$ and $\mathbf{l}$ intersects both faces in F. ◀

▶ **Theorem 11.** *If Conditions 1 and 2 are satisfied, Conditions 3.2., 3.1., 3.3., and 3.4. are equivalent.*



(a) Violating Condition 5. **(b)** Violating Condition 5. **(c)** Violating Condition 5. **(d)** Violating Condition 6.

Figure 9 Examples of sets F violating Conditions 5 or 6.

Proof. This follows directly from Lemmata 7, 8, 9, and 10. ◀

5 Geometric Conditions for $\mathcal{G}_{23}$

We use the same notation as in Section 4, i.e., $F = \{f_1, \dots, f_m\}$ is a set of faces sandwiched between $\mathbf{a}$ and $\mathbf{b}$, such that $\mathbf{F}$ is a connected 2-manifold. The edge set $B = \{b_1, \dots, b_n\}$ are the border edges of F. The order of edges in B is induced by the order of boundary edges in $\partial\mathbf{F}$.

The triangulation before and after the $\mathcal{G}_{23}$-swap is denoted by $\mathcal{T}'$ and $\mathcal{T}$. We define the orientation of $\mathbf{f}_i$ so that $\mathbf{a}$ is on the positive side of $\mathbf{f}_i$. The cells incident to these faces are $C' = \{c'_{a,1}, c'_{b,1}, \dots, c'_{a,m}, c'_{b,m}\}$ with $c'_{p,i} = f_i \cup \{p\}$ for $i = 1, \dots, m$ and $p \in \{a, b\}$. The new edge in $\mathcal{T}$ is $e = (a, b)$.

Next we define the conditions for which $\mathcal{G}_{23}$ will result in a valid triangulation.

▶ Condition 4. The triangulation $\mathcal{T}' = (\mathcal{V}, \mathcal{E}', \mathcal{F}', \mathcal{C}')$ is valid, and all cells of $\mathcal{T}'$ have positive orientation.

▶ Condition 5. The edges of B form exactly one simple cycle $(v_1, \dots, v_n)$.

This condition ensures that the faces in F are connected via edges, that there is only one connected component of faces, and that the faces form a bounded 2-manifold without holes. Examples of sets F that violating Condition 5 are shown in Figures 9a, 9b, and 9c.

▶ Condition 6. All vertices incident to a face in F are on the border B.

Condition 6 the absence of interior vertices in F that are not part of B. Those interior vertices would be removed by $\mathcal{G}_{23}$, but a swap may only modify the connectivity, but not add, remove, or move vertices. Figure 9d shows an example of a set F that violates Condition 6 due to an interior vertex.

▶ **Lemma 12.** *If Condition 6 is satisfied, the number of vertices in B is $n = m + 2$.*

Proof. If Condition 6 is satisfied, F is a partition of B. Considering Lemma 5 we can conclude $m = n - 2$. Therefore, $n = m + 2$. ◀

The $\mathcal{G}_{23}$-swap will now replace the cells C' by the cells $C = \{c_1, \dots, c_n\}$ with

$$c_i = (a, b, v_i, v_{i+1})$$

and faces $g_i = (a, b, v_i)$, where the index i is understood modulo n.

▶ Condition 7. One of the equivalent following conditions holds:

7.1. All cells $\mathbf{c}_i$ have positive orientation.

7.2. The dihedral angle θ_i between the faces $\mathbf{g}_i$ and $\mathbf{g}_{i+1}$ (in counterclockwise direction, seen from a in direction b) is in $(0, \pi)$.

▶ **Lemma 13.** *Condition 7.1. and Condition 7.2. are equivalent.*

Proof. $\mathbf{c}_i$ has positive orientation if and only if $\mathbf{c}_i$ is consistently oriented or non-degenerate. This is true if and only if the inner dihedral angle θ_i is in $(0, \pi)$. ◀

▶ **Theorem 14.** *If Conditions 4, 5, 6, and 7 are met, the triangulation $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{C})$ with $\mathcal{C} = (\mathcal{C} \setminus C') \cup C$ (and $\mathcal{E}$ and $\mathcal{F}$ accordingly) is valid.*

Proof. Due to Conditions 4 and 7.1., all cells of $\mathcal{C}$ have positive orientation. To prove that there are no holes in C, we check for border faces of the cells of C:

- The faces $b_i \cup \{p\}$ for $p \in \{a, b\}$ are border faces of both C' and C.
- The faces g_i are incident to c_{i-1} and c_i, i.e., g_i is not on the border of C.

Thus, there are no new border faces, i.e., there are no holes in C. ◀

6 Replacing Generalized Swaps by a Series of Basic Swaps

In [23] Shewchuk showed that the "multi-face removal" (equivalent to $\mathcal{G}_{23}$) and "edge removal" (equivalent to $\mathcal{G}_{32}$) can be replaced by a series of basic 2-3 and 3-2 swaps. The intermediate triangulations are topologically correct, but may contain inconsistently oriented or degenerate tetrahedra.

We will show that there always exists a series of basic 2-3, 3-2, and 4-4 swaps to mimic the effect of a $\mathcal{G}_{23}$- and a $\mathcal{G}_{32}$-swap, where all intermediate triangulations are valid. This result shows that the $\mathcal{G}_{23}$- and $\mathcal{G}_{32}$-swaps do not add additional potential that is not already possible with 2-3, 3-2 and 4-4 swaps. An optimization procedure like simulated annealing should theoretically be able to find a near-optimal solution also without utilizing $\mathcal{G}_{23}$ and $\mathcal{G}_{32}$. In practice, the convergence rate can be increased by implementing $\mathcal{G}_{23}$ and $\mathcal{G}_{32}$.

6.1 Replacing $\mathcal{G}_{32}$

Let $\mathbf{e}$ be an inner edge of triangulation T, B the set of border edges, and F a valid partition of B, so that the Conditions 1, 2, and 3 for $\mathcal{G}_{32}$ are satisfied.

▶ **Theorem 15.** *The same effect as the $\mathcal{G}_{32}$ swap operation of $\mathbf{e}$ and partition F can be obtained by a series of either*

- *$n - 3$ basic 2-3 swaps followed by a 3-2 swap, or*
- *$n - 4$ basic 2-3 swaps followed by a 4-4 swap, for $n \geq 4$.*

Proof. We use the same arguments as in the proof of Lemma 10.

For $n = 3$, F consists of exactly one face f_1, and the vertices of f_1 circle around $\mathbf{e}$ exactly once. Therefore, the conditions are satisfied to apply a 3-2 swap to $\mathbf{e}$, so we can substitute $\mathcal{G}_{32}$ by a single 3-2 swap.

For $n = 4$, and the single inner edge $d = (v_i, v_{i+2})$ with $i \in \{1, 2\}$ intersects with $\mathbf{e}$, the quadrilateral $(\mathbf{v}_i, \mathbf{a}, \mathbf{v}_{i+2}, \mathbf{b})$ is planar and convex, fulfilling the conditions of a 4-4 swap. This 4-4 swap replaces e by d and the four cells of C with the four cells of C'. Thus, the $\mathcal{G}_{32}$ swap can be replaced by a single 4-4 swap.

If $n = 4$ and d and e do not intersect, or if $n > 4$, there is at least one face in F with two edges in B that does not intersect $\mathbf{e}$. Let this face be $f_j = (v_{i-1}, v_i, v_{i+1})$. As in

the proof for Lemma 10, the edge $d = (v_{i-1}, v_{i+1})$ intersects the face g_i in its interior, so the cells c_{i-1} and c_i fulfill the condition for a 2-3 swap. This swap removes c_{i-1} and c_i from the triangulation, adds $c'_{a,j}$ and $c'_{b,j}$ and a temporary new cell $c = (a, b, v_{i-1}, v_{i+1})$. The remaining cells $(C \setminus \{c_{i-1}, c_i\}) \cup \{c\}$ together with the reduced partition $F \setminus \{f_j\}$ and the reduced border $(B \setminus \{(v_{i-1}, v_i), (v_i, v_{i+1})\}) \cup \{d\}$ fulfill Conditions 1–3. So, $\mathcal{G}_{32}$ can be applied to the reduced setting. By induction, the reduced setting can be processed with either $(n-1)-3$ 2-3 swaps, followed by a 3-2 swap, or with $(n-1)-4$ 2-3 swaps, followed by a 4-4 swap. Adding the 2-3 swap to remove f_j, the claim follows. ◀

6.2 Replacing $\mathcal{G}_{23}$

Since the $\mathcal{G}_{23}$ operation is the inverse of the $\mathcal{G}_{32}$ operation for the same partition F, $\mathcal{G}_{23}$ can be replaced by a series of basic swaps.

▶ **Theorem 16.** *The same effect as a $\mathcal{G}_{23}$ operation of a partition F sandwiched between a and b can be obtained by a series of either*

- *a single 2-3 swap, followed by $n-3$ 3-2 swaps, or*
- *a single 4-4 swap, followed by $n-4$ 3-2 swaps.*

Proof. While $\mathcal{G}_{23}$ replaces the cells C' by cells C, $\mathcal{G}_{32}$ does the inverse. $\mathcal{G}_{32}$ can be substituted by a series of basic swap operations $\mathcal{G}_{32} = s_1 \circ s_2 \circ \cdots \circ s_m$, with m being either $n-2$ (s_m being a 3-2 swap) or $n-3$ (s_m being a 4-4 swap), as in Theorem 15. For the same choice of F, we have

$$\mathcal{G}_{23} = {\mathcal{G}_{32}}^{-1} = (s_1 \circ \cdots \circ s_m)^{-1} = s_m^{-1} \circ \cdots \circ s_1^{-1}.$$

The inverse of a 3-2 swap is a 2-3 swap and vice versa, and the inverse of a 4-4 swap is a corresponding 4-4 swap. We start in $\mathcal{G}_{23}$ with s_m^{-1}, which is either a 2-3 swap or a 4-4 swap. Then we proceed with either $n-3$ or $n-4$ 3-2 swaps. ◀

7 Conclusions

We have presented different geometric conditions for generalized swap operations a 3D triangulation. These conditions are proved to be equivalent, such that one can use that particular condition in practice that is most appropriate given the specific needs of an implementation. In a mesh optimization application these swap operations are used to speed up the optimization process and to attenuate "getting stuck" in local minima.

Furthermore, we have shown that the generalized swap operations can be realized by simple 3-2, 2-3, and 4-4 swaps, which simplifies the implementation significantly. This decomposition of the generalized swap guarantees at the same time, that all intermediate triangulations are consistently oriented and do not contain degenerate cells, causing numerical problems in certain applications.

Based on these conditions, our future research plans are focused on applications of 3D mesh optimizations, e.g., in video compressions or bio-medical and bio-mechanical simulations.

Acknowledgments

This work was partly funded by the German Research Foundation (DFG) within the International Research Training Group 1131, “Visualization of Large and Unstructured Data Sets” at the University of Kaiserslautern. This work was also supported by the National Science Foundation under contract ACI 9624034 (CAREER Award) and a large Information Technology Research (ITR) grant.

References

1 N. Amenta, S. Choi, and R. K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2–3):127–153, 2001.

2 M. de Berg, M. van Kreveld, M. Overmary, and O. Schwarzkopf. *Computational Geometry*. Springer, 2nd edition, 2000.

3 H. L. de Cougny and M. S. Shephard. Parallel refinement and coarsening of tetrahedral meshes. *Int. J. Numer. Meth. Engng.*, 46:1101–1125, 1999.

4 L. Demaret, N. Dyn, and A. Iske. Image compression by linear splines over adaptive triangulations. *Signal Process.*, 86(7):1604–1616, 2006.

5 L. Demaret, N. Dyn, A. Iske, and M. Floater. Adaptive thinning for terrain modelling and image compression. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 321–340. Springer, 2004.

6 Klaus Denker, Burkhard Lehner, and Georg Umlauf. Online triangulation of laser-scan data. In R. Garimella, editor, *Proceedings of the 17th International Meshing Roundtable 2008*, pages 415–432, 2008.

7 N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolations. *IMA Journal of Numerical Analysis*, 10(1):137–154, Jan 1990.

8 M. Garland and P. Heckbert. Fast polygonal approximation of terrains and height fields. Technical report, CS Department, Carnegie Mellon University, 1995.

9 M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97*, pages 209–216, 1997.

10 M. Garland and Y. Zhou. Quadric-based simplification in any dimension. *ACM Trans. Graph.*, 24(2):209–239, 2005.

11 H. Hoppe. Progressive meshes. In *SIGGRAPH '96*, pages 99–108, 1996.

12 H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stüetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

13 B. Joe. Construction of three-dimensional improved-quality triangulations using local transformations. *SIAM J. Sci. Comp.*, 16(6):1292–1307, 1995.

14 B. M. Klingner and J. R. Shewchuk. Aggressive tetrahedral mesh improvement. In *16th Int. Meshing Roundtable*, pages 3–23, 2007.

15 O. Kreylos and B. Hamann. On simulated annealing and the construction of linear spline approximations for scattered data. *IEEE TVCG*, 7(1):17–31, 2001.

16 C. L. Lawson. Properties of n-dimensional triangulations. *CAGD*, 3:231–246, 1986.

17 B. Lehner. *Meshing Techniques for Image/Video Compression and Surface Reconstruction*. PhD thesis, TU Kaiserslautern, Germany, 2008.

18 B. Lehner, G. Umlauf, and B. Hamann. Image compression using data-dependent triangulations. In G. Bebis, editor, *ISVC 2007*, pages 351–362, 2007.

19 B. Lehner, G. Umlauf, and B. Hamann. Video compression using data-dependent triangulations. In *Computer Graphics and Visualization '08*, pages 244–248, 2008.

20 H. Pedrini. An improved refinement and decimation method for adaptive terrain surface approximation. In *Proceedings of WSCG*, pages 103–109, 2001.

21 V. Petrovic and F. Küster. Optimized construction of linear approximations to image data. In *Proc. 11th Pacific Conf. on Comp. Graphics and Appl.*, pages 487–491, 2003.

22 L. L. Schumaker. Computing optimal triangulations using simulated annealing. *CAGD*, 10(3-4):329–345, 1993.

23 J. R. Shewchuk. Two discrete optimization algorithms for the topological improvement of tetrahedral meshes, 2002. Unpublished manuscript, http://www.cs.berkeley.edu/∼jrs/papers/edge.pdf.

24 J. R. Shewchuk. What is a good linear element? Interpolation, conditioning, and quality measures. In *11th Int. Meshing Roundtable*, pages 115–126, 2002.

25 X. Yu, B. S. Morse, and T. W. Sederberg. Image reconstruction using data-dependent triangulation. *IEEE Comput. Graph. Appl.*, 21(3):62–68, 2001.

On Curved Simplicial Elements and Best Quadratic Spline Approximation for Hierarchical Data Representation

Bernd Hamann[1]

1 **Institute for Data Analysis and Visualization (IDAV)**
Department of Computer Science
University of California, Davis
Davis, CA 95616
hamann@cs.ucdavis.edu

Abstract

We present a method for hierarchical data approximation using curved quadratic simplicial elements for domain decomposition. Scientific data defined over two- or three-dimensional domains typically contain boundaries and discontinuities that are to be preserved and approximated well for data analysis and visualization. Curved simplicial elements make possible a better representation of curved geometry, domain boundaries, and discontinuities than simplicial elements with non-curved edges and faces. We use quadratic basis functions and compute best quadratic simplicial spline approximations that are C^0-continuous everywhere except where field discontinuities occur whose locations we assume to be given. We adaptively refine a simplicial approximation by identifying and bisecting simplicial elements with largest errors. It is possible to store multiple approximation levels of increasing quality. Our method can be used for hierarchical data processing and visualization.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Approximation, Bisection, Grid Generation, Finite Elements, Hierarchical Approximation, Simplicial Decomposition, Spline

Digital Object Identifier 10.4230/DFU.SciViz.2010.45

1 Introduction

Scalar and vector field data often contain discontinuities that should be preserved for data approximation and analysis purposes. It is important to represent domain boundaries—including geometry such as a car body, an aircraft, or a ship hull—and the locations of field discontinuities, represented by curves and surfaces. To better approximate these curves and surfaces we investigate the use of curved quadratic simplicial elements. We do not address the problem of extracting discontinuities from a given scalar or vector field data set; we assume that this information is known. We consider data defined over two-dimensional (2D) and three-dimensional (3D) domains.

We utilize only curved simplicial elements that are quadratic. In the 2D case, we use curved triangles whose edges may be straight line segments or parabolae; in the 3D case, we use curved tetrahedral elements whose edges/faces may be straight line segments/planar triangles or curved. Generally, we refer to both non-curved and curved simplicial elements as just simplicial elements. We use a quadratic polynomial transformation to map the so-called *standard simplex* to the corresponding simplicial region in 2D/3D space. Furthermore, we use a quadratic polynomial defined over each simplicial element to locally approximate the dependent

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 45–61

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

variable(s). We use curved elements with curved edges/faces to better approximate domain boundaries and discontinuities. All simplicial elements that do not "touch" geometry, domain boundaries, or discontinuities are non-deformed elements. Nevertheless, the polynomials we use over all simplicial elements are all quadratic.

Our overall goal is the construction of a hierarchical data over 2D or 3D domains using a best approximation approach based on curved quadratic finite elements and quadratic polynomials defined over these elements. We start with a coarse decomposition of the domain, using a relatively small number of simplicial elements and placing curved simplices in areas where boundaries and discontinuities occur. We then compute a (globally) best least squares approximation, a quadratic spline approximation for the dependent variable(s) that is C^0-continuous. (Due to the C^0 continuity requirement we can place simplices with curved edges/faces only along boundaries and where discontinuities occur, i.e., in areas where the curved edges/faces are not shared by other elements. The physical locations of discontinuities play the same roles as domain boundaries: Two simplicial elements may share the—geometrically—same edge/face defining the locus of a discontinuity, but the field function defined over the two elements is discontinuous along/on the shared edge/face.) Based on local errors that we compute for each simplicial element, we bisect a certain percentage of the elements with largest errors, update the simplicial domain decomposition accordingly, and compute a new best quadratic spline approximation. We iterate this process until a specified error condition is met or the number of simplicial elements exceeds some threshold.

Our approach belongs to the class of *refinement* methods. These methods are based on the principle of refining intermediate data approximations by inserting additional points or elements until a certain termination criterion is satisfied. We have developed our method with a focus on the needs of massive scientific data analysis and visualization, see [22, 39, 45]. To enable interactive frame rates for massive data visualization, for example, it is possible to use low-resolution best approximations everywhere or adaptively insert high-resolution approximations locally into an otherwise relatively coarse approximation. The overall approximation algorithm is based on these steps:

- **Initial simplicial domain decomposition.** Assuming that either a polygonal/polyhedral or an analytical definition is known for all boundaries and discontinuities in the 2D/3D domain of interest, we construct a coarse simplicial decomposition of this domain. We use curved edges/faces only in areas where they are needed to better approximate curved boundaries and/or discontinuities. (The quadratic transformations, mapping the standard simplex defined in so-called *parameter space* to deformed simplices in so-called *physical space*, are defined by specifying corresponding point pairs in the two spaces such that one obtains a one-to-one, bijective mapping.) Figure 1 shows a possible initial simplicial decomposition, including curved elements, of space around a wing cross section.
- **Best approximation.** In the 2D case, each simplicial element has six associated *knots*, one knot per corner and one knot per edge. Six knots in parameter space are associated with six points in physical space, and this defines the needed quadratic mapping for a simplex. (Accordingly, the number of knots is ten in the 3D case.) For simplicity, we consider only knots that are uniformly distributed along the edges of the standard simplex, see Appendix A. We associate a quadratic polynomial with each simplicial element, which approximates the dependent variable(s) over the corresponding region in space. We represent each quadratic basis polynomial in so-called *Bernstein-Bézier form*, see [12, 41]. Assuming that the function to be approximated, a scalar- or vector-valued function, is known in analytical form, it is possible to compute the unique best quadratic spline approximation defined as a linear combination of the set of quadratic basis functions.

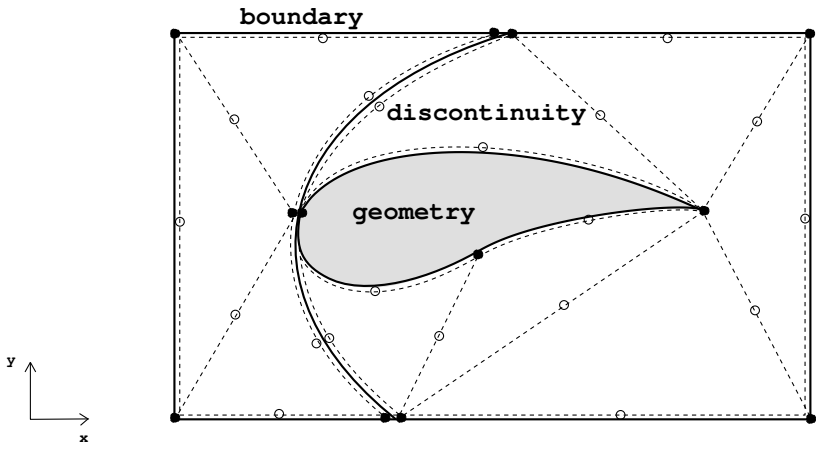


Figure 1 Decomposition of space around wing using curved 2D simplices (geometry, domain boundary, and discontinuity shown in bold).

The best approximation, understood in a least squares sense, is the result of solving the *normal equations*, see [9].

- **Adaptive bisection.** We compute a local error value for each simplicial element once a best approximation is known. We use the L_2 norm to compute simplex-specific error values. The set of simplices is ordered according to the simplex-specific, local error values. To compute a next-level best quadratic approximation we determine a certain percentage of simplices with largest error values and bisect them by splitting them at the mid point of their longest edge. If a simplex' longest edge is not unique, we choose the edge to be split randomly. In the case of curved edges we use arc length to determine the longest edge to be bisected. Splitting a specific simplex into two simplices induces additional splits for all those simplices that share the split edge. We update a simplicial domain decomposition by considering all edge bisections and compute a new best approximation. We repeat the process of identifying simplices with largest errors, bisecting these simplices, and computing a new best approximation until we obtain an approximation for which the maximal simplex-specific error is below a certain error threshold or until a maximal number of simplices is reached.
- **Hierarchical data representation.** To support hierarchical data processing and visualization, for example, we can store multiple best approximations of different simplicial resolutions. For each best approximation, we need to store the polynomial coefficients of each simplicial element—for its shape and the polynomial defined over it. Considering a non-curved simplicial element, we only need to store its three (four) corner points and the coefficients of the quadratic polynomial defined over the element. Considering a curved element, we need to store all polynomial coefficients defining the shape of the element in addition to the coefficients of the quadratic polynomial defined over the element. We store a fixed number of best approximations such that either the number of simplices increases in a specified fashion or the maximal simplex-specific error decreases in a certain way from one resolution to the next.

We discuss these steps in more detail in the sections to follow.

Related work in the areas of hierarchical data representation and approximation is discussed in [1, 4, 5, 6, 7, 11, 16, 17, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 38, 48,

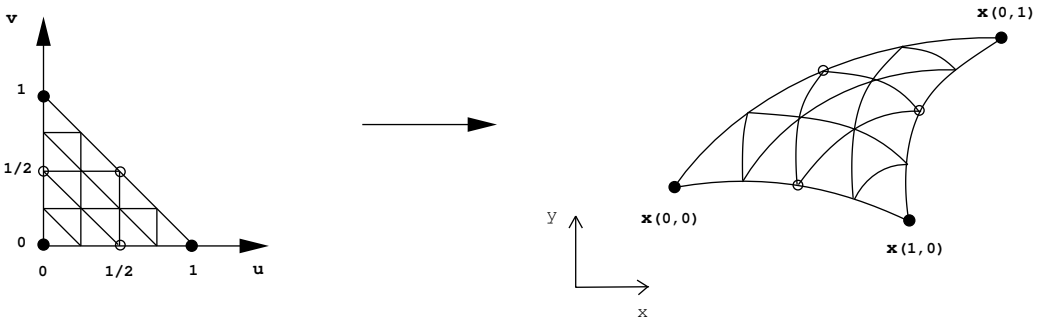


Figure 2 Mapping standard triangle to arbitrary curved triangle (iso-parametric lines shown in parameter and physical space).

51, 52, 59]. Best-approximation methods are described in [53, 55], and effective processing and visualization approaches for data approximated by higher-order elements are covered in [19, 55, 56, 57]. So-called *data-dependent triangulation* schemes, i.e., schemes concerned with the construction of piecewise linear approximations using near-optimal simplicial elements, are described in [10, 36, 43]. In [46, 47] various data structures are covered in-depth that can be used for efficient storage of hierarchical data approximations. From a broader perspective, our work is related to *grid generation*, and references for this area are [14, 32, 49, 50]. Finite element methods, which also are closely related to our work, are discussed in detail in [60].

2 Mapping the Standard Simplex

In the 2D case, the standard simplex in parameter space is the triangle with corners $(0,0)$, $(1,0)$, and $(0,1)$. The triangle with these three corners is mapped to a curved triangular region in physical space by mapping the six knots $\mathbf{u_i} = (u_{i,j}, v_{i,j}) = \left(\frac{i}{2}, \frac{j}{2}\right)$, $i,j \geq 0$, $i+j \leq 2$ (abbreviated in multi-index notation as $|\mathbf{i}| = 2$), in parameter space to six corresponding points $\mathbf{x_i} = (x_{i,j}, y_{i,j})$ in physical space, using a quadratic mapping. The quadratic mapping in the 2D case, using Bernstein-Bézier polynomials $B^2_{\mathbf{i}}(\mathbf{u})$ as basis functions, see [12, 41] and Appendix A, is given by

$$\mathbf{x}(\mathbf{u}) \;=\; \begin{pmatrix} x(u,v) \\ y(u,v) \end{pmatrix} \;=\; \sum_{|\mathbf{i}|=2} \mathbf{b_i}\; B^2_{\mathbf{i}}(\mathbf{u}) \;=\; \begin{pmatrix} \sum\limits_{|\mathbf{i}|=2} c_{i,j}\; B^2_{i,j}(u,v) \\ \sum\limits_{|\mathbf{i}|=2} d_{i,j}\; B^2_{i,j}(u,v) \end{pmatrix}. \tag{1}$$

The mapping between parameter and physical space must be one-to-one. Figure 2 depicts the general mapping of the standard triangle in parameter space to a curved triangle in physical space.

In the same way, we define the mapping of the standard tetrahedron with corners $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$ to a curved tetrahedron in physical space, mapping the ten knots $\mathbf{u_i} = (u_{i,j,k}, v_{i,j,k}, w_{i,j,k}) = \left(\frac{i}{2}, \frac{j}{2}, \frac{k}{2}\right)$, $|\mathbf{i}| = 2$, to ten corresponding points $\mathbf{x_i} = (x_{i,j,k}, y_{i,j,k}, z_{i,j,k})$ in physical space. Thus, the quadratic mapping in the 3D case is

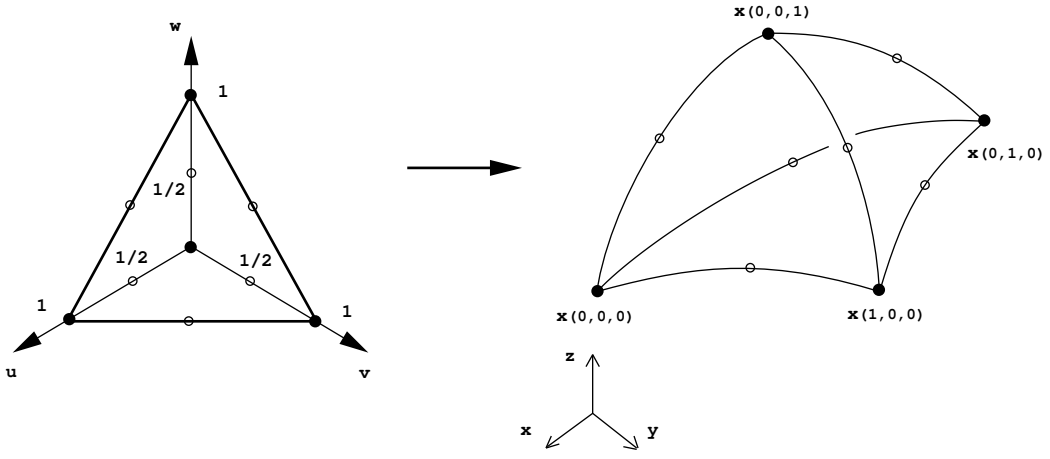


Figure 3 Mapping standard tetrahedron to arbitrary curved tetrahedron.

given by

$$\mathbf{x}(\mathbf{u}) \;=\; \begin{pmatrix} x(u,v,w) \\ y(u,v,w) \\ z(u,v,w) \end{pmatrix} \;=\; \sum_{|\mathbf{i}|=2} \mathbf{b_i}\; B^2_{\mathbf{i}}(\mathbf{u}) \;=\; \begin{pmatrix} \sum\limits_{|\mathbf{i}|=2} c_{i,j,k}\; B^2_{i,j,k}(u,v,w) \\ \sum\limits_{|\mathbf{i}|=2} d_{i,j,k}\; B^2_{i,j,k}(u,v,w) \\ \sum\limits_{|\mathbf{i}|=2} e_{i,j,k}\; B^2_{i,j,k}(u,v,w) \end{pmatrix}. \qquad (2)$$

Figure 3 depicts the general mapping of the standard tetrahedron to a curved tetrahedron.

We use quadratic Bernstein-Bézier polynomials as basis functions for the approximation of a field function defined over non-curved simplicial elements as well. We denote these quadratic basis functions as $B^2_{\mathbf{i}}(\mathbf{x}) = B^2_{i,j}(x,y)$ $(= B^2_{i,j,k}(x,y,z)$ in the 3D case). A generalization of the standard Bernstein-Bézier polynomials is necessary for curved simplicial elements. We define the needed generalized quadratic basis polynomials for curved elements in Appendix A. Figure 4 shows the graph of a quadratic polynomial defined over its associated curved triangular domain.

3 Initial Simplicial Domain Decomposition

The main objective driving the development of our method is the hierarchical representation of very large scientific data sets enabling real-time and adaptive data processing and visualization. Data sets resulting from computational simulations are typically defined on a grid, and the dependent variables are associated with either the vertices, also called *nodes* in the finite element literature, or the elements defining the grid. We assume that a data set is provided on a high-resolution grid. The original grid, its boundaries, and possibly known locations of field discontinuities (in the dependent variables) influence how we define an initial simplicial decomposition of the relevant 2D/3D domain.

The objective is to initially represent the 2D/3D domain with a relatively small number of curved simplicial elements, using curved elements only where they help to better approximate domain boundaries and known field discontinuities. In the 2D case, the grid points discretizing the domain boundary represent curves, while they represent surfaces in the 3D setting. For practical purposes we proceed as follows: First, we compute the bounding box of the original set of grid points and decompose this box into two (five) non-curved triangles (tetrahedra).

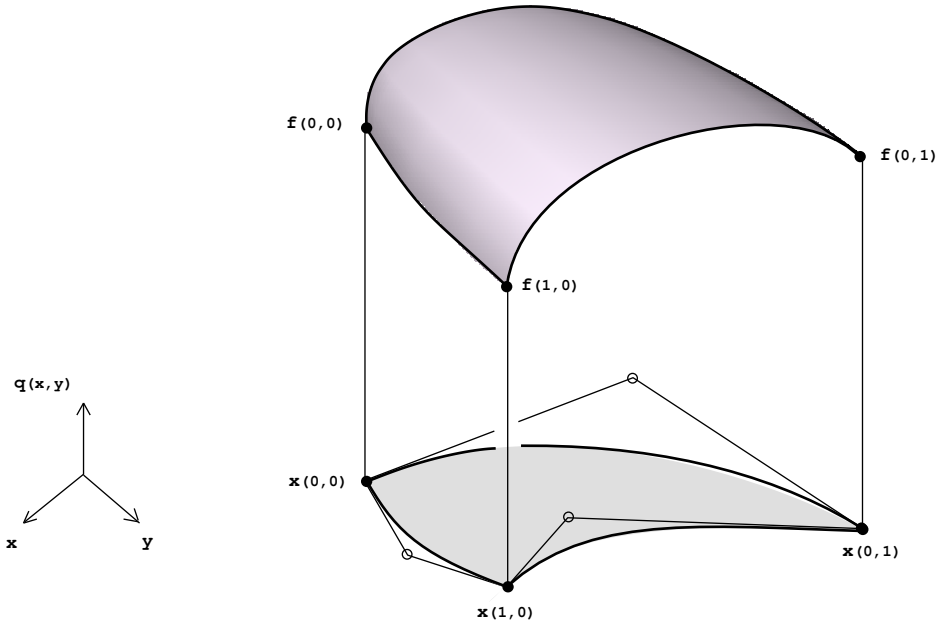


Figure 4 Graph of quadratic polynomial over its curved simplicial domain (Bernstein-Bézier control net shown for curved domain simplex).

Second, we clip these non-curved simplices against the curves (surfaces) defining the domain boundaries. Third, we identify the portion of the initial two (five) simplices that lies inside the domain over which the dependent variable(s) must be approximated; we represent this portion by using initially non-curved simplices only.

We consider perpendicular distance values to determine the quality of a simplicial domain decomposition. We compute the distances of the original grid boundary points from the boundary edges (faces) of the initially non-curved (boundary) simplices. If these distance values are larger than a certain threshold, we must solve a local optimization problem, i.e., we deform an edge/face of a non-curved simplex in a quadratic fashion such that the original grid points in the affected areas are (nearly) optimally approximated by quadratic curves (surfaces). We can solve this problem locally as a univariate (bivariate) approximation problem by considering the distances of original grid points in normal direction of the associated edges/faces of the simplicial boundary elements. We note that the construction of a globally optimal boundary curve/surface approximation is a subject in its own right, but it is not the focus of this paper. We continue our discussion by assuming that boundary approximation schemes suitable for incorporation into our overall scheme are available.

Geometry and domain boundaries can be identified easily from an originally supplied grid. One simply needs to identify elements with edges/faces that are not shared by other elements. It is much harder to identify the locations in 2D/3D space where field discontinuities, i.e., discontinuities of the functions describing the dependent variables, occur. Such discontinuities should be preserved in a simplicial approximation as much as possible. Such discontinuities, once there locations are known, can be treated by curved simplicial elements just like domain boundaries.

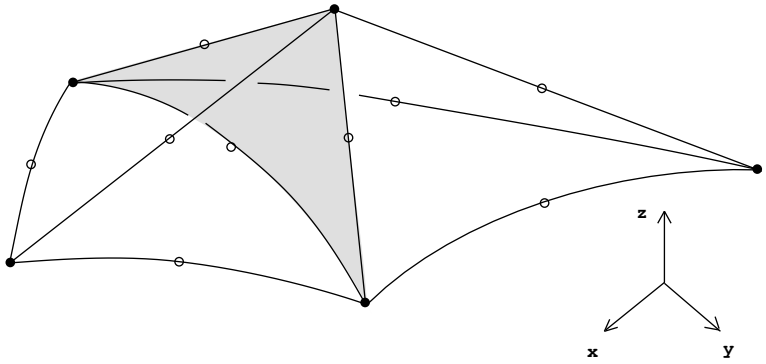


Figure 5 Shared face of two simplicial elements in 3D space is planar but may have curved edges.

The detection of discontinuities of scalar-valued functions over 2D/3D domains has been an active research area in several disciplines, including digital image analysis, pattern recognition/feature extraction from satellite imagery, and scientific data visualization. We refer to the methods described in [2, 40] for more detail. For our purposes, we assume that discontinuities can effectively be extracted from a given data set and that curves/surfaces are used to represent them in the 2D/3D domain. Topologically, we treat these curves (surfaces) in the same way as we treat boundary curves (surfaces) by using simplices with curved edges/faces where they touch these discontinuities. Thus, every discontinuity is approached from two sides, and the simplicial elements touching a discontinuity do not share vertices. (The geometrical information of vertices of these elements *is* shared by vertices along/on field discontinuities, but the coefficients used for field function approximation are different.) Once an initial decomposition of the domain is constructed, we compute the implied best quadratic spline approximation, which we describe in the next section.

In the 3D case, we ensure that each face that is shared by two simplices is planar. Nevertheless, certain edges of a shared planar face may be curved whenever these edges belong to a simplex face that approximates the domain boundary geometry (surface) or a discontinuity in the 3D domain. This situation is illustrated in Figure 5.

4 Best Approximation

We assume that the field/function to be approximated over a 2D/3D domain is known analytically. Should this not be the case, e.g., in the case of *scattered data* (randomly distributed points with associated function values but without connectivity information), it is always possible to construct an analytical representation by performing a prior data interpolation or approximation step, see [13, 37]. In the case that a data set is defined on a grid, the required analytical definition is given by a piecewise linear function for a simplicial (triangular, tetrahedral) grid and a piecewise bilinear/trilinear function in the case of quadrilateral/hexahedral grid cells. (We assume that function values are associated with grid vertices.) We denote the analytical function to be approximated over the domain by $F(\mathbf{x})$. Based on an initial simplicial domain decomposition, we compute the corresponding best piecewise quadratic approximation of $F(\mathbf{x})$ by solving the normal equations, see [9]. The normal equations determine the set of coefficients for the desired quadratic spline representation, a best approximation in the least squares sense.

Corner vertices of simplicial elements may be shared by any number of simplices, and we

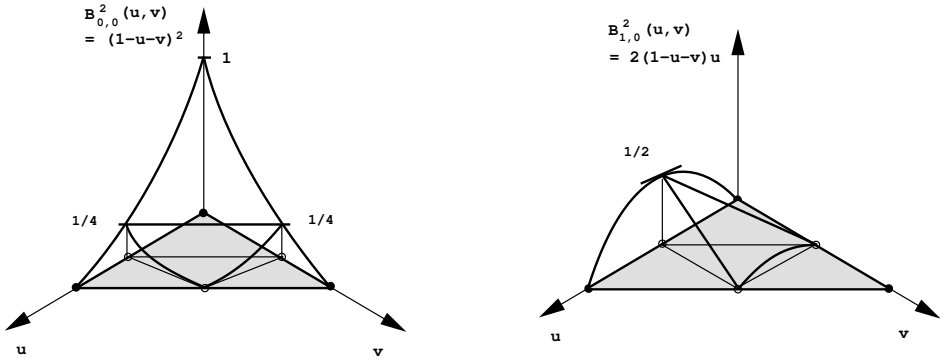


Figure 6 Types of basis functions. Basis function associated with shared corner (left) and shared edge (right).

denote the basis function that we associate with a corner vertex $\mathbf{v}_i$ by $f_i(\mathbf{x})$. An edge of a simplicial element may be shared by no more than two simplices in the 2D case and by an arbitrary number of simplices in the 3D case. We denote a basis function that we associate with a simplex edge e_j by $g_j(\mathbf{x})$. We refer to the set of simplices sharing a common corner vertex as the *platelet* of this corner, and we call the set of simplices sharing a common edge *edge neighbors*. Thus, a set of platelet simplices defines the region in space over which a basis function associated with the corresponding corner vertex is non-zero. Edge neighbors, associated with a particular edge, define the region in space over which a basis function associated with this edge is non-zero. Instead of providing a formal definition for these two types of basis functions, we refer to Figure 6 that depicts the types for the bivariate case.

We denote a best approximation as $a(\mathbf{x})$, and we write it as a linear combination of the basis functions associated with all distinct simplex corners and simplex edges. Assuming that there are m distinct corners and n distinct edges, we can write a best approximation as

$$a(\mathbf{x}) \;=\; \sum_{i=1}^{m} c_i \; f_i(\mathbf{x}) + \sum_{j=1}^{n} d_j \; g_j(\mathbf{x}). \tag{3}$$

We must solve the normal equations to obtain the unknown coefficients c_i and d_j. In matrix form, the normal equations are

$$\begin{pmatrix} \langle f_1,f_1\rangle & \dots & \langle f_1,f_m\rangle & \langle f_1,g_1\rangle & \dots & \langle f_1,g_n\rangle \\ \vdots & & & & & \vdots \\ \langle f_m,f_1\rangle & \dots & \langle f_m,f_m\rangle & \langle f_m,g_1\rangle & \dots & \langle f_m,g_n\rangle \\ \langle g_1,f_1\rangle & \dots & \langle g_1,f_m\rangle & \langle g_1,g_1\rangle & \dots & \langle g_1,g_n\rangle \\ \vdots & & & & & \vdots \\ \langle g_n,f_1\rangle & \dots & \langle g_n,f_m\rangle & \langle g_n,g_1\rangle & \dots & \langle g_n,g_n\rangle \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_m \\ d_1 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} \langle F,f_1\rangle \\ \vdots \\ \langle F,f_m\rangle \\ \langle F,g_1\rangle \\ \vdots \\ \langle F,g_n\rangle \end{pmatrix}, \tag{4}$$

where $\langle G,H\rangle$ denotes the inner product of the two functions G and H, i.e.,

$$\langle G,H\rangle \;=\; \int_{\text{common domain of } G \text{ and } H} G(\mathbf{x})\; H(\mathbf{x})\; d\mathbf{x}. \tag{5}$$

We must compute inner products involving curved and non-curved simplices. Since all simplicial elements in physical space are defined as quadratic (or linear) mappings of the

standard simplex, we can simplify integration by making use of the *change-of-variables theorem*, see [34], which relates integration in physical space to integration in parameter space for parametrically defined regions. In the 2D case, integrals are computed according to the formula

$$\int\limits_{\text{curved simplex}} G(x,y)\ dx\ dy \ = \int\limits_{\text{standard simplex}} G\Big(x(u,v),y(u,v)\Big)\ J(u,v)\ du\ dv, \tag{6}$$

where $J(u,v)$ denotes the *Jacobian* associated with the mapping of the standard simplex to the corresponding simplex in physical space. The Jacobian is the determinant

$$J(u,v) \ = \ |\mathbf{x_u}| \ = \ \begin{vmatrix} x_u & x_v \\ y_u & y_v \end{vmatrix} \ = \ \begin{vmatrix} \frac{\partial}{\partial u}x(u,v) & \frac{\partial}{\partial v}x(u,v) \\ \frac{\partial}{\partial u}y(u,v) & \frac{\partial}{\partial v}y(u,v) \end{vmatrix}. \tag{7}$$

(The 3D case is a straightforward extension.) When using a simple linear transformation to map the parameter space knots $\mathbf{u_i}$ to associated physical space points $\mathbf{x_i}$, the Jacobian has a constant value C. This constant value is given by the determinant

$$C \ = \ \begin{vmatrix} \mathbf{d}_{2,0} & \mathbf{d}_{0,2} \end{vmatrix} \tag{8}$$

in the 2D case and

$$C \ = \ \begin{vmatrix} \mathbf{d}_{2,0,0} & \mathbf{d}_{0,2,0} & \mathbf{d}_{0,0,2} \end{vmatrix} \tag{9}$$

in the 3D case, where the column vectors $\mathbf{d_i}$ are given by $\mathbf{d_i} = \mathbf{x_i} - \mathbf{x_0}$.

The matrices resulting for the best approximation problems for different levels of simplicial resolution are sparse, and several methods exist for bandwidth reduction, efficient factorization, and inversion of such sparse matrices, see [8, 15, 18, 42, 44]. Matrix bandwidth is related to the indexing scheme used for the set of basis functions, i.e., the indexing used for simplex corners and simplex edges. We apply a bandwidth reduction step prior to matrix factorization/inversion.

The computation of the inner products appearing in the normal equations requires multi-dimensional integration over non-curved and curved simplicial elements. While the change-of-variables theorem reduces this integration to integration over the standard simplex, we still need to perform numerical integration for the calculation of the inner products appearing on the right-hand side of the normal equations, i.e., integrals of the types $\langle F, f_i \rangle$ and $\langle F, g_j \rangle$, since $F(\mathbf{x})$ can be any integrable function. We use *Romberg integration* for the computation of these right-hand-side inner products, see [3, 27]. Appendix B lists some of the needed inner product values for quadratic Bernstein-Bézier polynomials.

Once we have computed a best approximation for a particular simplicial domain decomposition, we analyze the local approximation quality to identify those simplices that should be refined (bisected) to further improve approximation quality. In the following section, we discuss the general principles used for adaptive bisection.

5 Adaptive Bisection

For each simplicial element in a particular domain decomposition, we compute a local approximation error. We define this error as

$$E(S_i) \ = \int\limits_{\text{curved simplex } S_i} \Big(F(\mathbf{x}) - a(\mathbf{x})\Big)^2\ d\mathbf{x}. \tag{10}$$

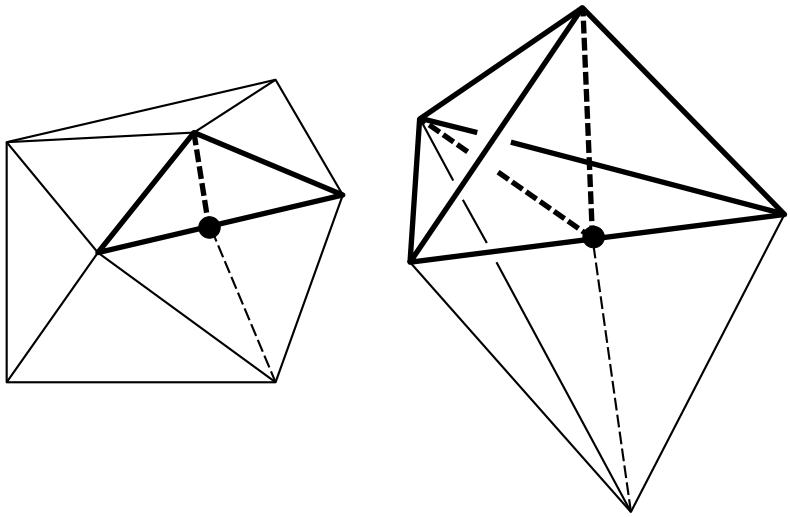


Figure 7 Bisection of simplices in bivariate and trivariate cases.

We order the set $\{S_i\}$ of simplicial elements in decreasing order of their associated error values $E(S_i)$. To construct a new, refined best approximation we specify a percentage of simplices to be bisected and choose the simplices with largest approximation errors.

We bisect a simplicial element marked for refinement by identifying an edge of maximal length, using arc length in the case of curved edges, and split this element by using the midpoint of the split edge as a new simplicial corner vertex. The bisection step is shown in Figure 7. All simplices sharing the split edge are bisected as well to avoid so-called hanging nodes and thus preserve a *conforming mesh*. The bisection steps lead to a new simplicial domain decomposition, and we must compute a new best quadratic spline approximation.

We continue to bisect a certain percentage of simplices in the resulting intermediate simplicial domain decompositions until either the number of simplices in a decomposition exceeds some threshold or an approximation is obtained whose maximal simplex-specific error value is smaller than some tolerance. In principle, it is possible to store all intermediate best quadratic spline approximations in addition to the originally supplied data, possibly including a grid. For practical purposes, this might not always be possible due to storage limitations. Therefore, the number of different best approximations that one stores usually depends on the original resolution of a given data set and its underlying grid, the "complexity" of a given analytical field function, the amount of storage available, and the criterion used to terminate adaptive bisection. The final result of our method is a set of independent best quadratic spline approximations to be used for the purposes of real-time, adaptive, or hierarchical analysis and visualization.

6 Data Visualization Issues

Our data approximation method based on curved simplicial elements must also be considered in the context of visualization techniques applied to data sets defined over 2D and 3D domains. In the case of scalar-valued data sets, the particularly relevant visualization approaches to be considered are (i) extraction and visualization of isocurves/isosurfaces or *contours*; (ii)

slicing the data domain with lines/planes (*slicing lines/planes*); and (iii) *ray casting*, see [19, 55, 56]. Applying these types of visualization techniques to curved simplicial elements over which the dependent variable varies in a quadratic fashion requires us to generalize standard visualization methods that often can deal only with elements with planar faces and linearly or trilinearly varying dependent scalar value.

It is reasonable to view an approximation consisting of curved quadratic simplicial elements to be competitive with a representation consisting of only non-curved, linear simplicial elements when the higher-degree polynomial representation can be rendered nearly as efficiently as the linear one. To study the competitiveness of the piecewise quadratic approximation scheme one must compare rendering efficiency and simplicity for two types of approximation: a piecewise quadratic approximation based on a combination of non-curved and curved simplices and a piecewise linear approximation based on only non-curved simplices. In order to compare two approximation schemes properly, one must require that their respective overall approximation errors are nearly the same.

The application of slicing methods, contouring techniques, and ray casting to non-curved quadratic elements is done routinely. As discussed in [58], for example, the intersection of a ray with an isosurface inside a non-curved 3D simplicial quadratic element, for example, reduces to solving a univariate quadratic equation. The *volume rendering integral* along a ray segment, see [35], is generally too complex to be integrated in closed form, and it is therefore computed numerically. A cut plane intersects a non-curved simplicial element in a polygon on which the scalar field is a quadratic function. Quadratic texture coordinates can be computed in software, or in hardware by taking advantage of texture look-up tables.

Visualization of curved simplicial elements is much more difficult. A quadratic mapping from parameter to physical space must be inverted prior to evaluating a scalar field function at a point in physical space. In the case of curved tetrahedral elements, this requires one to solve three quadratic equations in three variables simultaneously, which can be done with numerical techniques. One could require that a tetrahedral face shared by two tetrahedra is planar, and thus it would be possible to define the field function directly in terms of physical space. The construction of the necessary basis functions for this case is described in Appendix A. Similar problems arise when intersecting a ray with an isosurface or a curved simplex face. We intend to investigate in the future how to render curved simplices directly by solving the involved algebraic equations most elegantly and most efficiently.

A simple solution is to subdivide a curved simplex adaptively, depending on a view, and approximate a curved simplex by non-curved simplices resulting from a properly chosen subdivision scheme. One can replace edge endpoints with the respective edge midpoints. A reasonable criterion to use when deciding when to terminate the subdivision process could be based on the image-space projected maximal deviation of the (union of) the non-curved simplices from their curved "parent" simplex, to allude to just one possibility. Since we represent curved simplicial elements in Bernstein-Bézier form, one could also apply subdivision techniques used in computer-aided geometric design, see [12]. An algorithm like the one described in in [58] could then be applied to the set of non-curved simplices, having quadratic variation only in scalar value.

7 Conclusions

We have described a method for the construction of hierarchical approximations of functions over 2D and 3D domains. The method uses curved simplicial elements to represent the finite domain of a function to be approximated and constructs a best piecewise quadratic

approximation in the least squares sense. Curved simplicial elements are promising in the context of approximating complicated 2D or 3D domains and the dependent functions defined over these domains. Such higher-order elements allow one to construct approximations with relatively smaller error when compared to lower-order and non-curved elements.

8 Acknowledgments

This work was supported by the National Science Foundation under contracts ACI 9624034 (CAREER Award), through a Large Scientific and Software Data Set Visualization (LSSDSV) grant under contract ACI 9982251, and an Information Technology Research (ITR) sub-award. We thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV)) at the University of California, Davis.

References

1 Bajaj, C. L, Pascucci, V. and Zhuang, G. (1999), Progressive compression and transmission of arbitrary triangular meshes, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 307–316.

2 Ballard, D. H. and Brown, C. M. (1982), *Computer Vision*, Prentice-Hall, Inc., Englewood Cliffs, NJ.

3 Boehm, W. and Prautzsch, H. (1993), *Numerical Methods*, A K Peters, Ltd., Wellesley, MA.

4 Bonneau, G. P., Hahmann, S. and Nielson, G. M. (1996), BLaC-wavelets: A multiresolution analysis with non-nested spaces, in: Yagel, R. and Nielson, G. M., eds., *Visualization '96*, IEEE Computer Society Press, Los Alamitos, CA, pp. 43–48.

5 Bonneau, G. P. (1999), Optimal triangular Haar bases for spherical data, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 279–284.

6 Cignoni, P., De Floriani, L., Montani, C., Puppo, E. and Scopigno, R. (1994), Multiresolution modeling and visualization of volume data based on simplicial complexes, in: Kaufman, A. E. and Krüger, W., eds., *1994 Symposium on Volume Visualization*, IEEE Computer Society Press, Los Alamitos, CA, pp. 19–26.

7 Cohen-Or, D., Levin, D. and Remez, O. (1999), Progressive compression of arbitrary triangular meshes, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 67–72.

8 Cuthill, E. and McKee, J. (1969), Reducing the bandwidth of sparse symmetric matrices, in: *Proceedings of the ACM National Conference*, Association for Computing Machinery, New York, NY, pp. 157–172.

9 Davis, P. J. (1975), *Interpolation and Approximation*, Dover Publications, Inc., New York, NY.

10 Dyn, N., Levin, D., and Rippa, S. (1990), Algorithms for the construction of data dependent triangulations, in: Mason, J. C. and Cox, M. G., eds., *Algorithms for Approximation II*, Chapman and Hall, New York, NY, pp. 185–192.

11 Eck, M., DeRose, A. D., Duchamp, T., Hoppe, H., Lounsbery, M. and Stuetzle, W. (1995), Multiresolution analysis of arbitrary meshes, in: Cook, R., ed., *Proceedings of SIGGRAPH 1995*, ACM Press, New York, NY, pp. 173–182.

12 Farin, G. (2001), *Curves and Surfaces for CAGD: A Practical Guide*, fifth edition, Morgan Kaufmann Publishers, Inc., San Francisco, CA.

13 Franke, R. (1982), Scattered data interpolation: Tests of some methods, Math. Comp. 38, pp. 181–200.

14 George, P. L. (1991), *Automatic Mesh Generation*, Wiley & Sons, New York, NY.

15 Gibbs, N. E., Poole, W. G. and Stockmeyer P. K. (1976), An algorithm for reducing the bandwidth and profile of a sparse matrix, SIAM J. Numer. Anal. 13(2), pp. 236–250.

16 Gieng, T. S., Hamann, B., Joy, K. I., Schussman, G. L. and Trotts, I. J. (1997), Smooth hierarchical surface triangulations, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 379–386.

17 Gieng, T. S., Hamann, B., Joy, K. I., Schussman, G. L. and Trotts, I. J. (1998), Constructing hierarchies for triangle meshes, IEEE Transactions on Visualization and Computer Graphics 4(2), pp. 145–161.

18 Golub, G. H. and Van Loan, C. F. (1989), *Matrix Computations*, second edition, Johns Hopkins University Press, Baltimore, MD.

19 Gregorski, B. F., Wiley, D. F, Childs, H. R., Hamann, B. and Joy, K. I. (2006), Adaptive contouring with quadratic tetrahedra, in: Bonneau, G.-P., Ertl, T. and Nielson, G. M., eds., *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Springer-Verlag, Heidelberg, Germany, pp. 3–15.

20 Gross, M. H., Gatti, R. and Staadt, O. (1995), Fast multiresolution surface meshing, in: Nielson, G. M. and Silver, D. eds., *Visualization '95*, IEEE Computer Society Press, Los Alamitos, CA, pp. 135–142.

21 Grosso, R., Lürig, C. and Ertl, T. (1997), The multilevel finite element method for adaptive mesh optimization and visualization of volume data, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 387–394.

22 Hagen, H., Müller, H. and Nielson, G. M., eds. (1993), *Focus on Scientific Visualization*, Springer-Verlag, New York, NY.

23 Hamann, B. (1994), A data reduction scheme for triangulated surfaces, Computer Aided Geometric Design 11(2), pp. 197–214.

24 Hamann, B. and Chen, J. L. (1994a), Data point selection for piecewise linear curve approximation, Computer Aided Geometric Design 11(3), pp. 289–301.

25 Hamann, B. and Chen, J. L. (1994b), Data point selection for piecewise trilinear approximation, Computer Aided Geometric Design 11(5), pp. 477–489.

26 Hamann, B. and Jordan, B. W. (1998), Triangulations from repeated bisection, in: Dæhlen, M., Lyche, T. and Schumaker, L. L., eds., *Mathematical Methods for Curves and Surfaces II*, Vanderbilt University Press, Nashville, TN, pp. 229–236.

27 Hamann, B., Jordan, B. W. and Wiley, D. A. (1999), On a construction of a hierarchy of best linear spline approximations using repeated bisection, IEEE Transactions on Visualization and Computer Graphics 5(1/2), pp. 30–46, p. 190 (errata).

28 Heckel, B., Weber, G. H., Hamann, B. and Joy, K. I. (1999), Construction of vector field hierarchies, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 19–25.

29 Hoppe, H. (1996), Progressive meshes, in: Rushmeier, H., ed., *Proceedings of SIGGRAPH 1996*, ACM Press, New York, NY, pp. 99–108.

30 Hoppe, H. (1997), View-dependent refinement of progressive meshes, in: Whitted, T., ed., *Proceedings of SIGGRAPH 1997*, ACM Press, New York, NY, pp. 189–198.

31 Hoppe, H. (1999), New quadric metric for simplifying meshes with appearance attributes, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 59–66.

32 Knupp, P. M. and Steinberg, S. (1993), *Fundamentals of Grid Generation*, CRC Press, Boca Raton, FL.

33 Kreylos, O. and Hamann, B. (1999), On simulated annealing and the construction of linear spline approximations for scattered data, in: Gröller, E., Löffelmann, H. and Ribarsky, W., eds., *Data Visualization '99*, Springer-Verlag, Vienna, Austria, pp. 189–198.

34 Marsden, J. E. and Tromba, A. J. (1988), *Vector Calculus*, third edition, W. H. Freeman and Company, New York, NY.

35 Max, N. L. (1995), Optical models for direct volume rendering, IEEE Transactions on Visualization and Computer Graphics 1(2), pp. 99–108.

36 Nadler, E. (1986), Piecewise linear best L_2 approximation on triangulations, in: Ward, J. D., ed., *Approximation Theory V*, Academic Press, Inc., San Diego, CA, pp. 499–502.

37 Nielson, G. M. (1993), Scattered data modeling, IEEE Computer Graphics and Applications 13(1), pp. 60–70.

38 Nielson, G. M., Jung, I.-H. and Sung, J. (1997a), Haar wavelets over triangular domains with applications to multiresolution models for flow over a sphere, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 143–149.

39 Nielson, G. M., Müller, H. and Hagen, H., eds.(1997b), *Scientific Visualization: Overviews, Methodologies, and Techniques*, IEEE Computer Society Press, Los Alamitos, CA.

40 Pavlidis, T. (1980), *Structural Pattern Recognition*, second printing, Springer-Verlag, New York, NY.

41 Piegl, L. A. and Tiller, W. (1996), *The NURBS Book*, second edition, Springer-Verlag, New York, NY.

42 Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992), *Numerical Recipes in C*, second edition, Cambridge University Press, New York, NY.

43 Rippa, S. (1992), Long and thin triangles can be good for linear interpolation, SIAM J. Numer. Anal. 29(1), pp. 257–270.

44 Rosen, R. (1968), Matrix bandwidth minimization, in: *Proceedings of the ACM National Conference*, ACM publication no. P-68, Brandon Systems Press, Princeton, NJ, pp. 585–595.

45 Rosenblum, L. J., Earnshaw, R. A., Encarnação, J. L., Hagen, H., Kaufman, A. E., Klimenko, S., Nielson, G. M., Post, F. and Thalmann, D., eds. (1994), *Scientific Visualization—Advances and Challenges*, IEEE Computer Society Press, Los Alamitos, CA.

46 Samet, H. (1990), *The Design and Analysis of Spatial Data Structures*, Addison Wesley, New York, NY.

47 Samet, H. (2006), *Foundations of Multidimensional and Metric Data Structures*, Elsevier B. V., Amsterdam, The Netherlands.

48 Staadt, O. G., Gross, M. H. and Weber, R. (1997), Multiresolution compression and reconstruction, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 337–346.

49 Thompson, J. F., Warsi, Z. U. A. and Mastin, C. W. (1985), *Numerical Grid Generation*, North-Holland, New York, NY.

50 Thompson, J. F., Soni, B. K. and Weatherill, N. P., eds. (1999), *Handbook of Grid Generation*, CRC Press, Boca Raton, FL.

51 Trotts, I. J., Hamann, B. and Joy, K. I. (1999), Simplification of tetrahedral meshes with error bounds, IEEE Transactions on Visualization and Computer Graphics 5(3), pp. 224–237.

52 Trotts, I. J., Hamann, B., Joy, K. I. and Wiley, D. F. (1998), Simplification of tetrahedral meshes, in: Ebert, D. S., Hagen, H. and Rushmeier, H. E., eds., *Visualization '98*, IEEE Computer Society Press, Los Alamitos, California, pp. 287–295.

53 Wiley, D. F., Bertram, M. and Hamann, B. (2004), On a construction of a hierarchy of best linear spline approximations using a finite element approach, IEEE Transactions on Visualization and Computer Graphics 10(5), pp. 548–563.

54 Wiley, D. F., Bertram, M., Jordan, B. W., Hamann, B., Joy, K. I., Max, N. L. and Scheuermann, G. (2003), Hierarchical spline approximation, in: Farin, G., Hamann, B. and

Hagen, H., eds., *Hierarchical and Geometrical Methods in Scientific Visualization*, Springer-Verlag, Heidelberg, Germany, pp. 63–88.

55 Wiley, D. F., Childs, H. R., Gregorski, B. F., Hamann, B. and Joy, K. I. (2003), Contouring curved quadratic elements, in: Bonneau, G.-P., Hahmann, S. and Hansen, C. D., eds., *Data Visualisation 2003*, Eurographics Association, Aire-la-Ville, Switzerland, pp. 167–176.

56 Wiley, D. F., Childs, H. R., Hamann, B. and Joy, K. I. (2004), Ray casting curved-quadratic elements, in: Deussen, O., Hansen, C. D., Keim, D. A. and Saupe, D., eds., *Data Visualization 2004*, Eurographics Association, Aire-la-Ville, Switzerland, pp. 201–209.

57 Wiley, D. F., Childs, H. R., Hamann, B., Joy, K. I. and Max, N. L. (2002), Best quadratic spline approximation for hierarchical visualization, in: Ebert, D. S., Brunet, P. and Navazo, I., eds., *Data Visualisation 2002*, Eurographics Association, Aire-la-Ville, Switzerland, pp. 133–140.

58 Williams, P. L., Max, N. L. and Stein, C. M. (1998), A high accuracy volume renderer for unstructured data, IEEE Transactions on Visualization and Computer Graphics 4(1), pp. 37–54.

59 Xia, J. C. and Varshney, A. (1996), Dynamic view-dependent simplification for polygonal meshes, in: Yagel, R. and Nielson, G. M., eds., *Visualization '96*, IEEE Computer Society Press, Los Alamitos, CA, pp. 327–334.

60 Zienkiewicz, O. C. and Taylor, R. L. (2006), *The Finite Element Method*, sixth edition, Elsevier B. V., Amsterdam, The Netherlands.

A Quadratic Basis Polynomials

Our method requires quadratic basis polynomials for non-curved and curved simplicial elements. We review the definition of the standard Bernstein-Bézier polynomials used for non-curved elements before generalizing these polynomials for curved elements.

The quadratic Bernstein-Bézier polynomial basis functions, defined for the standard simplex in parameter space, are

$$B^2_{\mathbf{i}}(\mathbf{u}) \;=\; \frac{2!}{(2-i-j)!\;i!\;j!}\;(1-u-v)^{2-i-j}\;u^i\;v^j \tag{11}$$

in the bivariate case and

$$B^2_{\mathbf{i}}(\mathbf{u}) \;=\; \frac{2!}{(2-i-j-k)!\;i!\;j!\;k!}\;(1-u-v-w)^{2-i-j-k}\;u^i\;v^j\;w^k \tag{12}$$

in the trivariate case. Through a linear parameter transformation we can evaluate these quadratic polynomials over all non-curved simplices in physical space. Figure 8 illustrates the graphs of two quadratic Bernstein-Bézier basis functions in the 2D case over the standard triangle.

We must define quadratic basis polynomials for field function approximation over curved simplices in a different way. These more general polynomials are not the result of applying a simple linear parameter transformation. We define quadratic basis polynomials for curved simplices in a way such that they are a generalization of the standard Bernstein-Bézier polynomials for non-curved simplices and guarantee continuity in function value along/on the shared edges/faces of all simplices. We define these generalized quadratic basis polynomials in physical space: In the 2D case, it is possible to think of a set of six simplex-specific quadratic basis polynomials, denoted as $\{Q_{i,j}(x,y)\}$, as a set of six quadratic polynomials satisfying certain interpolation conditions. We specify interpolation conditions at points $(x_{k,l}, y_{k,l})$ that are distributed uniformly with respect to arc length along the edges of a curved simplex.

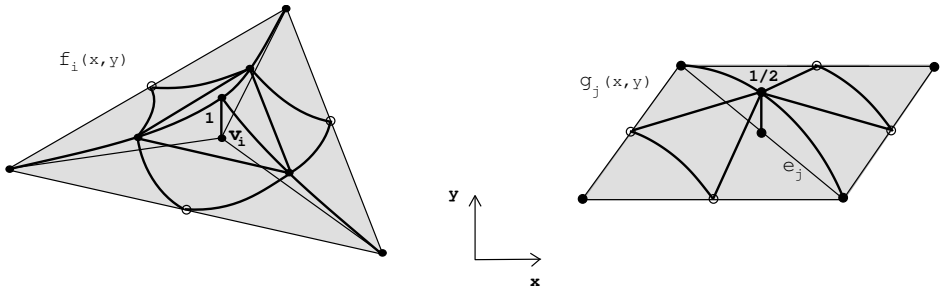


Figure 8 Graphs of quadratic Bernstein-Bézier basis polynomials.

Using the short-hand notation $Q_{i,j}^{k,l}$ for $Q_{i,j}(x_{k,l}, y_{k,l})$, the interpolation conditions, when written in matrix form, are given by

$$\begin{pmatrix} Q_{0,0}^{0,0} & Q_{0,0}^{1,0} & Q_{0,0}^{2,0} & Q_{0,0}^{0,1} & Q_{0,0}^{1,1} & Q_{0,0}^{0,2} \\ Q_{1,0}^{0,0} & Q_{1,0}^{1,0} & Q_{1,0}^{2,0} & Q_{1,0}^{0,1} & Q_{1,0}^{1,1} & Q_{1,0}^{0,2} \\ Q_{2,0}^{0,0} & Q_{2,0}^{1,0} & Q_{2,0}^{2,0} & Q_{2,0}^{0,1} & Q_{2,0}^{1,1} & Q_{2,0}^{0,2} \\ Q_{0,1}^{0,0} & Q_{0,1}^{1,0} & Q_{0,1}^{2,0} & Q_{0,1}^{0,1} & Q_{0,1}^{1,1} & Q_{0,1}^{0,2} \\ Q_{1,1}^{0,0} & Q_{1,1}^{1,0} & Q_{1,1}^{2,0} & Q_{1,1}^{0,1} & Q_{1,1}^{1,1} & Q_{1,1}^{0,2} \\ Q_{0,2}^{0,0} & Q_{0,2}^{1,0} & Q_{0,2}^{2,0} & Q_{0,2}^{0,1} & Q_{0,2}^{1,1} & Q_{0,2}^{0,2} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 1 & 4 \end{pmatrix}. \tag{13}$$

These interpolation conditions lead to the standard Bernstein-Bézier polynomials when applied to a non-curved simplex. The construction of the basis polynomials in the 3D case is based on the same principle.

Each generalized quadratic basis polynomial is zero outside the particular simplex for which it is defined. The quadratic basis polynomials associated with curved simplices are not as easily constructed and evaluated as those associated with non-curved simplices. Nevertheless, once the generalized quadratic basis polynomials are determined for all curved simplices, we can still compute inner products involving them by applying the change-of-variables theorem.

B Inner Products of Basis Polynomials

In the following, we define some of the required values of inner products of quadratic polynomials. We only consider the case of these polynomials being defined over the standard simplex in parameter space. For the quadratic Bernstein-Bézier basis polynomials $B_{\mathbf{i}}^2(\mathbf{u})$ defined for knots spaced uniformly along the edges of the standard simplex one obtains these values for inner products $I_{i,j}^{k,l} = \langle B_{i,j}, B_{k,l} \rangle$ in the 2D case:

$$\begin{pmatrix} I_{0,0}^{0,0} & I_{0,0}^{1,0} & I_{0,0}^{2,0} & I_{0,0}^{0,1} & I_{0,0}^{1,1} & I_{0,0}^{0,2} \\ I_{1,0}^{0,0} & I_{1,0}^{1,0} & I_{1,0}^{2,0} & I_{1,0}^{0,1} & I_{1,0}^{1,1} & I_{1,0}^{0,2} \\ I_{2,0}^{0,0} & I_{2,0}^{1,0} & I_{2,0}^{2,0} & I_{2,0}^{0,1} & I_{2,0}^{1,1} & I_{2,0}^{0,2} \\ I_{0,1}^{0,0} & I_{0,1}^{1,0} & I_{0,1}^{2,0} & I_{0,1}^{0,1} & I_{0,1}^{1,1} & I_{0,1}^{0,2} \\ I_{1,1}^{0,0} & I_{1,1}^{1,0} & I_{1,1}^{2,0} & I_{1,1}^{0,1} & I_{1,1}^{1,1} & I_{1,1}^{0,2} \\ I_{0,2}^{0,0} & I_{0,2}^{1,0} & I_{0,2}^{2,0} & I_{0,2}^{0,1} & I_{0,2}^{1,1} & I_{0,2}^{0,2} \end{pmatrix} = \frac{1}{180} \begin{pmatrix} 6 & 3 & 1 & 3 & 1 & 1 \\ 3 & 4 & 3 & 2 & 2 & 1 \\ 1 & 3 & 6 & 1 & 3 & 1 \\ 3 & 2 & 1 & 4 & 2 & 3 \\ 1 & 2 & 3 & 2 & 4 & 3 \\ 1 & 1 & 1 & 3 & 3 & 6 \end{pmatrix}. \tag{14}$$

In the 3D case, the needed values of inner products $I_{i,j,k}^{l,m,n} = \langle B_{i,j,k}, B_{l,m,n} \rangle$ are

$$\begin{pmatrix}
I_{0,0,0}^{0,0,0} & I_{0,0,0}^{1,0,0} & I_{0,0,0}^{2,0,0} & I_{0,0,0}^{0,1,0} & I_{0,0,0}^{1,1,0} & I_{0,0,0}^{0,2,0} & I_{0,0,0}^{0,0,1} & I_{0,0,0}^{1,0,1} & I_{0,0,0}^{0,1,1} & I_{0,0,0}^{0,0,2} \\
I_{1,0,0}^{0,0,0} & I_{1,0,0}^{1,0,0} & I_{1,0,0}^{2,0,0} & I_{1,0,0}^{0,1,0} & I_{1,0,0}^{1,1,0} & I_{1,0,0}^{0,2,0} & I_{1,0,0}^{0,0,1} & I_{1,0,0}^{1,0,1} & I_{1,0,0}^{0,1,1} & I_{1,0,0}^{0,0,2} \\
I_{2,0,0}^{0,0,0} & I_{2,0,0}^{1,0,0} & I_{2,0,0}^{2,0,0} & I_{2,0,0}^{0,1,0} & I_{2,0,0}^{1,1,0} & I_{2,0,0}^{0,2,0} & I_{2,0,0}^{0,0,1} & I_{2,0,0}^{1,0,1} & I_{2,0,0}^{0,1,1} & I_{2,0,0}^{0,0,2} \\
I_{0,1,0}^{0,0,0} & I_{0,1,0}^{1,0,0} & I_{0,1,0}^{2,0,0} & I_{0,1,0}^{0,1,0} & I_{0,1,0}^{1,1,0} & I_{0,1,0}^{0,2,0} & I_{0,1,0}^{0,0,1} & I_{0,1,0}^{1,0,1} & I_{0,1,0}^{0,1,1} & I_{0,1,0}^{0,0,2} \\
I_{1,1,0}^{0,0,0} & I_{1,1,0}^{1,0,0} & I_{1,1,0}^{2,0,0} & I_{1,1,0}^{0,1,0} & I_{1,1,0}^{1,1,0} & I_{1,1,0}^{0,2,0} & I_{1,1,0}^{0,0,1} & I_{1,1,0}^{1,0,1} & I_{1,1,0}^{0,1,1} & I_{1,1,0}^{0,0,2} \\
I_{0,2,0}^{0,0,0} & I_{0,2,0}^{1,0,0} & I_{0,2,0}^{2,0,0} & I_{0,2,0}^{0,1,0} & I_{0,2,0}^{1,1,0} & I_{0,2,0}^{0,2,0} & I_{0,2,0}^{0,0,1} & I_{0,2,0}^{1,0,1} & I_{0,2,0}^{0,1,1} & I_{0,2,0}^{0,0,2} \\
I_{0,0,1}^{0,0,0} & I_{0,0,1}^{1,0,0} & I_{0,0,1}^{2,0,0} & I_{0,0,1}^{0,1,0} & I_{0,0,1}^{1,1,0} & I_{0,0,1}^{0,2,0} & I_{0,0,1}^{0,0,1} & I_{0,0,1}^{1,0,1} & I_{0,0,1}^{0,1,1} & I_{0,0,1}^{0,0,2} \\
I_{1,0,1}^{0,0,0} & I_{1,0,1}^{1,0,0} & I_{1,0,1}^{2,0,0} & I_{1,0,1}^{0,1,0} & I_{1,0,1}^{1,1,0} & I_{1,0,1}^{0,2,0} & I_{1,0,1}^{0,0,1} & I_{1,0,1}^{1,0,1} & I_{1,0,1}^{0,1,1} & I_{1,0,1}^{0,0,2} \\
I_{0,1,1}^{0,0,0} & I_{0,1,1}^{1,0,0} & I_{0,1,1}^{2,0,0} & I_{0,1,1}^{0,1,0} & I_{0,1,1}^{1,1,0} & I_{0,1,1}^{0,2,0} & I_{0,1,1}^{0,0,1} & I_{0,1,1}^{1,0,1} & I_{0,1,1}^{0,1,1} & I_{0,1,1}^{0,0,2} \\
I_{0,0,2}^{0,0,0} & I_{0,0,2}^{1,0,0} & I_{0,0,2}^{2,0,0} & I_{0,0,2}^{0,1,0} & I_{0,0,2}^{1,1,0} & I_{0,0,2}^{0,2,0} & I_{0,0,2}^{0,0,1} & I_{0,0,2}^{1,0,1} & I_{0,0,2}^{0,1,1} & I_{0,0,2}^{0,0,2}
\end{pmatrix}$$

$$= \frac{1}{1260} \begin{pmatrix}
6 & 3 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 1 \\
3 & 4 & 3 & 2 & 2 & 1 & 2 & 2 & 1 & 1 \\
1 & 3 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 \\
3 & 2 & 1 & 4 & 2 & 3 & 2 & 1 & 2 & 1 \\
1 & 2 & 3 & 2 & 4 & 3 & 1 & 2 & 2 & 1 \\
1 & 1 & 1 & 3 & 3 & 6 & 1 & 1 & 3 & 1 \\
3 & 2 & 1 & 2 & 1 & 1 & 4 & 2 & 2 & 3 \\
1 & 2 & 3 & 1 & 2 & 1 & 2 & 4 & 2 & 3 \\
1 & 1 & 1 & 2 & 2 & 3 & 2 & 2 & 4 & 3 \\
1 & 1 & 1 & 1 & 1 & 1 & 3 & 3 & 3 & 6
\end{pmatrix}. \tag{15}$$

We use Romberg integration to compute inner products of quadratic basis polynomials $f_i(\mathbf{x})$ and $g_j(\mathbf{x})$ and the function $F(\mathbf{x})$ to be approximated.

Towards Automatic Feature-based Visualization

Heike Jänicke[1] and Gerik Scheuermann[1]

1 **Image and Signal Processing Group**
University of Leipzig
Leipzig, Germany
{jaenicke,scheuermann}@informatik.uni-leipzig.de

Abstract

Visualizations are well suited to communicate large amounts of complex data. With increasing resolution in the spatial and temporal domain simple imaging techniques meet their limits, as it is quite difficult to display multiple variables in 3D or analyze long video sequences. Feature detection techniques reduce the data-set to the essential structures and allow for a highly abstracted representation of the data. However, current feature detection algorithms commonly rely on a detailed description of each individual feature. In this paper, we present a feature-based visualization technique that is solely based on the data. Using concepts from computational mechanics and information theory, a measure, local statistical complexity, is defined that extracts distinctive structures in the data-set. Local statistical complexity assigns each position in the (multivariate) data-set a scalar value indicating regions with extraordinary behavior. Local structures with high local statistical complexity form the features of the data-set. Volume-rendering and iso-surfacing are used to visualize the automatically extracted features of the data-set. To illustrate the ability of the technique, we use examples from diffusion, and flow simulations in two and three dimensions.

1998 ACM Subject Classification I.3.6 Methodology and Techniques, I.3.7 Three-Dimensional Graphics and Realism

Keywords and phrases Feature Detection Techniques, Feature-based Visualization, Local Statistical Complexity

Digital Object Identifier 10.4230/DFU.SciViz.2010.62

1 Introduction

When analyzing their data-sets, one of the important questions of researchers is: Did I see everything that is relevant? Usually the domain experts can name several structures that they are interested in and that have significant influence on the system's evolution. Commonly, such structures are called features. In fluid dynamics, for example, they comprise structures like vortices, separation and attachment lines, cycles, and stagnation points. Detecting and visualizing these structures automatically is of great help for the domain experts. They get a simplified description of the system and can immediately understand basic properties of their data-set.

In order to detect these prominent structures in a data-set automatically, mathematical descriptions are required. Some features like stagnation points can be detected very easily, as they are simply zeros in the vector field. Other features like vortices, however, are very hard to define mathematically. Several different detection methods based on vorticity, λ_2, or the Sujudi and Haimes algorithm exist, but neither is capable of detecting vortices in all scenarios (Galilean invariance). The vortex example illustrates that more complex features are often hard to describe with a simple algorithm or formula, which gets even tougher in an unsteady setting. Here, a less restrictive feature definition would be beneficial.

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 62–77

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

A second problem that arises when looking for features is the fact that there is no general definition of a feature. In general, features are phenomena, structures or objects in a data set of interest for the underlying problem [19]. Thus, features strongly depend on the application and the user. Users from computational fluid dynamics (CFD), magnetic resonance imaging (MRI) and biological system simulation will be looking for different features and for each field a different set of tools is required to detect the structures the domain experts are interested in.

Even when provided with the appropriate set of tools, the user still has to run several algorithms to detect all the different features and usually has to specify parameters for each of them. Hence, the user has to start five to ten algorithms, set parameters, wait for the results, check whether something has been found, verify the results and look for structures that are not included in the list of standard features. Doing this entire procedure for several data-sets can become quite wearisome and much easier feature detection process would be desirable.

Summarizing these last three scenarios, we found the following weak points of the standard feature detection procedure:

- Most features can be found without domain knowledge even by a novice. Why can't computers do this?
- A feature may depend on the application. User dependence sounds weird in natural sciences (or engineering).
- If the data describes a physical simulation, a feature should depend only on the data.

To deal with these problems, the feature detection procedure described in the last scenario has to be highly simplified. The algorithm for the identification of relevant structures we think of, should look something like this:

1. Load the simulation data.
2. Run the feature detection algorithm.
3. Get a visualization with highlighted features (i.e. the most important regions).

Moreover, we want the logic behind the algorithm to be easy to understand and that the algorithm does not need a definition or name for all the different types of features it detects. The second requirement ensures that new structures can be found that have not been identified as features before.

Hence, the goal of the paper is to present a new way towards a feature-based visualization that does not need a priori definitions of structures that are considered relevant. On the contrary, relevance is to be directly defined by the data itself and the user is presented those structures that differ from the basic patterns in the data-set, i.e., the features of the data-set.

2 Related work

Much work has been done in the field of feature detection and visualization. In general four different concepts can be distinguished: image processing, topological analysis, physical characteristics, and partition-based approaches. *Image processing techniques*, e.g. Ebling et al. [6], Schlemmer et al. [24] and Heiberg et al. [9], often apply pattern matching approaches. Here a two or three-dimensional pattern is predefined and similar structures are found in the data-set. Although these techniques are very flexible with respect to finding certain patterns with different scale and/or orientation, the user still has to define a sample pattern as reference for each of the structures he/she is looking for. *Topological analysis* clusters regions of similar behavior/structure. Examples in this area can be found in the survey by Scheuermann et al. [23]. The topological analysis of a data-set provides an automatic simplification. However, there is no classification of the importance of the different structures that have been identified. Many feature detection methods are based on the analysis of *physical characteristics* (e.g. Garth et al. [7], Roth [20]), as these are the most intuitive descriptions for domain experts. Though many excellent methods fall into this category, they all have the problem

that they are very restrictive concerning the definition. A detailed description of flow feature detection techniques that fall into these three categories can be found in the survey by Post et al. [19]. The idea behind partition-based approaches is to separate the domain into regions of similar structure or behavior. Streamline predicates [22] and pathline attributes [28], for example, cluster integral lines in the data-set with given properties. The method we are going to present falls into this category, as it partitions the domain into areas that feature distinct structures and those that do not. Partitioned-based approaches are summarized in the paper by Salzbrunn et al. [21].

As mentioned before, these standard feature detection methods commonly rely on a given definition or description of the feature to be found. What we are looking for is a feature description given by the data-set itself. The third step of the algorithm we have in mind (Section 1) already reveals the direction we are aiming at. We do not want to provide exact feature definitions, but are looking for regions of high importance in the data-set. Important is to be understood in an information-theoretic way, i.e. we want to identify the regions with the highest information content or complexity.

In the literature, a large variety of complexity measures are available, e.g., [2, 4, 8, 16, 27]. Common measures originating from the analysis of strings of data are Shannon entropy [27] and algorithmic information [1]. Shannon entropy is a measure of the uncertainty associated with a random variable, whereas the algorithmic information is roughly speaking the length of the shortest program capable of generating a certain string. Both measures have in common that they are measures of randomness. In complex systems however, randomness is commonly not considered to be complex. Likewise, Hogg and Huberman [11] state that complexity is small for completely ordered and completely disordered patterns and reaches a maximum inbetween. A different approach was taken by Grassberger [8], who defined complexity as the minimal information that would have to be stored for optimal predictions. Based on this idea, statistical complexity [4] was introduced identifying the complexity of a system with the amount of information needed to specify its causal states, i.e., its classes of identical behavior. In order to analyze random fields, a point-by-point version was formulated by Shalizi [25] called local statistical complexity.

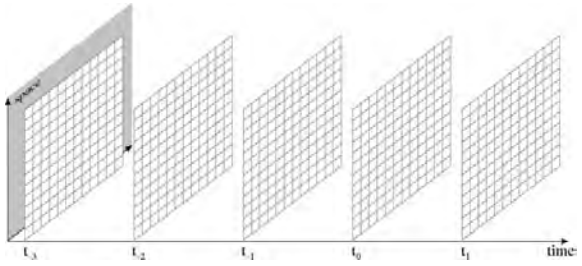
3 Specifications

The following work is based on the ideas by Shalizi et al. [26], which assumes the following properties of a data-set:

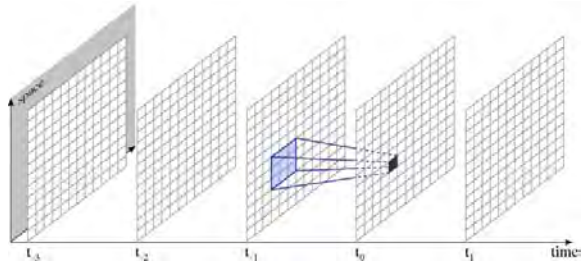
1. The data stems from a PDE simulation (in engineering or natural sciences).
2. The solver is based on a finite difference scheme on a Cartesian grid.
3. The data is unsteady and all time-steps and independent variables are available.

The first requirement ensures that the process creating the data is the same at each position in the resulting field. As PDEs are the standard definition of physical systems, e.g. the Navier-Stokes equations for fluid flow, this demand sets no limitations. The second requirement allows for the comparison of local neighborhoods. A sample Cartesian grid is given in Figure 1a. Using finite difference schemes as solver, clearly defines a local neighborhood that is used to compute the value in the next time-step as illustrated in Figure 1b. Moreover, initial conditions and boundary conditions are required for the computation. In the results section we will use data-sets that were computed using more sophisticated solvers and show that this is no crucial restriction. The third requirement ensures that exact conclusions about the influence of different positions and variables can be made.

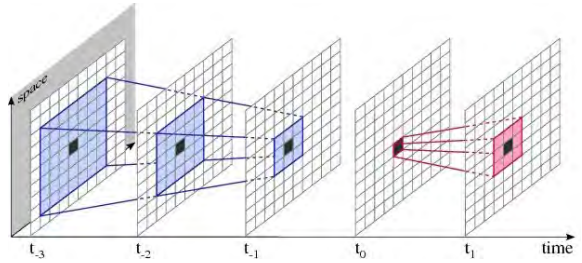
Looking closer at these three requirements, we see that they correspond to the construction rules of cellular automata, which are well researched.



(a) Cartesian grid.



(b) Finite difference.



(c) Light-cone configurations.

Figure 1 Different structures in a Cartesian grid: (a) Empty grid. (b) Sample neighborhood used to compute finite differences. (c) Light-cone structures used for the computation of local statistical complexity.

4 Cellular Automata

A cellular automaton (CA) is a discrete model of a system, with the game of life being the best-known example. The automaton consists of a regular uniform lattice with a discrete variable at each cell. The configuration of an automaton at a certain time step is completely specified by the values of the variables at each site. Following predefined local rules the configuration can change at each discrete time step. A rule defines which value a cell will take in the next step, depending on the values of its neighborhood in the present. Typically, the neighborhood of a cell consists of the cell itself and all immediately adjacent cells. An example for a rule is: If the cell has value 0 and at least two of its neighbors have value 1, change the cell's value to 1. For each time step all values are updated simultaneously.

An example of a 1D cellular automaton is given in Figure 2. The domain can be separated into two different classes: stable local patterns and defects. The stable local patterns are the areas, that look like the background of the image. The defects are the triangles in different sizes that move across the image. Although the stable patterns dominate after some time, the defects are the ones that determine the long term behavior. Shalizi et al. [26] proposed a filter for the automatic extraction of coherent structures, i.e. defects, in CA. Their filter is called local statistical complexity and automatically

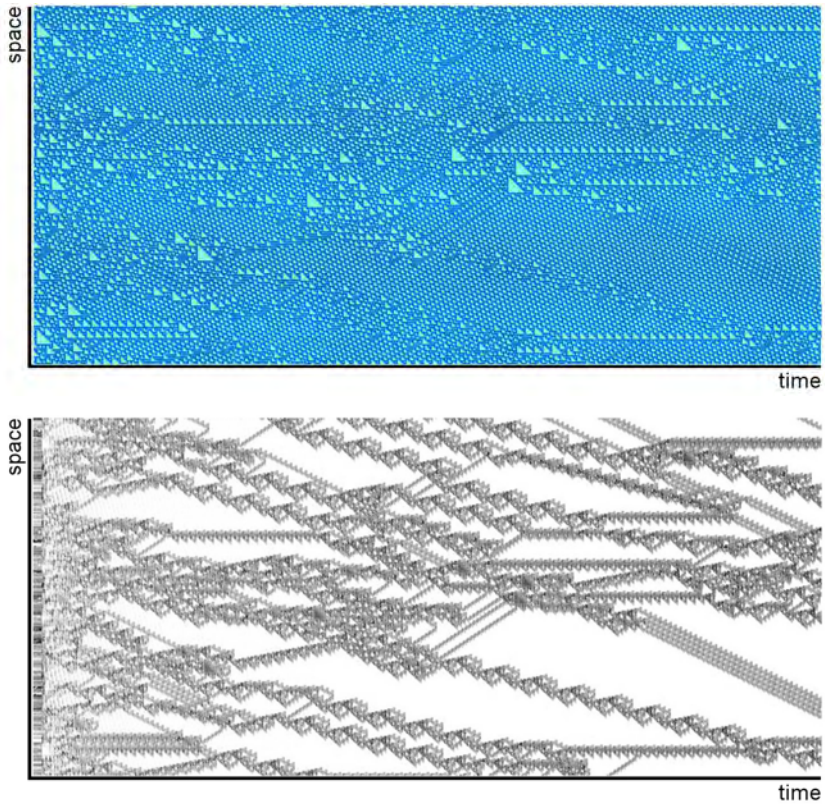


Figure 2 Cellular automaton in 1D (top) and corresponding local statistical complexity field (bottom).

detects prominent formations of arbitrary size and shape in unsteady data-sets. Figure 2(bottom) shows the filtered image of the 1D cellular automaton, highlighting the defects that move around in the original data-set.

5 Local Statistical Complexity

Local statistical complexity extracts those regions in an unsteady field, where a lot of information from the local past is required to predict the dynamics in the local future. This happens where the temporal evolution is very unusual compared to what happens in the rest of the field. In general, users are interested in a subset of these distinctive regions, as they know the basic structure of their data-set and want to find regions that behave differently. Especially for large intricate and little understood data-sets local statistical complexity is a helpful tool to guide the user to regions that might be relevant for him or her.

Local statistical complexity focuses on the local temporal evolution of the field. The local past of position p in the field consists of all the points that might influence p. As effects propagate at finite speed, the past has the shape of a light-cone that is directed towards the past. The apex is located at p. This concept is likewise used when computing simulations using finite differences or finite elements. Here the value at position $\vec{x}$ in time-step t is computed from the neighborhood of the point in the previous time-step $t-1$ (Fig. 1c). (An exception is pressure in incompressible flow.) The future is given by a light-cone that is directed in the opposite direction, i.e., the future. Each light-cone comprises a set of positions. The values at these positions together with the neighborhood information are called a configuration. A configuration can be thought of as a pattern that extends in time, space

and if appropriate over multiple variables. By definition future configurations contain the value at the apex, past configurations do not.

For each past-cone configuration we would like to be able to predict, what might happen in the future. The only value that we can predict exactly, is the one at the future-cone apex, as it results from the calculation rule of the simulation method (remember Fig. 1b). To predict the remaining values in the future-cone, we need statistics. We group several similar past-configurations and compute a histogram over the different futures that occur. This estimated distribution tells us which future configurations are likely for this particular class of behavior in the past. This procedure is repeated for all different groups of past-configurations.

Analyzing the histograms we computed in the previous step, we will observe that some of them are very similar. This means that the differences we detected in the past-configurations have no significant influence on the dynamics in the future. Thus, we merge all those past groups that have very similar histograms. The different groups that result after the merging are called causal states. A causal state represents a cause-and-effect relationship between what was observed in the past and what might happen in the future. So, if we have a past configuration and can determine its causal state, we can estimate the most probable future dynamics.

Now that we can predict the dynamics in the future given the past configuration, we want to find a minimal lossless encoding for this information. The code with the shortest expected length is given by a Huffman-code. A Huffman-code assigns frequent symbols short codewords and rare symbols longer ones. The entropy $H[X]$ is a measure of the smallest average codeword length that is theoretically possible for the given alphabet X. For functions $f(x)$ $x \in X$, mutual information $I[f(X),X]$ equals entropy $H[f(X)]$. In order to find an optimal encoding for the past-configurations, we have to find a function f that minimizes the mutual information $I[f(PastConf);PastConf]$. Shalizi et al. [26] showed that the unique function that minimizes the mutual information is the mapping to the causal states. Thus, if we store at each position the Huffman-code of the corresponding causal state, we resolve the file with shortest expected length that still gives us all informations about the dynamics in the local future.

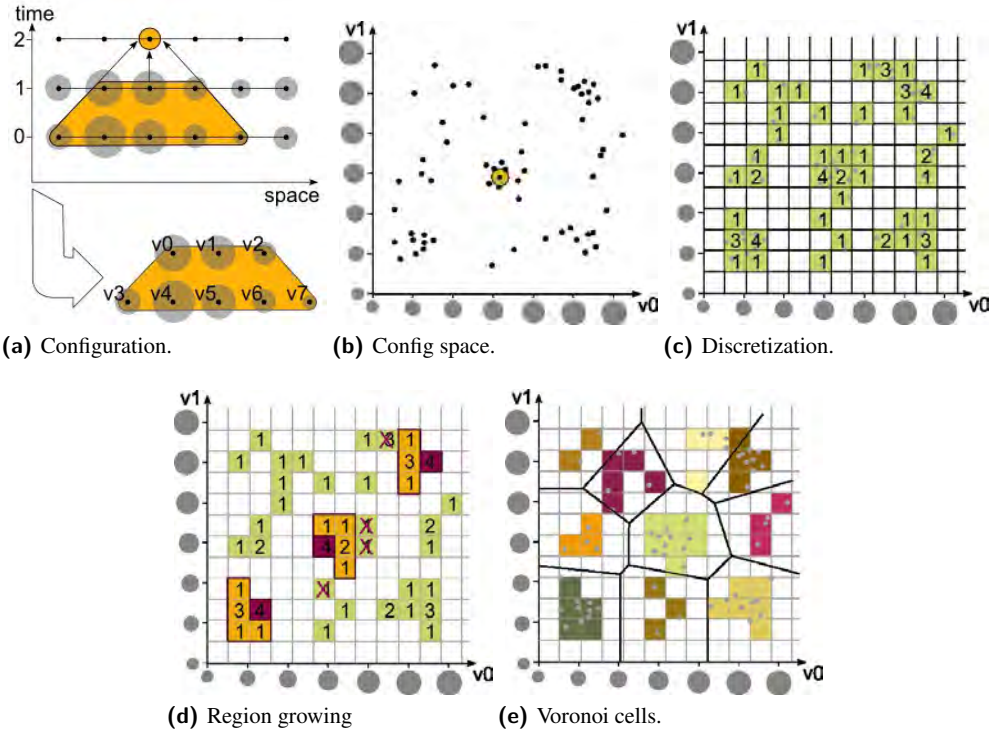
The encoded file can finally be used to detect distinctive regions. The Huffman-code assigns each causal state a codeword whose length depends on the number of positions that are assigned to it. Causal states with a very long codeword feature dynamics in the future that occur very rarely in the field. Local statistical complexity measures for a past-configuration the length of the codeword of the corresponding causal state, i.e., the amount of information that is needed to predict the causal state/the dynamics in the future. The longer the codeword, the more likely it is that something extraordinary is going to happen in the local future of this position. More information on the theory and implementation of local statistical complexity and causal states can be found in [26, 14].

6 Application to Finite Difference Schemes

Complexity analysis using local statistical complexity can be applied to scientific simulations as finite difference schemes, a direct analog to CA rules, can be used to discretize PDEs. The following simple example of an isotropic diffusion, e.g., ion concentration in water, is used for illustrations. Given a concentration $f(\vec{x},t_0)$ at each position $\vec{x} \in B$ at time t_0, the temporal development of this concentration $f(\vec{x},t)$ is observed. The governing PDE is

$$\frac{\partial f}{\partial t}(\vec{x},t) = D\Delta f(\vec{x},t) \tag{1}$$

with a constant diffusion coefficient D, time derivative $\frac{\partial f}{\partial t}(\vec{x},t)$ and Laplacian $\Delta f(\vec{x},t)$. As boundary conditions constant concentrations are assumed: $f(\vec{x},t) = f(\vec{x},t_0)$ for $x \in \partial B$. A simple finite



(a) Configuration. **(b)** Config space. **(c)** Discretization.

(d) Region growing **(e)** Voronoi cells.

Figure 3 Density-driven Voronoi tessellation: (a) A past configuration extracted from the data-set consisting of eight variables. (b) This configuration marked in high-dimensional configuration space (only first two of the eight variables illustrated). (c) Initial fine-grained discretization of the configuration space. (d) Density-driven region growing starting in densest regions. (e) Final Voronoi tessellation of the configuration space. (f) Final partitioning of the domain.

difference scheme in the plane consists of a Cartesian lattice $L = \{0,\ldots,255\} \times \{0,\ldots,255\}$, a given concentration $f_0 : L \to \mathbb{R}$, and the difference equation

$$\begin{aligned} f(x_1,x_2,t+1) \quad = \quad & \frac{1}{16} f(x_1-1,x_2+1,t) + \frac{1}{8} f(x_1,x_2+1,t) + \frac{1}{16} f(x_1+1,x_2+1,t) + \\ & \frac{1}{8} f(x_1-1,x_2+0,t) + \frac{1}{4} f(x_1,x_2+0,t) + \frac{1}{8} f(x_1+1,x_2+0,t) + \\ & \frac{1}{16} f(x_1-1,x_2-1,t) + \frac{1}{8} f(x_1,x_2-1,t) + \frac{1}{16} f(x_1+1,x_2-1,t) \end{aligned} \tag{2}$$

which is also known as applying a binomial 3×3 filter to a digital image in image processing [12]. In this example L is the lattice of the CA, f contains the values over time and Eq. 2 gives the complete rule. As $c = 1$, the configurations are as illustrated in Fig. 1c. The reader familiar with either finite difference schemes or image processing might imagine a larger stencil or filter for $c > 1$. Similar schemes can be applied to any PDE, allowing for analysis using local statistical complexity.

7 Computation of Local Statistical Complexity

The first step in visualizing the local statistical complexity of a data-set consists of the computation of causal states. Causal states are defined by:

$$\text{Causal State} = \varepsilon(l^-) = \{\lambda : P(l^+|\lambda) = P(l^+|l^-)\}. \tag{3}$$

Hence, a causal state is the equivalence class of all past-cones (l^-) that have the same distribution ($P(l^+|l^-)$) over possible futures (l^+), i.e., each causal state predicts a certain future and the possible futures of different causal states differ.

To determine the causal states that occur within a data-set, the conditional probabilities $P(l^+|l^-)$ have to be estimated. As exactly the same pattern l^+ or l^- commonly only occurs once in a scientific data-set, the probabilities cannot be estimated directly, but similar configurations have to be grouped for the estimation. The grouping has to fulfill two requirements. First, all samples in the data-set have to be assigned to a group and second, the size of each group in high-dimensional space (dimensionality is given by the number of entries in the cone, cf. Fig. 1c) has to be the same to allow for a correct estimation.

In [13], Jänicke et al. proposed a fast strategy to estimate probabilities with a single sweep through the data. We use this approach based on density-driven Voronoi tessellation, which consists of three steps:

1. *Discretization*: Compute the past- and future-cone (Fig. 3a) at each position and store the discretized cones in two trees.
2. *Density-driven Voronoi Tessellation*: Partition the high-dimensional discrete cone space (Fig. 3b) using a Voronoi tessellation (Fig. 3(c-e)) that takes the underlying distribution of cone configurations into account. This step is performed for the past and future tree separately. Resulting IDs are stored for each leaf in the two trees.
3. *Probability Estimation*: For each past cell, the corresponding future cells are counted and used to estimate the probabilities.

The idea behind density-driven Voronoi Tessellation is to let the discretization adapt to the structure of the high-dimensional data. The initial discretization in Step 1. is used to estimate a local density. Starting from densest regions, a region growing algorithm is applied that iteratively captures the entire space. The method ensures that the Voronoi cells have equal size and that clusters are well preserved.

To identify causal states, the conditional probabilities have to be estimated. This is achieved by counting the number of occurrences of different future classes per past Voronoi cell. Dividing by the total number of configurations per past cell, gives the conditional probability $P(l^+|l^-)$. In a last step, those Voronoi past cells are grouped that feature a similar distribution over futures using a χ^2-test [10]. The resulting grouped classes are the causal states of the process.

Each of these causal states represents a spatio-temporal pattern, indicating what might happen next if a certain past was observed. After the identification of the causal states, new fields that hold the ID of the causal state at each position are created. As we are not interested in the local pattern but in the complexity of the current position, we have to evaluate the local statistical complexity of each causal state and assign appropriate values to the field of causal state IDs.

Local statistical complexity measures how much information from the local past is required to predict the dynamics in the local future at a certain position. If the dynamics of a configuration match the average behavior in the data-set, only little information is required. On the contrary if something unusual happens, more information is required. To measure how extraordinary some local dynamics are, Shalizi et al. [26] proposed local statistical complexity, which was extended to scientific simulation data by Jänicke et al. ([14, 13]). The local statistical complexity at a certain position p in the field is defined as the mutual information between the corresponding configuration's past (l^-) and its causal state ($\varepsilon(l^-)$):

$$LSC(p) = I[\varepsilon(l^-); l^-]. \tag{4}$$

Mutual information is a measure from information theory, which tells how much information one

random variable contains about another one:

$$I[A;B] = \sum_{a\in A; b\in B} P(a,b)\log_2 \frac{P(a,b)}{P(a)P(b)} \tag{5}$$

where $P(a)$ is the probability that the random variable A takes the value a and $P(a,b)$ is the corresponding joint probability of variables A and B. Using this definition, the local statistical complexity of a cone configuration tells how much information from the past is required to identify its causal state. If one knows the causal state, the dynamics in the future are clear as well. Hence, if a lot of information is required to identify the causal state, the local dynamics are extraordinary compared to what is happening in the rest of the data-set.

8 Results

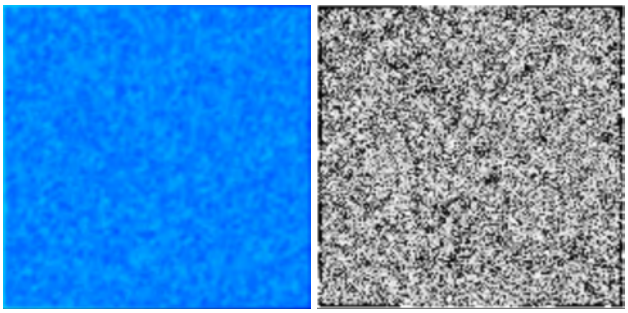
The three data-sets we are going to analyze have increasing complexity. The first one is an isotropic diffusion which is a perfect analogon to CA. In the second example we will analyze swirling flow. This 2d examples consists of multiple variables and contains different features experts are interested in. The third test-case is a simulation of the flow around a delta wing. In this large 3d example several intricate features are present. The results of local statistical complexity will be compared to standard feature detection techniques for both CFD examples to verify the correctness of the automatic detected features.

8.1 Isotropic Diffusion

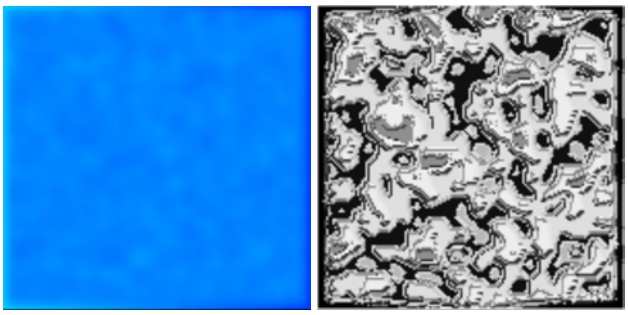
An isotropic diffusion, simulated using finite differences as explained in Section 6, is a simple example of a large variety of diffusion processes, i.e., equalization of differences in concentration, heat, matter or momentum, appearing in nature. The dataset is simulated by repeated filtering using a binomial filter. In the diffusion field, the cells at the left border are set to 1, and those at the right border to 0. Upper and lower boundaries are initialized with linearly decreasing values that range from 1 to 0. The inner part is initialized with random values between 0.0 and 1.0. The process displayed in the upper row of Fig. 4, is defined on a square lattice with 150 cells in each direction. 800 time-steps are simulated.

The left half of the images in Figure 4 shows the evolution of the diffusion. In time-step 1 the image consists of many small coherent structures that still feature a large variety of values. After 20 time-steps these homogeneous regions have become much larger and the range of values has shrunk. At the boundaries small bands with the extremal values are visible. This process continues in time-step 50. While the center becomes more homogeneous, the gradients starting from the boundaries grow. In time-step 800 half of the domain has reached the equilibrium of the diffusion process.

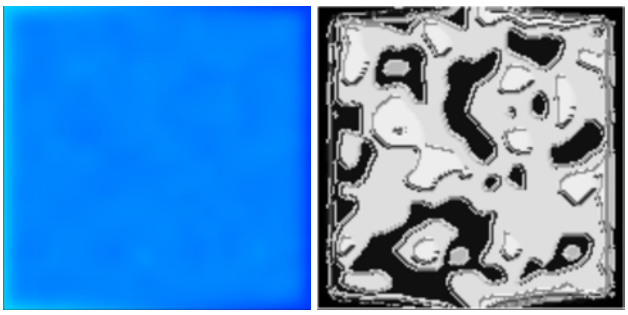
On the right hand-side of this series of snapshots, the corresponding complexity fields are depicted. In the first time-step the entire domain is covered by small black and gray spots. The areas that appear in light gray, are those that hold values close to 0.5, the most common value in this data-set. Black cells hold formations that have either very different or extremal values in their configurations. In time-step 20 the diffusion has formed larger homogeneous regions, which are found by local statistical complexity. Again, black areas indicate extreme values and gray areas normal ones. These areas grow (time-step 50) until in time-step 200 the entire center of the data-set holds value 0.5. Thus, this pattern is the basic one and considered to be uninteresting. In time-step 800 the gradient covers half of the data-set. As we only analyze those time-steps in which the gradient evolves, these configurations with increasing/decreasing values are something extraordinary, whereas configurations containing only value 0.5, the standard result of the diffusion process, are considered to be normal.



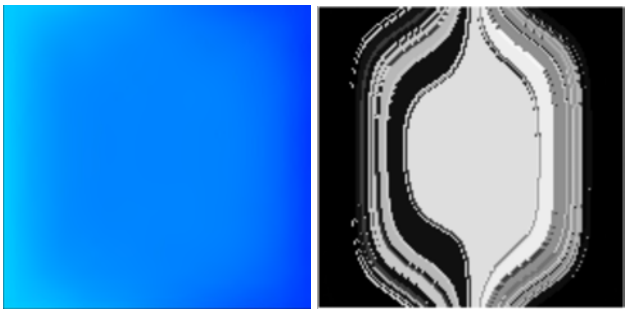
(a) Time-step 1.



(b) Time-step 20.

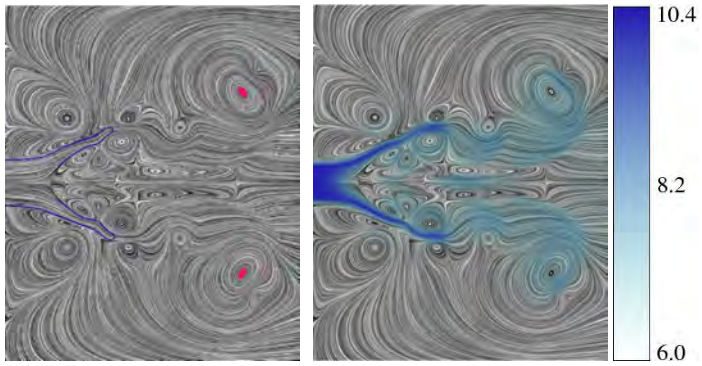


(c) Time-step 50.

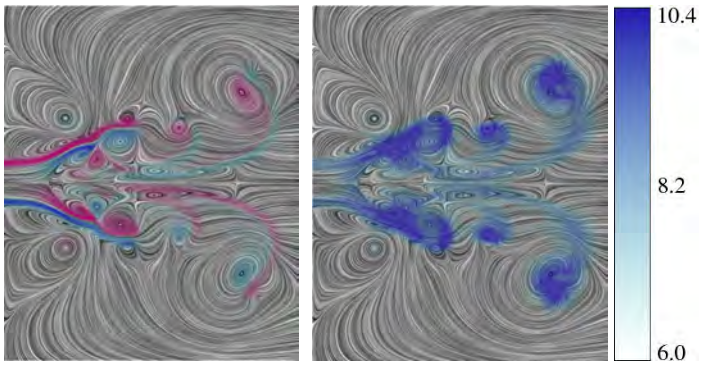


(d) Time-step 800.

Figure 4 Evolution of the diffusion data-set (left - original values, right - lsc field): (a) (left) Random initialization in time-step 1. (right) Local patterns that are close to equilibrium occur in light-gray in the lsc-field, which indicates small complexity. Pixels colored in black include extremal values which seldom occur in the entire unsteady data-set. (b) Time-step 20: Coherent structures start to form. (c) Time-step 50: Coherent structures in the center grow. Large areas reach equilibrium (light-gray in center). The gradient grows starting from the boundary. (d) Time-step 800: The center has reached equilibrium (value 0.5) and the boundary gradient grows further.



(a) Original. **(b)** Norm of velocity.



(c) Vorticity. **(d)** Local statistical complexity.

Figure 5 Swirling flow: In each image the line integral convolution (LIC) of the velocity field is overlayed with an additional quantity. (a) The conical shear region (blue) outlines the region where the flow enters the domain. Two red points mark one of the ring-like vortex structures. (b) The norm of velocity overlay highlights regions with a strong current and reveals the relevant structures. (c) Vorticity indicates strong swirling motion. The color-coding gives the orientation. (d) Local statistical complexity automatically extracts analog structures.

8.2 Swirling Flow

The development of a recirculation zone in a swirling flow is investigated by numerical simulation. This type of flow is relevant to several applications where residence time is important to enable mixing and chemical reactions.

The unsteady flow in a swirling jet is simulated with an accurate finite-difference method. The Navier-Stokes equations for an incompressible, Newtonian fluid are set up in cylindrical coordinates assuming axi-symmetry in terms of streamfunction and azimuthal vorticity. All equations are dimensionless containing the Reynolds number Re and the swirl number S as defined by Billant et al. [3]

$$Re \equiv \frac{v_z(0,z_0)D}{\nu} \qquad S \equiv \frac{2v_\theta(R/2,z_0)}{v_z(0,z_0)} \tag{6}$$

where $z_0 = 0.4D$, $D = 2R$ is the nozzle diameter and ν the kinematic viscosity, as dimensionless parameters.

The PDEs are discretized with fourth order central difference operators for the non-convective terms and with a fifth order, upwind-biased operator [17] for the convective terms. The time integrator is an explicit s-stage, state space Runge-Kutta method ([5], [15]), the present method is fourth order

accurate with $s = 5$. The time step is controlled by the minimum of two criteria: The limit set by linearized stability analysis and the limit set by the error norms of an embedded third order Runge-Kutta scheme [5]. The Helmholtz PDE for streamfunction $\tilde{\Psi}(r,z,t)$ is solved with an iterative method using deferred corrections and LU-decomposition of the coefficient matrix. The deferred corrections method is designed to reduce the bandwidth of the coefficient matrix. It converges rapidly using about ten to twenty steps.

The flow domain is the meridional plane $\mathscr{D} = \{(r,z) : 0 \leq r \leq R, 0 \leq z \leq L\}$ with $R = 5D$, $L = 8D$ and D denoting the nozzle diameter at the entrance boundary. The flow domain is mapped onto the unit rectangle which is discretized with constant spacing. The mapping is separable and allows to a limited extent crowding of grid points in regions of interest. The present simulation uses $n_r = 91$ and $n_z = 175$ grid points in radial and axial directions. The boundary conditions are of Dirichlet type at the entrance section and the outer boundary and at the exit convective conditions are imposed for the azimuthal vorticity. The initial conditions are stagnant flow and the entrance conditions are smoothly ramped up to their asymptotic values within four time units.

The simulation results for $Re = 10^3$, $S = 1.1$ (within the range of the experiments [3], [18]) used for the complexity analysis are ten time steps after the formation of the recirculation bubble (which forms at $t = 6.02$) at times $t = 33.63092$ to $t = 33.70560$. The flow is unsteady and does not approach a steady asymptotic state as the velocity and vorticity fields show (Fig. 5(a-c)).

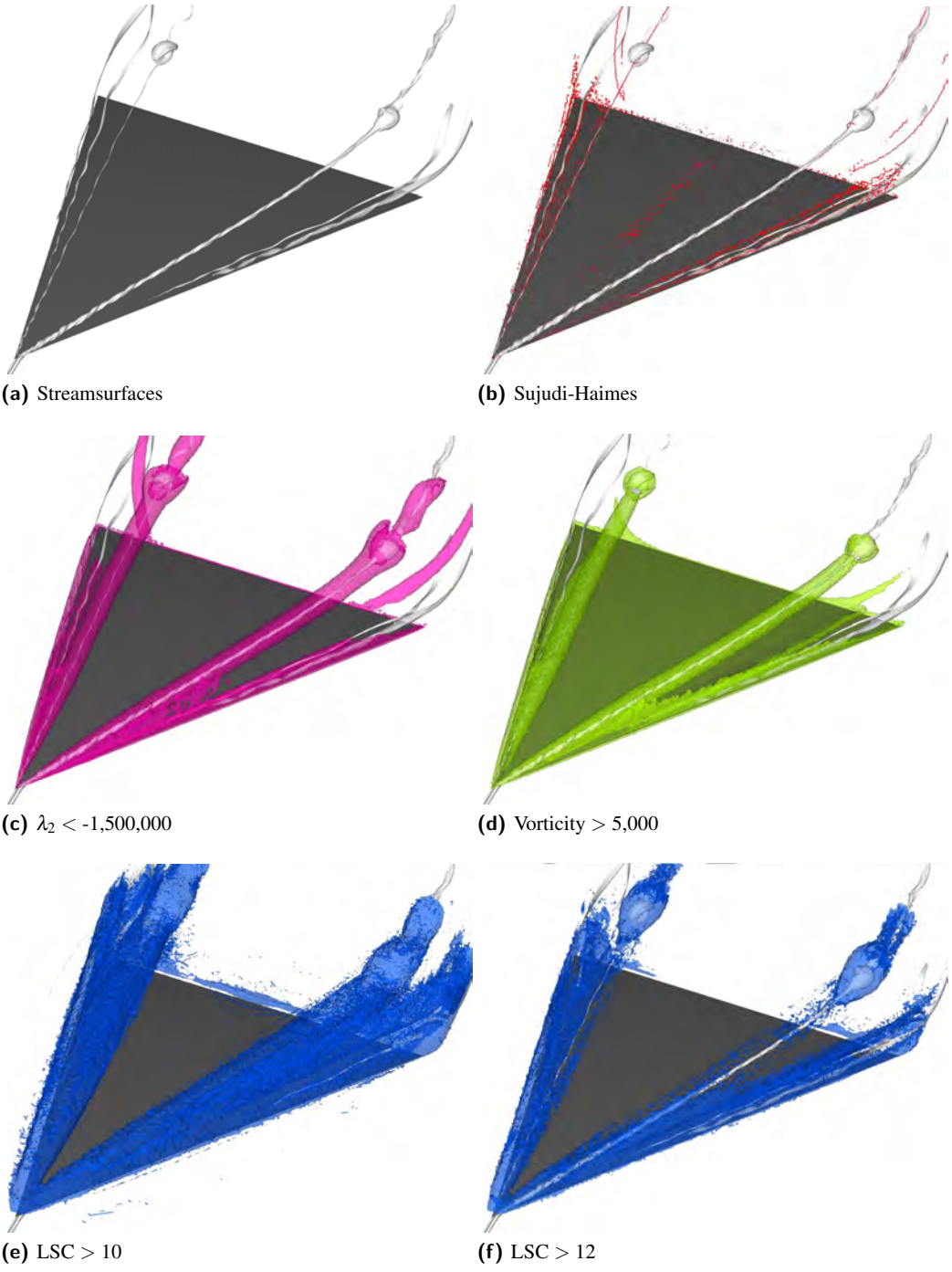
Figure 5(a) shows a line integral convolution (LIC) of the velocity field, featuring several vortices. Relevant features are highlighted in this image. The structure outlined in blue is the conical shear region surrounding the inlet of the swirling flow. The two red dots indicate one ringlike vortex structure. The coreline of this vortex lies in a plane orthogonal to displayed cross-section and passed through the red points. Comparing this image to the one overlayed with the norm of velocity (Fig. 5b), we see that the simple LIC image gives a misleading impression of the flow as several of the clearly visible vortices are detected in regions close to noise.

The vorticity overlay in Figure 5c results in a similar image as norm of velocity. Basically the same structures are highlighted. Differences occur at the inlet, where, as expected by the technique, only the shear flow is highlighted. Moreover, the ring-like vortex structures are more pronounced than the connecting structures. The color-coding provides an additional hint telling the orientation of the rotation.

Local statistical complexity is computed for the combination of velocity and vorticity and layed over the original LIC to provide context (Fig. 5d). Both features, the shear region and the ring-like vortex structures, are automatically detected by local statistical complexity. Unlike vorticity, local statistical complexity marks both features as equally complex. Both, the conical shear region, as well as the vortex structure are assigned highest complexity, while the vortices exhibit only small vorticity, compared to the shear flow.

8.3 Delta Wing

This data-set represents the airflow around a delta wing at low speeds with an increasing angle of attack. Multiple vortex structures form on the wing due to the rolling-up of the viscous shear layers that separate from the upper surface. These formations of three vortices can be observed on either side of the wing (Fig. 6a). With increasing angle of attack the intensity of the primary vortices (purple) increases until in time-step 700 a vortex breakdown occurs. This phenomenon is characterized by rapid deceleration of both the axial and tangential mean velocity components inside the vortex. During breakdown, the axial mean velocity component vanishes and becomes negative on the axis of the vortex, corresponding to appearance in the flow structure of a stagnation point followed by a recirculation bubble. The analysis of vortex breakdown is highly interesting, as it is one of the limiting factors of extreme flight maneuvers. The extraction and visualization of the individual



(a) Streamsurfaces **(b)** Sujudi-Haimes

(c) $\lambda_2 <$ -1,500,000 **(d)** Vorticity $>$ 5,000

(e) LSC $>$ 10 **(f)** LSC $>$ 12

Figure 6 Delta Wing: (a) Streamsurfaces to indicate the vortices above the delta-wing. (b) Sujudi-Haimes vortex detection algorithm applied to the vector field. (c) Isosurface in the λ_2-field (isovalue = -1,500,000). (d) Isosurface in the vorticity field (isovalue = 5000). (e,f) Isosurface in the local statistical complexity field of the norm of velocity (Isovalue = 10 (e), Isovalue = 12 (f)).

structures, however, is still a challenging task as the different structures are nested and interact with each other. The unstructured grid was resampled on a 292x224x75 grid ($\sim$ 4.1 Million positions) and consists of more than 1000 time-steps. The images in Figure 6 depict time-step 700.

Images 6(b-d) give an overview over standard vortex detection techniques. The algorithm by Sujudi and Haimes [29] (Fig. 6b) is a technique that detects vortex core-lines. Applied to the delta wing, this method perfectly extracts the core-line of the major vortices. However, we only get a vague indication of the core-lines close to the surface, whose vortices are less dominant and interact with each other. The λ_2-criterion extracts the "hull" of the vortex. Finding an appropriate isovalue (-1,500,000) to separate the two minor vortices without missing the recirculating bubble takes some time. The isosurface of the magnitude of the vorticity (Fig. 6d) gives approximately the same result.

Figures 6(d–f) show the local statistical complexity of the norm of velocity. Figure 6e shows all positions that are assigned a complexity value greater than 10 (maximum: 14.7). The visualized structures do not only comprise the vortices and the recirculation bubble, but also the regions at the outer corners of the wing, where the flow from the smaller vortices and the flow from underneath the wing interact and form a swirling motion that is classified by the other techniques as vortex. Increasing the complexity value further (Fig. 6f), we see that the individual vortices are better separated. The major vortices are no longer visible as their complexity value is smaller than those of the small vortices. This observation means, that the local temporal evolution of the norm of the velocity is very distinct for vortices and for the recirculating bubble. The exceptional behavior of the norm of the velocity is a typical characteristic for recirculating bubbles, as was explained earlier. With our method we can extract these distinctive formations automatically without defining a definite pattern beforehand. This feature is an important characteristic of our method, as it is capable of identifying structures that exhibit an extraordinary formation without precisely describing its pattern.

9 Conclusion

In this paper we described a filter called local statistical complexity based on concepts from information theory which automatically extracts coherent structures from unsteady multi-fields. It assigns each position in the data-set a scalar value whose magnitude depends on how extraordinary the local dynamics at the current position are. Color-mapping or isosurfacing can be used to visualize the most distinct structures in the data-set.

Local statistical complexity is intrinsic to unsteady multi-field visualization as this is required by the theory and quantities of different type (scalar, tensor, vector valued) can be used simultaneously in the computation. The process reduces the multi-field to a single scalar field giving the importance of each position. The entire process is fully automatic and requires no application-specific knowledge. (The user has to provide two parameters for the Voronoi tessellation, which could be estimated as well.)

Current problems arise, when analyzing divergence-free flow, as the concept of local influence propagation is not preserved. When the dynamics are too turbulent, memory costs increase a lot, as many different configurations have to be stored. The alternative is to compute coarser causal states, which makes the results more inaccurate.

In our future work we would like to work on a complete mathematical basis in the continuous case. More research has to be done regarding the influence of the parameters in the Voronoi tessellation process. The original concept was designed for PDEs solved using finite differences. Extending the theory to other numerical schemes is a further task that should be addressed in the future.

10 Acknowledgements

The authors wish to thank Wolfgang Kollmann for helpful discussion concerning the application of local statistical complexity to data-sets from fluid dynamics and for providing the swirling flow data-set. The delta wing data-set was provided by Markus Rütten from the German Aerospace Centre (DLR).

References

1 R Badii and A Politi. *Complexity: Hierarchical Structures and Scaling in Physics*. Cambridge University Press, Cambridge, 1997.

2 W. Bialek, I. Nemenman, and N. Tishby. Predictability, complexity, and learning. *Neural Computation*, 13:2409–2463, 2001.

3 P. Billant, J.M. Chomaz, and P. Huerre. Experimental study of vortex breakdown in swirling jets. *JFM*, 376:183–219, 1999.

4 James P. Crutchfield and Karl Young. Inferring statistical complexity. *Phys. Rev. Lett.*, 63(2):105–108, July 1989.

5 G. Wanner E. Hairer. *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems, 2nd ed.* Springer Verlag, Berlin, 1996.

6 Julia Ebling and Gerik Scheuermann. Clifford convolution and pattern matching on vector fields. In *Proceedings of the 14th IEEE Visualization 2003*, pages 193–200, Washington, DC, USA, 2003. IEEE Computer Society.

7 Christoph Garth, Florian Gerhardt, Xavier Tricoche, and Hans Hagen. Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1464–1471, 2007.

8 Peter Grassberger. Toward a quantitative theory of self-generated complexity. *International Journal of Theoretical Physics*, 25(9):907–938, September 1986.

9 Einar Heiberg, Tino Ebbers, Lars Wigstrom, and Matts Karlsson. Three-dimensional flow characterization using vector pattern matching. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):313–319, 2003.

10 P. G. Hoel, S. C. Port, and C. J. Stone. *Introduction to Statistical Theory: Ch. 3 - Testing Hypotheses*. New York: Houghton Mifflin, 1971.

11 T Hogg and BA Huberman. Order, complexity and disorder. *Xerox Palo Alto Research Center (preprint)*, 1985.

12 Bernd Jähne. *Digital image processing: concepts, algorithms and scientific applications*. Springer-Verlag, London, UK, 1991.

13 Heike Jänicke, Michael Böttinger, Xavier Tricoche, and Gerik Scheuermann. Automatic Detection and Visualization of Distinctive Structures in 3D Unsteady Multi-fields. *Computer Graphics Forum*, 27(3):767–774, 2008.

14 Heike Jänicke, Alexander Wiebel, Gerik Scheuermann, and Wolfgang Kollmann. Multifield visualization using local statistical complexity. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1384–1391, 2007.

15 Christopher A. Kennedy, Mark H. Carpenter, and R. Michael Lewis. Low-storage, explicit Runge-Kutta schemes for the compressible Navier-Stokes equations. *Appl. Numer. Math.*, 35(3):177–219, 2000.

16 A. Kolmogorov. Logical basis for information theory and probability theory. *IEEE Trans. on Information Theory*, 14(5):662–664, September 1968.

17 Yuguo Li. Wavenumber-extended high-order upwind-biased finite-difference schemes for convective scalar transport. *J. Comput. Phys.*, 133(2):235–255, 1997.

18 H. Liang and T. Maxworthy. An experimental investigation of swirling jets. *Journal of Fluid Mechanics*, 525:115–159, February 2005.

19 Frits H. Post, Benjamin Vrolijk, Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

20 M. Roth. *Automatic Extraction of Vortex Core Lines and Other Line-Type Features for Scientific Visualization*. PhD Thesis, Swiss Federal Institute of Technology, ETH Zürich, 2000.

21 T. Salzbrunn, H. Jänicke, T. Wischgoll, and G. Scheuermann. The State of the Art in Flow Visualization: Structure Based Techniques. In *Proc. Simulation and Visualization (SimVis)*, pages 75–92, 2007.

22 T. Salzbrunn and G. Scheuermann. Streamline Predicates. *IEEE TVCG*, 12(6):1601–1612, 2006.

23 Gerik Scheuermann and Xavier Tricoche. Topological Methods for Flow Visualization. In Charles. D. Hansen and Christopher R. Johnson, editors, *The Visualization Handbook*, pages 341–358. Elsevier, Amsterdam, 2005.

24 Michael Schlemmer, Manuel Heringer, Florian Morr, Ingrid Hotz, Martin Hering-Bertram, Christoph Garth, Wolfgang Kollmann, Bernd Hamann, and Hans Hagen. Moment invariants for the analysis of 2d flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1743–1750, 2007.

25 Cosma Rohilla Shalizi. Optimal nonlinear prediction of random fields on networks. In Michel Morvan and Éric Rémila, editors, *Discrete Models for Complex Systems, DMCS'03*, volume AB of *DMTCS Proceedings*, pages 11–30. Discrete Mathematics and Theoretical Computer Science, 2003.

26 Cosma Rohilla Shalizi, Robert Haslinger, Jean-Baptiste Rouquier, Kristina Lisa Klinkner, and Cristopher Moore. Automatic filters for the detection of coherent structure in spatiotemporal systems. *Physical Review E*, 73:036104, 2006.

27 C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, July 1948.

28 K. Shi, H. Theisel, H. Hauser, T. Weinkauf, K. Matkovic, H.-C. Hege, and H.-P. Seidel. Path Line Attributes - an Information Visualization Approach to Analyzing the Dynamic Behavior of 3D Time-Dependent Flow Fields. In *Topology-Based Methods in Visualization, Proceedings of the 2007 Workshop (to appear)*, 2007.

29 D. Sujudi and R. Haimes. Identification of swirling flow in 3d vector fields. *AIAA 95-1715*, 1995.

CSG Operations of Arbitrary Primitives with Interval Arithmetic and Real-Time Ray Casting

Younis Hijazi[1], Aaron Knoll[2,4], Mathias Schott[3], Andrew Kensler[3], Charles Hansen[3,4], and Hans Hagen[2,4]

1 **IGG-LSIIT, UMR 7005 CNRS, University of Strasbourg, France**
hijazi@unistra.fr
2 **Computergrafics Lab, University of Kaiserslautern, Germany**
knolla@gmail.com, hagen@informatik.uni-kl.de
3 **SCI Institute, University of Utah, US**
{aek,hansen}@cs.utah.edu
4 **International Research Training Group (IRTG) 1131 "Visualization of Large and Unstructured Data Sets, Applications in Geospatial Planning, Modeling and Engineering"**

Abstract

We apply Knoll et al.'s algorithm [9] to interactively ray-cast constructive solid geometry (CSG) objects of arbitrary primitives represented as implicit functions. Whereas modeling globally with implicit surfaces suffers from a lack of control, implicits are well-suited for arbitrary primitives and can be combined through various operations. The conventional way to represent union and intersection with interval arithmetic (IA) is simply using min and max but other operations such as the product of two forms can be useful in modeling joints between multiple objects.

Typical primitives are objects of simple shape, e.g. cubes, cylinders, spheres, etc. Our method handles arbitrary primitives, e.g. superquadrics or non-algebraic implicits. Subdivision and interval arithmetic guarantee robustness whereas GPU ray casting allows for fast and aesthetic rendering. Indeed, ray casting parallelizes efficiently and trivially and thus takes advantage of the continuous increasing computational power of hardware (CPUs and GPUs); moreover it lends itself to multi-bounce effects, such as shadows and transparency, which help for the visualization of complicated objects. With our system, we are able to render multi-material CSG trees of implicits robustly, in interactive time and with good visual quality.

1998 ACM Subject Classification I.3.3 Picture/Image Generation, I.3.5 Computational Geometry and Object Modeling, I.3.7 Three-Dimensional Graphics and Realism

Keywords and phrases Implicit Surface, Constructive Solid Geometry, Interval Arithmetic, Ray Casting

Digital Object Identifier 10.4230/DFU.SciViz.2010.78

1 Introduction

Constructive solid geometry objects involving implicit surfaces can be an effective geometric representation. Arbitrary-form implicit surfaces can be used to model a wide variety of shapes, as well as perform interpolation and smoothing filters of multiple varieties of data. Constructive solid geometry allows for generalized trimming of these surfaces. Moreover, CSG implicits make for a compact and flexible model, in which the CSG object itself can be represented simply by implicit functions consisting of *min* and *max* operators.

Interactive, pixel-exact rendering of implicits poses a challenge to extraction and rasterization methods. Ray casting methods employing interval arithmetic have conventionally been among the most robust solutions for rendering general-form implicit surfaces, but also among the slowest.

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 78–89

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

However, recent SIMD techniques for the CPU [10] and GPU [9] have shown that IA bisection can be a practical method for interactive rendering. The contribution of this paper is to show how, in addition to conventional closed-form implicit functions, interval arithmetic methods can be employed in efficiently rendering constructive solid geometry.

2 Related work

In 1982, Roth [17] presented the first algorithm for directly rendering CSG without precomputing the combined boundary representations. His algorithm used the CSG operators to classify the intersections found by ray casting. Goldfeather et al. [2] showed in 1986 how an initial restructuring of the tree could allow CSG to be directly rendered using Z-buffer rasterization. In 1992 Duff [1] demonstrated the use of IA and subdivision for rendering CSG implicits. Nielson [14] presented applications of implicits and CSG in the context of scattered data interpolation. Kirsch et al. [8] provided an enhancement of Goldfeather's algorithm. Günter et al. [3] performed CSG modeling in real-time while Romeiro et al. [16] focused on large CSG models. Not directly related to CSG, the community Hyperfun [6] builds models using the F-rep representation which includes the CSG one.

3 Background

3.1 Ray casting implicits: a root-finding problem

An *implicit surface S* in $3D$ is defined as the set of solutions of an equation

$$f(x,y,z) = 0 \tag{1}$$

where $f : \Omega \subseteq \mathbb{R}^3 \to \mathbb{R}$. In ray casting, we seek the intersection of a ray

$$\vec{p}(t) = \vec{o} + t\vec{d} \tag{2}$$

with this surface S. By simple substitution of these position coordinates, we derive a unidimensional expression

$$f_t(t) = f(o_x + td_x, o_y + td_y, o_z + td_z) \tag{3}$$

and solve where $f_t(t) = 0$ for the smallest $t > 0$. Therefore ray casting a $3D$ implicit function reduces to a $1D$ root-finding problem.

Approaches for arbitrary implicits include:

- **Closed-form solutions**, which although fast, may suffer from numerical problems in 32-bit float arithmetic.
- **Point-sampling** [4] evaluates the function at interval endpoints and exploits the rule of signs. This is typically fast, but not generally robust (see Fig. 1(a)).
- **Sturm sequences** [18] break the ray segment into monotonic intervals by recursively bracketing zeros of all derivatives. This is slow and requires differentiability.
- **Piecewise algebraic surfaces** [11], though efficient, are limited to low-degree algebraics when relying on an analytical root-finding scheme.
- **Lipschitz methods** [7] which rely on bounding Lipschitz constants to determine where root-finding methods will converge. This works on a subclass of algebraics.
- **Distance functions** [5] require derivation of a signed distance function from an arbitrary point in space to the surface, and also requires Lipschitz.

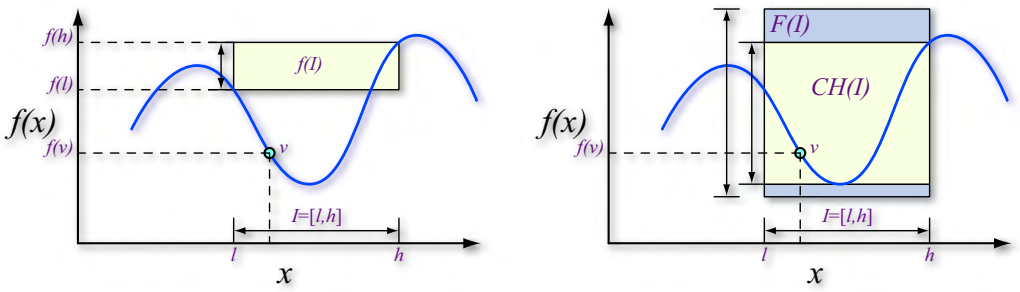


Figure 1 The inclusion property. (a) *Left*: When a function f is non-monotonic on an interval I, evaluating the lower and upper components of a domain interval is insufficient to determine a convex hull over the range. (b) *Right*: This is not the case with an inclusion extension F, which, when evaluated, will enclose all minima and maxima of the function within that interval. Ideally, $F(I)$ is equal or close to the bounds of the convex hull, $CH(I)$.

- **Inclusion algebra methods** which evaluate an inclusion extension of the implicit (see Fig. 1(b)), and use that for spatial rejection or determining monotonicity. These work for any computable function, but require implementation of an inclusion arithmetic library.

This paper will focus on the latter approach, as it is robust and general, and requires nothing more than a function definition. Historically, it has also been the slowest, primarily due to inefficient implementation and impractical numerical assumptions.

3.2 CSG and implicits

The three basic operators in constructive solid geometry are the boolean union, intersection and difference. Considering two solid objects A and B respectively represented by the implicit functions f_A and f_B and with the following convention: $f < 0$ inside the solid and $f > 0$ outside the solid (here $f = 0$ defines the solid), we can easily express those operations in terms of implicit functions. Indeed the union between A and B is defined by

$$A \cup B \;=\; min(f_A, f_B). \tag{4}$$

The intersection between A and B is defined by

$$A \cap B \;=\; max(f_A, f_B). \tag{5}$$

Finally the difference between A and B is defined by

$$A \setminus B \;=\; max(f_A, -f_B). \tag{6}$$

Thus the construction of a complex CSG object using n boolean operators reduces to the expression of a single implicit function formed by *min*, *max*, and the implicit primitives.

3.3 Interval Arithmetic

Interval arithmetic (IA) was introduced by Moore [13] as an approach to bounding numerical rounding errors in floating point computation. The same way classical arithmetic operates on real numbers, interval arithmetic defines a set of operations on intervals. We denote an interval as $\underline{\bar{x}} = [\underline{x}, \bar{x}]$, and the

Algorithm 1 *min* and *max* in IA with Cg.

```
typedef float2 interval;

interval imin(interval a, interval b)
{
  return interval(min(a.x,b.x),min(a.y,b.y));
}

interval imax(interval a, interval b)
{
  return interval(max(a.x,b.x),max(a.y,b.y));
}
```

base arithmetic operations are as follows:

$$\underline{\bar{x}}+\underline{\bar{y}} = [\underline{x}+\underline{y},\bar{x}+\bar{y}], \tag{7}$$

$$\underline{\bar{x}}-\underline{\bar{y}} = [\underline{x}-\bar{y},\bar{x}-\underline{y}], \tag{8}$$

$$\underline{\bar{x}}\times\underline{\bar{y}} = [min(\underline{xy},\underline{x\bar{y}},\underline{\bar{x}y},\overline{\bar{x}\bar{y}}),max(\underline{xy},\underline{x\bar{y}},\underline{\bar{x}y},\overline{\bar{x}\bar{y}})]. \tag{9}$$

Moore's fundamental theorem of interval arithmetic [13] states that for any function f defined by an arithmetical expression, the corresponding interval evaluation function F is an *inclusion function* of f (where F is the interval *extension* of f):

$$F(\underline{\bar{x}}) \supseteq f(\underline{\bar{x}}) = \{f(x) \mid x \in \underline{\bar{x}}\}. \tag{10}$$

The inclusion property provides a robust rejection test, i.e.

$$0 \notin F(\underline{\bar{x}}) \Rightarrow 0 \notin f(\underline{\bar{x}}). \tag{11}$$

Inclusion operations are powerful in that they are composable: if each component operator preserves the inclusion property, then arbitrary compositions of these operators will as well. As a result, in practice *any* computable function may be expressed as inclusion arithmetic [12]. For example, the two IA functions we are mostly interested in for performing CSG are *min* and *max* (see Algorithm 1).

3.4 Ray Casting CSG implicits with IA

The inclusion property extends to multivariate implicits as well, making it suitable for a spatial rejection test in ray casting. Moreover, by substituting the inclusion extension of the ray equation (Equation 2) into the implicit extension $CSG(x,y,z)$, we have a univariate extension $CSG_t(X,Y,Z)$. To check whether any given ray interval $\underline{\bar{t}} = [\underline{t},\bar{t}]$ *possibly* contains our surface, we simply check if $0 \in CSG_t(\underline{\bar{t}})$. As a result, once the inclusion library is implemented, any function composed of its operators can be rendered robustly.

4 Ray Casting CSG implicits with IA on the GPU: results and discussion

Previously we showed how a complex CSG object reduces to a single implicit function. To render these objects efficiently, we turn to the GPU implicit IA bisection algorithm of Knoll et al. [9]. This

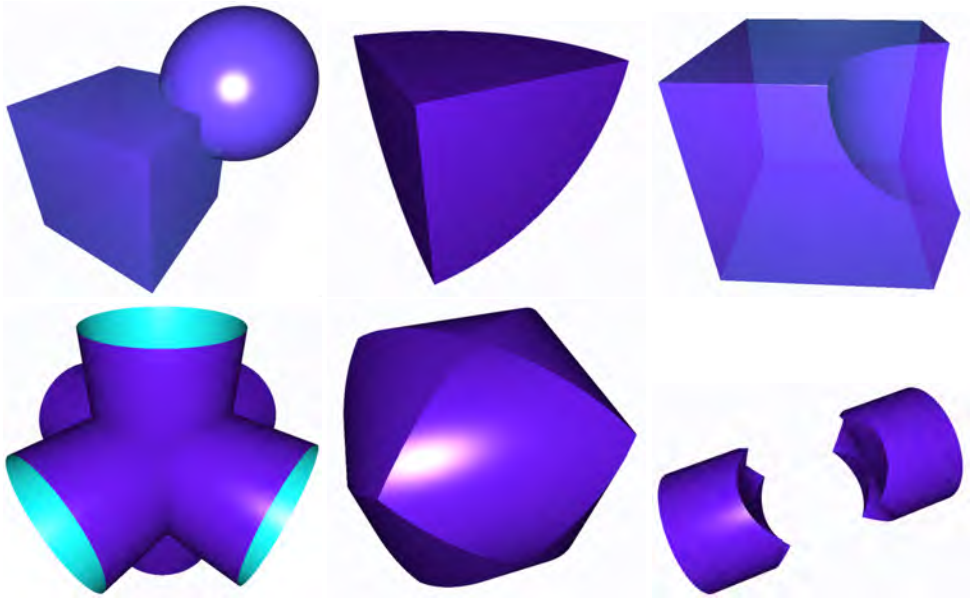


Figure 2 Toy examples. *First row*: union, intersection and difference of a cube and a sphere (20, 160, and 28 fps). *Second row*: union, intersection and difference of three cylinders (91, 84, 127 fps).

method employs simple floating-point modulus to effect a stackless recursion method, bisecting along the ray and computing the interval extension of the implicit function along each bisected segment. The following CSG examples are obtained using this technique with relatively small ε (in the order of $1e-5$). Indeed, when dealing with multiple implicits, a precision of $1e-3$ (typically sufficient for non-CSG objects) is too large for guaranteeing good visual quality, especially around the intersections areas between the primitives (see Section 4.6). All benchmarks are measured in frames per second on an NVIDIA 8800 GTX, at 1024x1024 frame buffer resolution. The equations of the CSG primitives are provided in Table 1 of the Appendix.

4.1 Basic CSG operations

Figure 2 shows a simple example of implicit CSG functionality, using a cube (modeled as a high-order superquadric) and a sphere. We have added transparency in some figures for a better understanding of the resulting object.

4.2 More difficult examples

We can handle implicits defined by arbitrary complicated functions in the same way as simpler forms. Figure 3 demonstrates two more difficult functions: the citrus and the heart. CSG requires that its components be closed manifolds (i.e. without boundary); in other words their combination defines a solid object.

Figure 7 (in the Appendix) demonstrates a panel of CSG objects involving several primitives such as the tangle, the decocube, superquadrics, ellipsoids, etc.

4.3 Arbitrary blending and dynamic CSG

Arbitrary blending: Implicits inherently support blending operations between multiple basis functions. Such forms need only be expressed as an arbitrary $4D$ implicit $f(x,y,z,w)$, where w varies over

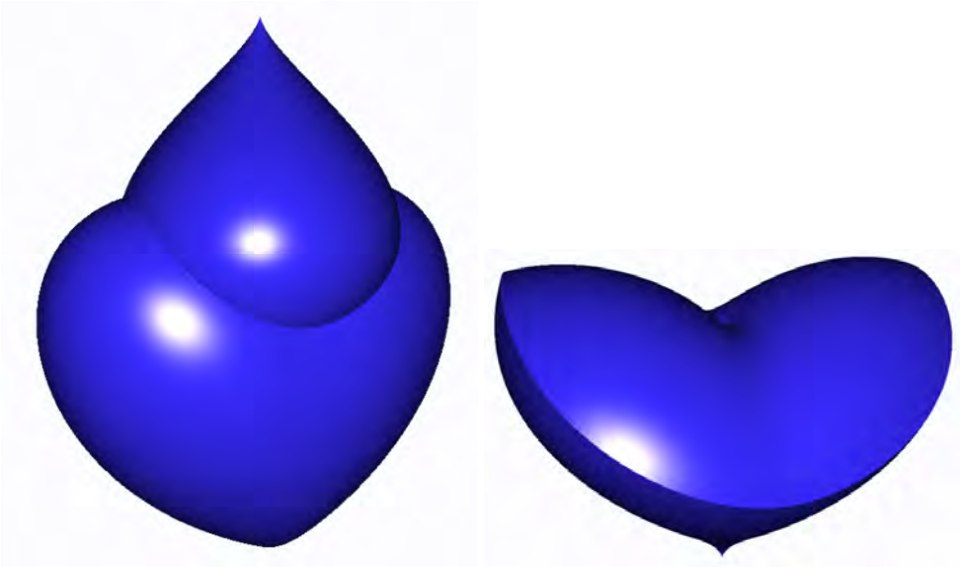


Figure 3 CSG of citrus and heart. *Left*: union (50 fps). *Right*: intersection (48 fps).

time. As ray-casting is performed purely on-the-fly with no precomputation, we have great flexibility in dynamically rendering these functions. Useful morphing methods include product implicits, linear interpolation between surfaces, the hyperbolic and super-elliptic blends; and gaussian or sigmoid blending, shown in Fig. 8 (see Appendix) between the decocube and the sphere. As the blending scheme is also represented as an implicit function in our method, we are able to construct any blend we want.

Dynamic CSG: By setting variables in the CSG objects instead of fixed values, e.g. for a radius, we are able to model time-varying CSG operations. Figure 4 shows a dynamic CSG object: the union of a cube and a radius-varying sphere.

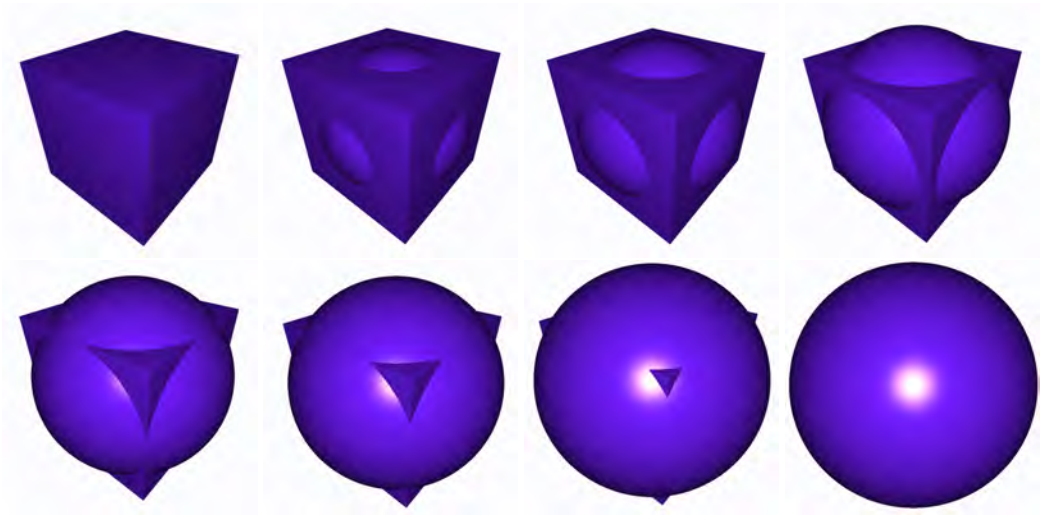


Figure 4 Dynamic CSG: union of a cube and a radius-varying sphere, running at 48-117 fps.

4.4 Multi-material CSG

In addition to using the IA minimum and maximum operators to directly compute the interval extensions of CSG objects, we can evaluate the extensions separately and employ boolean arithmetic to determine which surfaces are intersected by a given ray interval. In addition, we can specify level-set conditions on the individual implicit components, similarly to the CSG methods described in F-rep literature [15]. Given an implicit $f(\omega)$ and a condition $g(\omega)$, inclusion arithmetic allows us to verify $g_+ = \{\underline{g}(\omega) \geq 0\}$ or $g_- = \{\overline{g}(\omega) \leq 0\}$, given the interval form of the inclusion extension G over an interval domain $\omega \subseteq \Omega$. Then, one can render $f \cap g_+$ or $f \cap g_-$ for arbitrary level sets of g. Boolean evaluation of 3-manifold level sets allows us to perform many of the same CSG effects, and at the same time determine which component object is intersected. This allows us to shade components differently as desired (Fig. 5). In addition, increased algorithmic sensitivity near CSG joints due to wider bounds (see Section 4.6) is not an issue using this method.

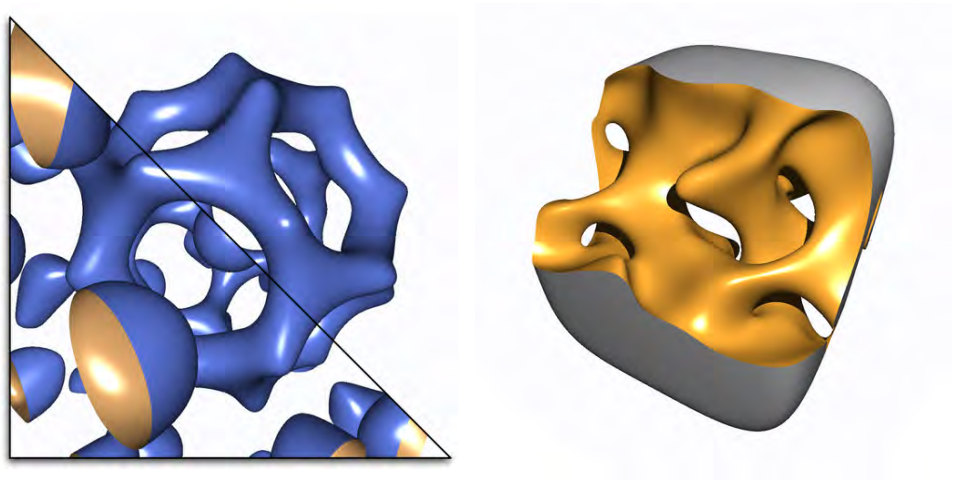


Figure 5 CSG objects using level-set conditions. *Left*: icos.csg (13 fps). *Right*: sesc.csg (9 fps).

4.5 Ray casting effects

As our algorithm relies purely on ray-casting, we can easily support per-pixel lighting models and multi-bounce effects, many of which would be difficult with rasterization (Fig. 6). We briefly describe those modalities.

Transparency: Transparency is useful in visualizing implicits (see Fig. 2 and 6(a)), particularly functions with odd connectivity or disjoint features. It costs around $3\times$ as much as one primary ray per pixel.

Reflections: Reflections are a good example of how built-in features of rasterization hardware can be seamlessly combined with the implicit ray casting system. Looking up a single reflected value from a cubic environment map invokes no performance penalty. Tracing multiple reflection rays in an iterative loop is not significantly more expensive ($20-30\%$), and yields clearly superior results (see Fig. 6(a)(d)).

Gradient shading: Gradient shading is one example of features that can easily be extracted from a ray-casted object; it can help understand its topology. The gradient is computed approximately using central differences. Figure 6(b) shows the gradient shading on an intermediate blend between a decocube and a sphere.

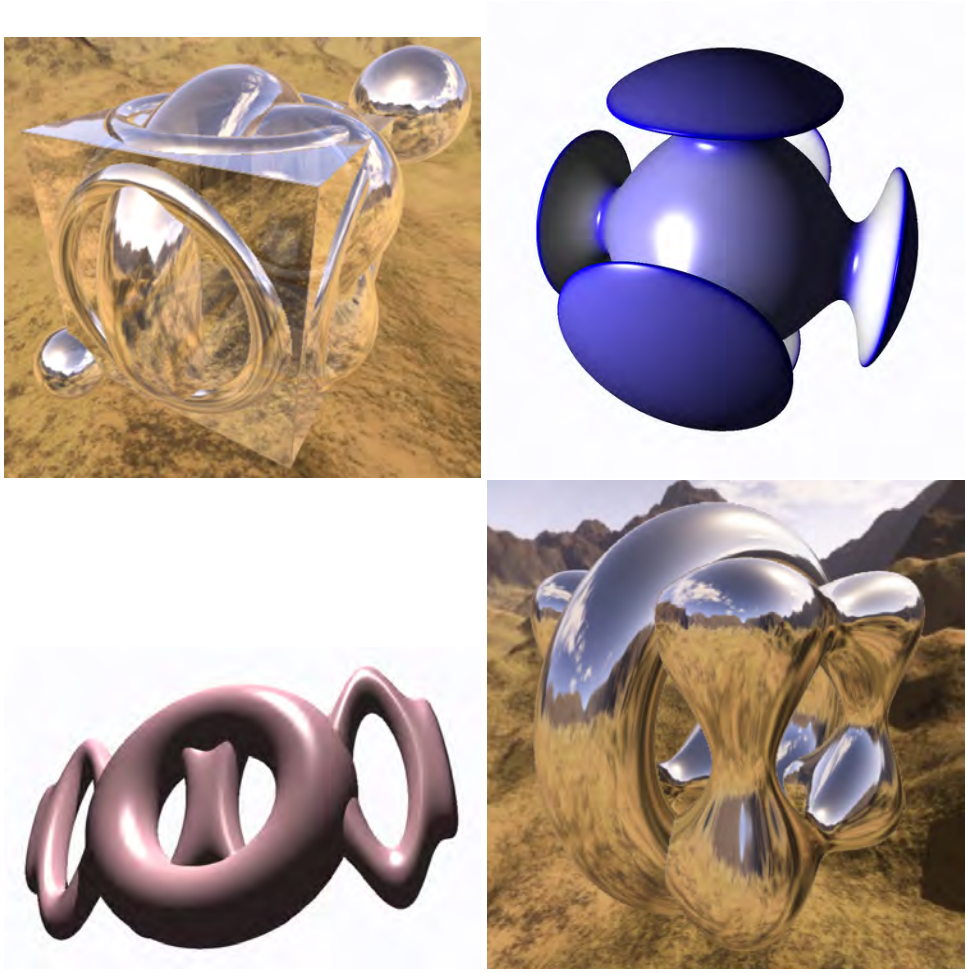


Figure 6 Shading Effects. *Top left to bottom right*: (a) reflections and transparency on multiple-unions CSG object (11 fps); (b) gradient shading on a decocube/sphere blending (41 fps); (c) shadows on *4-Bretzel* ∪ *torus* (30 fps); and (d) *tangle* ∪ *torus* with up to six reflection rays (11.5 fps).

Shadows: Shadows often entail around $20-50\%$ performance penalty. One can equally use a coarser precision for casting shadow rays than primary rays. An example of shadows is illustrated in Fig. 6(c).

4.6 Algorithmic Sensitivity

Much efficiency of the IA bisection technique is owed to the fact that fairly low sensitivity is required for accurate rendering. For many implicit forms without CSG, a termination criterion such as $\varepsilon = 2^{-11} \approx 0.0005$ is sufficient for accurate rendering. However, in the case of CSG objects, the use of IA minimum and maximum operators cause local bounds to expand, particularly near joints. As a result, a finer discretization is required by our rendering technique to reconstruct the correct surface. Generally, this requirement is not significantly greater ($\varepsilon = 2^{-16} \approx 1e-5$ typically suffices); however this constraint is view-dependent as well as dependent on the form of the implicit itself. Nonetheless, we find IA ray bisection is less sensitive to CSG joints than to fine features in the implicit itself (for example the asymptotic features of the Steiner surface shown in [9]). Moreover, despite the moderately finer ε required to render CSG objects, this sensitivity has little impact on the frame rate (perhaps 10%-20%) compared to the costs of additional IA computation. We note that

greater algorithmic sensitivity is not an issue for multi-material objects computed using the boolean evaluation method of Section 4.4.

5 Conclusions and Future Work

We have demonstrated a system which can render multi-material CSG objects of implicits robustly, in interactive time and with good visual quality. Moreover we can add multi-bounce effects, such as shadows and transparency, which help for the understanding of complicated objects. Our system is general: it handles arbitrary primitives; robust: it relies on robust techniques; and efficient: it exploits recent GPU's capabilities.

There are several directions for future work. One desirable direction would be to develop a CSG language similar to [6] and adapt the existing GUI to be able to model large multi-material CSG objects. Extending the ray casting system with a bounding volume hierarchy traversal would allow for a scene graph of piecewise implicit primitives for use in modeling or visualization, and would accelerate rendering. Also comparing interval and (reduced) affine arithmetic as in [9] for the task of CSG modeling may lead to interesting observations. Another direction would be to work on the interaction paradigm of the system so that the user could intuitively build primitives, including free-form surfaces using control points. Using this system to prototype trimmed moving least squares implicits, for example, would be an interesting application. Finally, a virtual reality environment would be perfectly well-suited for such a direct-interaction CSG modeling system.

References

1 Tom Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 131–138, New York, NY, USA, 1992. ACM Press.

2 Jack Goldfeather, Jeff P M Hultquist, and Henry Fuchs. Fast constructive-solid geometry display in the pixel-powers graphics system. *SIGGRAPH Comput. Graph.*, 20(4):107–116, 1986.

3 Brian Guenter and Marcel Gavriliu. Exact procedural csg modeling for real time graphics. Technical Report at Microsoft Research.

4 Pat Hanrahan. Ray tracing algebraic surfaces. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 83–90, New York, NY, USA, 1983. ACM Press.

5 J. C. Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

6 HyperFun team. HyperFun Project. http://www.hyperfun.org/.

7 D. Kalra and A. H. Barr. Guaranteed ray intersections with implicit surfaces. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 297–306, New York, NY, USA, 1989. ACM Press.

8 Florian Kirsch and Jürgen Döllner. Rendering techniques for hardware-accelerated image-based csg. In *Journal of WSCG*, pages 221–228, 2004.

9 Aaron Knoll, Younis Hijazi, Andrew Kensler, Mathias Schott, Charles D. Hansen, and Hans Hagen. Fast ray tracing of arbitrary implicit surfaces with interval and affine arithmetic. *Comput. Graph. Forum*, 28(1):26–40, 2009.

10 Aaron Knoll, Younis Hijazi, Ingo Wald, Charles Hansen, and Hans Hagen. Interactive ray tracing of arbitrary implicits with simd interval arithmetic. In *Proceedings of the 2nd IEEE/EG Symposium on Interactive Ray Tracing*, pages 11–18, 2007.

11 Charles Loop and Jim Blinn. Real-time GPU rendering of piecewise algebraic surfaces. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 664–670, New York, NY, USA, 2006. ACM Press.

12 Don Mitchell. Robust ray intersection with interval arithmetic. In *Proceedings on Graphics Interface 1990*, pages 68–74, 1990.

13 R. E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, NJ, 1966.

14 Gregory M. Nielson. Radial hermite operators for scattered point cloud data with normal vectors and applications to implicitizing polygon mesh surfaces for generalized csg operations and smoothing. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 203–210, Washington, DC, USA, 2004. IEEE Computer Society.

15 Alexander A. Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

16 Fabiano Romeiro, Luiz Velho, and Luiz Henrique de Figueiredo. Hardware-assisted rendering of csg models. In *XIX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'06)*, pages 139–146, 2006.

17 Scott D. Roth. Ray Casting for Modeling Solids. *Computer Graphics and Image Processing*, 18(2):109–144, 1982.

18 J.J. van Wijk. Ray tracing objects defined by sweeping a sphere. *Computers & Graphics*, 9:283–290, 1985.

A Equations of the implicit primitives

Table 1 Formulas of the CSG primitives.

sphere	$x^2+y^2+z^2-r^2$
pseudo-cube	$x^{500}+y^{500}+z^{500}-r^2$
cylinder	x^2+y^2-1
torus	$(1-\sqrt{(x^2+y^2)})^2+z^2-.125$
4-bretzel	$\frac{1}{10}(x^2(1.21-x^2)^2(3.8-x^2)^3-10y^2)^2+60z^2-2$
tangle	$x^4-rx^2+y^4-5y^2+z^4-5z^2+11.8$
decocube	$((x^2+y^2-0.8^2)^2+(z^2-1)^2)((y^2+z^2-0.8^2)^2+$ $(x^2-1)^2)((z^2+x^2-0.8^2)^2+(y^2-1)^2)-0.02$
superquadric	$x^{200}+(.5y^4+.5z^4)^4-1$
ellipsoid	$.25x^2+.25y^2+z^2-1$
heart	$(2x^2+y^2+z^2-1)^3-(.1x^2+y^2)z^3$
citrus	$x^2+z^2-4y^3(1-.5y)^3$
trigonometric	$(1-\sqrt{(x^2+y^2)})^2+\sin(z)^3-.125$
icos.csg	$ic(x,y,z)=2-(\cos(x+\tau y)+\cos(x-\tau y)+\cos(y+\tau z)+$ $\cos(y-\tau z)+\cos(z-\tau x)+\cos(z+\tau x))\,,\quad \tau=\frac{1+\sqrt{5}}{2}$ CSG condition (on inclusion intervals): $(0\in ic)$ and $sphere_{inner}<0$ and $sphere_{outer}>0$
sesc.csg	CSG of superellipsoid (se) and sinusoid convolution (sc): $se(x,y,z)=x^6+\frac{1}{2}(y^4+z^4)^4-20$ $sc(x,y,z)=xy+\cos(z)+1.741\sin(2x)\sin(z)\cos(y)+\sin(2y)\sin(x)\cos(z)$ $+\sin(2z)\sin(y)\cos(x)-\cos(2x)\cos(2y)$ $+\cos(2y)\cos(2z)+\cos(2z)\cos(2x)+0.05$ CSG condition (on inclusion intervals): $((sc>0)$ and $(0\in se))$ or $((se<0)$ and $(0\in sc))$
multiple-unions csg	$\min(\min(\min(\min(\min(x^{500}+y^{500}+z^{500}-.25,$ $(x-1)^2+(y-1)^2+(z-1)^2-.2),((x^2+y^2-0.8^2)^2+(z^2-1)^2)$ $((y^2+z^2-0.8^2)^2+(x^2-1)^2)((z^2+x^2-0.8^2)^2+(y^2-1)^2)-0.02),$ $(2x^2+y^2+z^2-1)^3-(.1x^2+y^2)z^3),$ $(1-\sqrt{(x^2+y^2)})^2+z^2-.125),(x+1)^2+(y+1)^2+(z+1)^2-.1)$

B More examples of CSG implicits

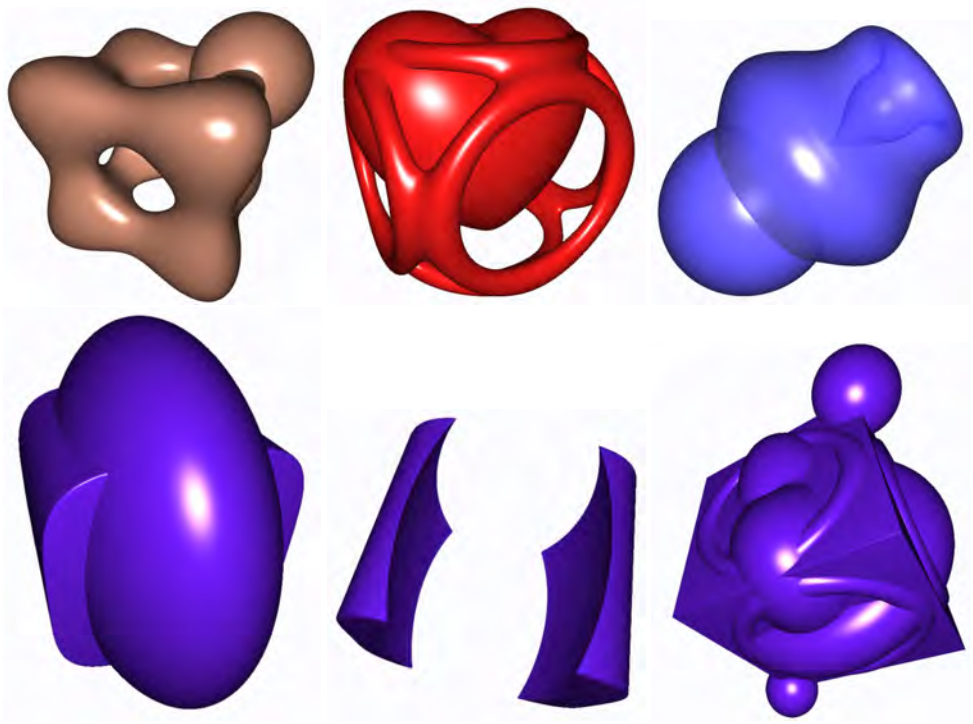


Figure 7 CSG with arbitrary primitives. *First row*: *tangle* ∪ *sphere* (12.7 fps), *decocube* ∪ *heart* (22 fps) and *trigonometric function* ∪ *sphere* (16 fps). *Second row*: *superquadric* ∪ *ellipsoid* (41 fps), *superquadric* \ *ellipsoid* (60 fps) and multiple-unions CSG object (21 fps).

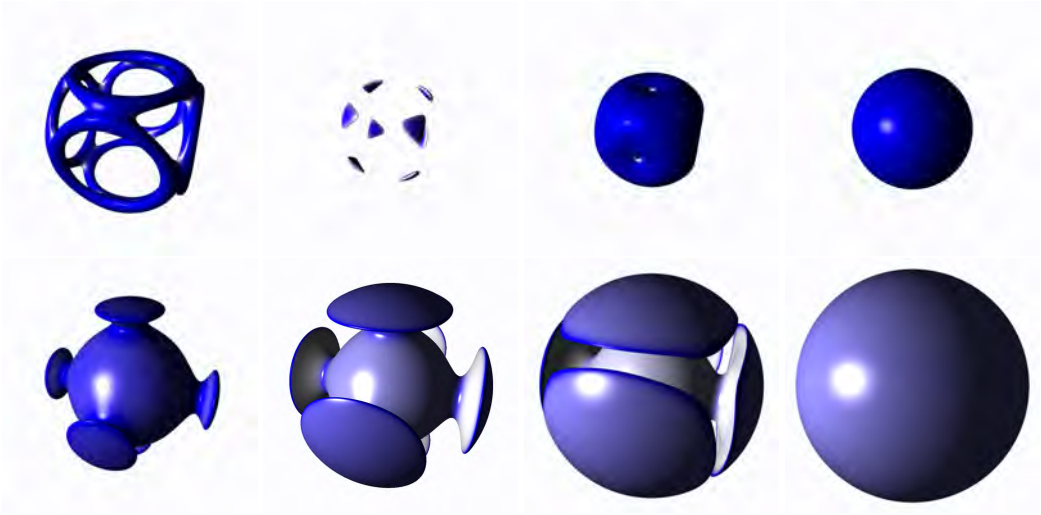


Figure 8 4*D* sigmoid blending of the decocube and a sphere running at 33 – 50 fps.

Exploring Visualization Methods for Complex Variables

Andrew J. Hanson[1] and Ji-Ping Sha[1]

1 **Computer Science Department and Mathematics Department**
Indiana University
Bloomington, IN 47405 USA
{hansona,jsha}@indiana.edu

Abstract

Applications of complex variables and related manifolds appear throughout mathematics and science. Here we review a family of basic methods for applying visualization concepts to the study of complex variables and the properties of specific complex manifolds. We begin with an outline of the methods we can employ to directly visualize poles and branch cuts as complex functions of one complex variable. $\mathbb{C}\mathbf{P}^2$ polynomial methods and their higher analogs can then be exploited to produce visualizations of Calabi-Yau spaces such as those modeling the hypothesized hidden dimensions of string theory. Finally, we show how the study of N-boson scattering in dual model/string theory leads to novel cross-ratio-space methods for the treatment of analysis in two or more complex variables.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, J.2 Physical Sciences and Engineering

Keywords and phrases Visualization, Complex Manifolds, High Dimensions

Digital Object Identifier 10.4230/DFU.SciViz.2010.90

1 Introduction

Mathematical visualization of issues involving complex variables is a fundamental problem that, sooner or later, is related to almost any problem in science. Our goal here is to review some general methods that can be used to make the abstract features of complex variables more concrete by exploiting computer graphics technology, and to illustrate these methods with some interesting applications. We begin with a number of general concepts, and conclude with some examples related to problems of mathematical physics motivated by string theory.

The basic methods for the representation of the shapes of homogeneous polynomial equations in $\mathbb{C}\mathbf{P}^2$ were explored in detail in ([3]), and this will be the starting point for many of our basic visualizations. We will also briefly summarize some more recent results of ([4]) treating some geometric objects arising naturally in the complex analysis of integrals appearing in the N-boson scattering amplitudes of the dual models of early string theory.

2 Visualizing Complex Analysis

Complex Numbers

We may think of a complex number in several ways. The most traditional form comes from the observation that, while the trivial equation $x^2 = 1$ can be solved in the domain of real numbers, the closely related equation $x^2 = -1$ cannot: one must introduce an "imaginary

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 90–109

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

number" obeying $i^2 = -1$ in order to be able to represent the solutions to all algebraic equations of a single variable.

The most general form of the solution to an algebraic equation in one variable thus has two parts, a real part and an imaginary part, which can be written in terms of two real numbers x and y as

$$z = x + iy \ . \tag{1}$$

We also introduce the complex conjugation operation,

$$\overline{z} = x - iy \ , \tag{2}$$

which in turn leads to the concept of the modulus-squared,

$$z\overline{z} \equiv |z|^2 = x^2 + y^2 \ . \tag{3}$$

The essential properties of products of complex numbers follow directly from the properties of the symbol i, yielding

$$z_1 z_2 = (x_1 + iy_1)\,(x_2 + iy_2) = (x_1 x_2 - y_1 y_2) + i\,(x_1 y_2 + x_2 y_1) \ . \tag{4}$$

A more formal way of writing this would be to consider Eq. (4) as a realization of an abstract algebra relating pairs of numbers, where the corresponding (commutative, associative) algebra is defined as

$$(x_1,\, y_1) \star (x_2,\, y_2) = (x_1 x_2 - y_1 y_2,\, x_1 y_2 + x_2 y_1) \ . \tag{5}$$

Equations (4) and (5) are indistinguishable in any mathematical sense, though some practitioners may feel strongly about being more comfortable with one or the other.

For completeness, we note that another unique property of complex numbers is that, besides the trivial case of real multiplication, only complex multiplication is both commutative and preserves the value of the modulus under multiplication,

$$|z_1 z_2| = |z_1|\,|z_2| \ . \tag{6}$$

2.0.0.1 Visualizing a Complex Point

Once we have Eq. (1), we may ask immediately how we visualize a complex point. One approach is that of Figure 1(a), which simply treats x and y as Cartesian variables, and so every complex number is depicted as a point in the 2D plane. However, this does not allow us to easily treat infinity, which is a critical element in the mathematical analysis of functions of a complex variable. Thus Figure 1(a) is only a local view of the actual manifold that mathematicians refer to as "the complex line" because of its one-dimensional complex nature, and that physicists and engineers, for example, would refer to as the "complex plane" because of its two-dimensional real nature. In order to treat the space of one complex variable in a way that infinity is no longer a special point, and can be included naturally in all the tasks of complex analysis, we must find a way to express coordinates on the space in a way that is more general than simple Cartesian coordinates. The solution to this problem is to treat the representation of one complex variable using *one-dimensional complex projective space* or $\mathbb{C}\mathbf{P}^1$, which is the space of *pairs* of complex numbers $(z_0,\, z_1)$ that are taken to be equivalent under multiplication by any nonvanishing complex number λ, which is to say

$$(z_0,\, z_1) \sim (\lambda z_0,\, \lambda z_1) \ . \tag{7}$$

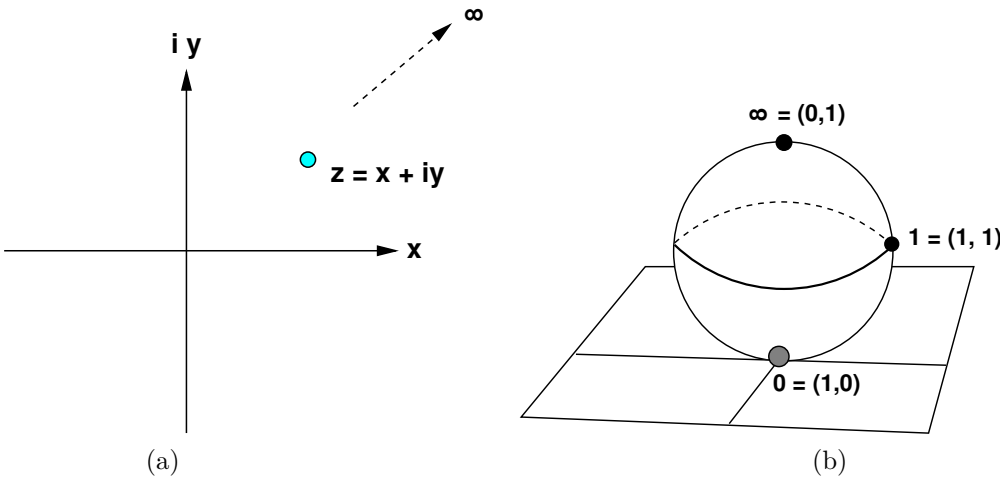


Figure 1 (a) The complex plane. (b) The full space of one complex variable, the one-dimensional complex projective space $\mathbb{C}\mathbf{P}^1$, which is topologically the same as an ordinary sphere, and is thus also known as the Riemann sphere.

Note that this is a two-ended ray of equivalences, since λ may take either sign. We see that in Figure 1(a) we have chosen, e.g., the local coordinates $z_0 = 1$ and $z = z_1/z_0 = z_1$. The point at ∞ now has a precise realization as the coordinate that results when we let $z_0 \to 0$; however, in the context of complex projective space, we never allow this to happen, since we can always write "infinity" as the finite homogenous pair $(0, z_1)$. There are thus essentially two patches in the coordinate system, one where z_0 is allowed to be zero, but not z_1, so the coordinate system is $(z, 1)$, and a second where z_1 is allowed to be zero, but not z_0, so the coordinate system is $(1, z)$. The origin of each of these coordinate systems is the infinity of the other, and the two coordinate systems at all other points are related by multiplication by z_1/z_0 or by z_0/z_1.

When we make *local pictures*, therefore, we must choose one of these two coordinate systems, and accept that we cannot draw at infinity until we change coordinate systems. The entire complex plane thus consists of two parts:

- **North pole.** ($z_1 = 0 \to$ OK, but $z_0 \neq 0$).
- **South Pole.** ($z_0 = 0 \to$ OK, but $z_1 \neq 0$).

When we patch these two neighborhoods together around the equator, we find that the result is a topological sphere, so $\mathbb{C}\mathbf{P}^1 \sim \mathbf{S}^2$, as shown schematically in Figure 1(b).

2.0.0.2 Fixing the coordinate system

Complex projective space admits a standard group of transformations, $\mathbf{PGL}(2, \mathbb{C})$ or the linear-fractional transformations, that parameterize all possible transformations of the coordinate system on $\mathbb{C}\mathbf{P}^1$. The specific transformation on the homogeneous coordinates can be written using the $\mathbf{PGL}(2, \mathbb{C})$ matrix elements

$$[M] = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} , \tag{8}$$

with $\det M \neq 0$, as

$$\begin{bmatrix} z_0' \\ z_1' \end{bmatrix} = M \cdot \begin{bmatrix} z_0 \\ z_1 \end{bmatrix}$$

or

$$(z_0',\, z_1') = (\alpha z_0 + \beta z_1,\, \gamma z_0 + \delta z_1) \ , \tag{9}$$

in the homogeneous, ray-equivalent coordinates, or as

$$z' = \frac{\alpha z_0 + \beta z_1}{\gamma z_0 + \delta z_1} \tag{10}$$

in the $\gamma z_0 + \delta z_1 \neq 0$ set of inhomogeneous coordinates.

The group of linear fractional transformations thus has three free complex parameters that can be used to map any three complex points in the complex plane to any chosen points to fix the degrees of freedom under the map. As illustrated in Figure 1(b), these are conventionally chosen in the following way:

- "0" is projective $(1, 0)$,
- "1" is projective $(1, 1)$, and
- "∞" is projective $(0, 1)$.

2.1 Cross Ratios and Cross-Ratio Coordinates

An important feature of complex projective space is that there is a family of *invariants* under the linear fractional transformations (9) known as the *cross ratios*, defined as follows:

$$u(w, x, y, z) = \frac{(w - y)}{(w - z)} \Bigg/ \frac{(x - y)}{(x - z)} = \frac{(w - y)(x - z)}{(w - z)(x - y)} \ . \tag{11}$$

In particular, one can verify that there are two distinct cross ratios of four variables,

$$u_1 \quad = \quad u(z, z_1, z_0, z_2) = \frac{(z - z_0)(z_1 - z_2)}{(z - z_2)(z_1 - z_0)} \tag{12}$$

$$u_2 \quad = \quad u(z_1, z_2, z, z_0) = \frac{(z_1 - z)(z_2 - z_0)}{(z_1 - z_0)(z_2 - z)} \tag{13}$$

that are related by the constraint

$$1 = u_1 + u_2 \ . \tag{14}$$

Since there are three remaining complex degrees of freedom in the $\mathbf{PGL}(2, \mathbb{C})$ matrix $[M]$ after accounting for projective equivalence, we can exhaust those degrees of freedom by *choosing a coordinate system* on $\mathbb{C}\mathbf{P}^1$ that fixes three complex points. This fact ties in with the definition of the group-invariant cross ratios because it allows us to fix three of the variables in the cross ratio to be, for example, 0, 1, and ∞, thus fixing

$$u_1 \quad = \quad u(z, 1, 0, \infty) = z \tag{15}$$

$$u_2 \quad = \quad u(1, \infty, z, 0) = 1 - z \ . \tag{16}$$

2.1.0.3 Cross-ratio space

However, even this is not the whole story. As pointed out in ([4]), from Eq. (14) we can deduce the existence of yet another projective space, the *cross-ratio space*, which results from creating a new set of homogeneous coordinates, this time in $\mathbb{C}\mathbf{P}^2$, by realizing that we must add a third variable, u_0, to Eq. (14) to make it homogenous:

$$u_0 = u_1 + u_2 \ . \tag{17}$$

This equation can be solved projectively in three different sets of variables, corresponding to choosing the local coordinates $u_0 = 1$, $u_1 = 1$, or $u_2 = 1$, and three intervals in inhomogeneous coordinates as $A = [0, 1]$, $B = [1, \infty]$, and $C = [-\infty, 0]$. The triples of variables solving the constraint equation (17) can then be written

$$\begin{aligned} A(t): &\quad [1,\, t,\, (1-t)] \\ B(t): &\quad [(1-t),\, 1,\, -t] \\ C(t): &\quad [-t,\, (1-t),\, -1]\ . \end{aligned} \tag{18}$$

The variables of region A solve $1 = u_1 + u_2$ with $u_1 = t$, B solves $1 = u_1 + u_2$ with $u_1 = 1/(1-t)$ when all is multiplied by $(1-t)$, and C solves $1 = u_1 + u_2$ with $u_1 = (t-1)/t$ when all is multiplied by t. We note that $C(1) = -A(0)$, so that in fact we have a double covering of the constraint space: the constraint equation solutions must be adjoined to their negatives to form a piecewise continuous curve in $\mathbb{C}\mathbf{P}^2$.

This concludes our introduction to the basic concepts we need to build various visualizations related to a single complex variable. Next we work out some examples in complex analysis.

2.2 Visualizing a Simple Pole

The simplest example of a complex function is a constant function,

$$z = a + ib.$$

Choosing a particular local $\mathbb{C}\mathbf{P}^1$ coordinate system allows us to plot this as a point in a plane as in Figure 1(a). However, even this is not quite as simple as it looks. First we recall that ordinary real graphs of functions are written as

$$y = f(x)\ ,$$

so that we use one space dimension to graph the value of the independent variable and a second one to graph the result. Thus the correct complex analog would involve *two* complex variables: z describing the value of the independent variable, and, say,

$$w = f(z) = \mathrm{Re}\, f(z) + i\, \mathrm{Im}\, f(z)$$

to describe the (complex) result of evaluating the function. Thus, we might consider the graphing process to be described more clearly using two variables, $z_1 = x_1 + iy_1$ and $z_2 = x_2 + iy_2$, where

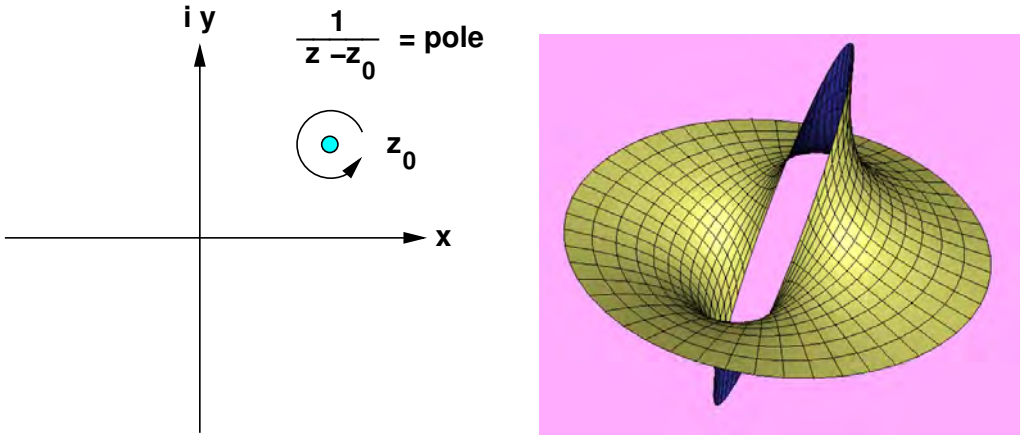
$$z_2 = f(z_1)\ . \tag{19}$$

We can easily see how this works with the classic example of a simple pole at the origin,

$$\frac{1}{z} = \frac{x - iy}{x^2 + y^2}\ .$$

Using the two-variable form, we find that the result involves four variables, (x_1, y_1, x_2, y_2), and one complex or two real equations, so the shape described is a surface with components given by the real and imaginary parts of the following:

$$z_2 = x_2 + iy_2 = \frac{1}{z_1} = \frac{x_1 - iy_1}{x_1^2 + y_1^2}\ . \tag{20}$$



(a) (b)

Figure 2 (a) Conventional picture of a complex pole $f(z) = 1/(z - z_0)$ at z_0, indicating the positive sense of a contour to pick up the residue. (b) Visualizing the geometric shape of a complex pole $w = 1/z$ showing $\mathrm{Re}\, w$ as a function of z. The imaginary part looks basically the same.

The result must be projected from 4D to 3D to be rendered using standard graphics methods. In Figure 2, we show the location of a general pole $f(z) = 1/(z - z_0)$ using a textbook 2D complex analysis plot, and then show the visualization of the complex surface corresponding to the pole using $\mathrm{Re}\, z_2 = x_2$ as the third axis.

Remark: The analysis of a pole typically involves one more step, namely the description of a circular contour integral surrounding the pole. From the classic theorems of complex analysis, this integral

$$\int_{\text{closed circle}} \frac{dz}{z}$$

vanishes if the contour does not enclose the pole, and has the constant value $2\pi i$ as long as the contour encloses the pole. The proof is trivial in polar coordinates with $z = r\exp(i\theta)$:

$$\int_{\text{circle with radius } r} \frac{dz}{z} = \int_0^{2\pi} \frac{ire^{i\theta}d\theta}{re^{i\theta}} = 2\pi i \ .$$

2.3 Integrating with Branch Cuts

Moving on from simple poles, we next examine functions with multiple roots, and hence multiple branches of the Riemann surfaces that are needed to precisely define the functions. The square root already has ample complexity to challenge our visualization technology. If we consider the contour integral of the function $w = \sqrt{1 - z^2}$ along a path that passes around $z = +1$, we find the standard textbook drawing in Figure 3 describing the integral

$$\int_{a+b} dz\, \sqrt{1 - z^2} \ .$$

The main characteristic distinguishing a branch cut in analysis is that, while, e.g., the phase of the function changes by a full (2π) along a path going around a pole, it changes by a precise fraction, namely $2\pi/n$, along a path going from one side of an n-th root branch point to the other. Thus, for example, the relative phase between the integrand on the path of the

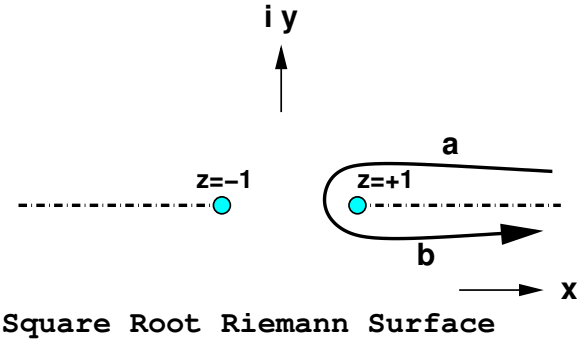


Figure 3 Complex contour integral around the square-root branch point of $\sqrt{1-z^2}$ at $z=+1$.

a branch and the path of the b branch for the square root branch cut shown in Figure 3 is

$$e^{2\pi i/2} = -1 \ .$$

Here, once again, we need to go beyond conventional diagrams such as Figure 3 to create a useful visualization. One approach to functions with multiple branch points is to realize that the Riemann surface to be displayed should not just represent a single branch, e.g.,

$$w = +\sqrt{1-z^2} \tag{21}$$

or

$$w = -\sqrt{1-z^2} \ , \tag{22}$$

but should represent *all* branches. With a little thought we can see that everything is summarized nicely in the equation

$$w^2 + z^2 = 1 \ . \tag{23}$$

There are many different ways to plot this surface (remember, 4 real variables, with 2 real equations means it is a surface), including just using the separate pieces from Eqs. (21) and (22) directly. We typically prefer the methods introduced in ([3]), which will be described shortly, and which recreate the entire surface from a very simple fundamental domain via complex phase transformations around the fixed points of the surface. The basic problem goes back once again to the difference between homogeneous and inhomogeneous coordinates: we should really be looking at Eq. (23) as a homogeneous equation in $\mathbb{C}\mathbf{P}^2$ of the form

$${z_0}^2 + {z_1}^2 + {z_2}^2 = 0 \ . \tag{24}$$

Choosing any one of the $\mathbb{C}\mathbf{P}^2$ variables (z_0, z_1, z_2) to be a constant ($z_0 = i$ is just as good as $z_0 = 1$) gives a partial shape that does not include infinity (where the constant variable vanishes, e.g., $z_0 \to 0$). Thus we are left with holes in the surface that are represented as rings that go off to infinity when we plot the surface using a local pair of inhomogeneous variables as in Figure 4. The square root branch points and cuts can be explicitly seen in the projection of Figure 4 as the ending points of the X-shaped crossings.

2.4 Visualizations of Homogeneous Polynomials in $\mathbb{C}\mathbf{P}^2$

The square root Riemann surface is a special case of a general family of polynomials that are of interest. Here we review the properties and visualization methods for the simplest

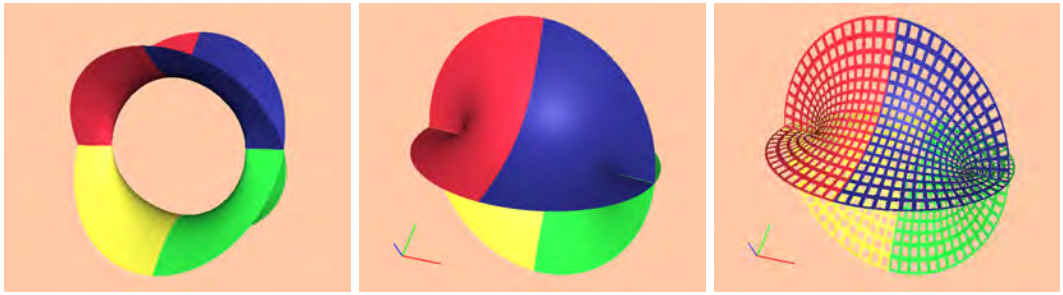


Figure 4 Assorted views of the full square-root Riemann surface with Re w projected to the 3rd axis. The surface is a topological sphere, but the local inhomogeneous coordinates obscure that fact since there are two rings going off to the surface at infinity. The inner ends of the X-shaped crossings are the branch points.

homogeneous polynomials that arise in the study of $\mathbb{C}\mathbf{P}^2$. Starting from the n-th root of a polynomial of one complex variable with zeros at the n roots of unity, that is

$$w = (1 - z^n)^{1/n} \tag{25}$$

and following the same procedure as for the square root, we arrive at the corresponding homogeneous polynomial in $\mathbb{C}\mathbf{P}^2$:

$$z_0{}^n + z_1{}^n + z_2{}^n = 0 \ . \tag{26}$$

As we have noted, there are a variety ways to solve this equation, including:

- n **roots:** Set $z_0{}^n = -1$ and solve for the n roots of $(1 - z^n)^{1/n}$ (which are found by multiplying by a phase $\exp(2\pi ik/n)$, $k = 0, \ldots, n-1$).
- **Spinor variables:** Parameterize the solution using the variables typically used to define null spinors, $w_0{}^2 + w_1{}^2 + w_2{}^2 = 0$ ([1]):
  $$\begin{aligned} z_0 &= \left(i(x^2+y^2)\right)^{2/n} \\ z_1 &= \left(x^2-y^2\right)^{2/n} \\ z_2 &= (2xy)^{2/n} \ . \end{aligned}$$
  Because of the projective equivalence, only n^2 of the n^3 phase choices available here are meaningful.
- n^2 **roots:** In the method of ([3]), the relative phases of n^2 different congruent patches tie together to create the full topological surface, where the obvious locations of the fixed points of the z_1 and z_2 phase transformations expose many key features of the surface. The method starts as before by setting $z_0{}^n = -1$. Then we exploit the complex trigonometric identity
  $$\cos(\theta + i\xi)^2 + \sin(\theta + i\xi)^2 = 1$$
  to define one patch, the fundamental domain, as the quadrant where θ gives a positive real part for cos and sin, namely $0 \le \theta \le \pi/2$ and $-\xi_{\max} \le \xi \le +\xi_{\max}$. Thus the first and most elementary of the n^2 patches is
  $$\begin{aligned} w_1 &= (\cos(\theta + i\xi))^{2/n} \\ w_2 &= (\sin(\theta + i\xi))^{2/n} \ . \end{aligned}$$
  The remaining patches are found by making phase transformations on both z_1 and z_2 until the entire surface is covered; all the patches are then labeled by the n^2 integer pairs

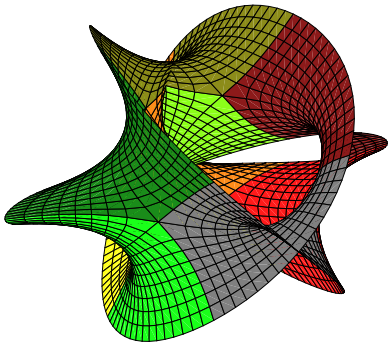


Figure 5 $n = 3$ in $\mathbb{C}\mathbf{P}^2$: The cubic is a torus.

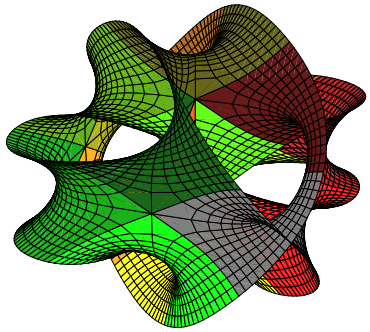


Figure 6 $n = 4$ in $\mathbb{C}\mathbf{P}^2$: The quartic is a section of the **K3** surface, a 4-manifold.

(k_1, k_2), where $k_1 = 0, \ldots, n-1$, $k_2 = 0, \ldots, n-1$, and the parametric solutions of the equations become

$$\begin{aligned} z_1 &= \exp(2\pi i k_1/n) w_1 \\ z_2 &= \exp(2\pi i k_2/n) w_2 \ . \end{aligned}$$

Typical results are shown in the Figures as follows:

- **Cubic Torus.** The cubic is topologically a torus (genus 1), though it is hard to see due to the infinities in local coordinates. It is also technically the standard polynomial in $\mathbb{C}\mathbf{P}^2$ that is a Calabi-Yau space. See Figure 5.
- **Slice of K3 Quartic.** K3 is described by the quartic polynomial of complex dimension 2 (4 real dimensions) in $\mathbb{C}\mathbf{P}^3$. This is the *unique simply-connected* Calabi-Yau 4-manifold, and we can write the equation locally as

$$(z_1)^4 + (z_2)^4 + (z_3)^4 = 1 \ . \tag{27}$$

This is one complex constraint in 3D complex space, and thus is a manifold with 2 complex, 4 real, dimensions. Setting, e.g., $z_3 = 0$, gives a slice that is a surface in $\mathbb{C}\mathbf{P}^2$ with genus 3. See Figure 6.

- **Slice of Calabi-Yau Quintic.** It is hypothesized that 10-dimensional string theory includes 4 dimensions of space-time and 6 dimensions that are curled up into a Calabi-Yau space at the scale of the Planck length. A popular (but by no means unique) candidate for this space is the quintic in $\mathbb{C}\mathbf{P}^4$ given locally by the equation

$$(z_1)^5 + (z_2)^5 + (z_3)^5 + (z_4)^5 = 1 \ . \tag{28}$$

This is one complex constraint in 4D complex space, and thus is a manifold with 3 complex, 6 real, dimensions. Setting, e.g., $z_3 = z_4 = 0$, gives a slice that is a surface in $\mathbb{C}\mathbf{P}^2$ with genus 6. See Figure 7.

In general, it can be shown that every homogeneous polynomial of degree $N + 1$ in $\mathbb{C}\mathbf{P}^N$ is in fact a Calabi-Yau space and therefore admits a Ricci-Flat metric. Calabi conjectured and Yau proved the existence of these metrics ([8]), but, except for trivial cases such as the $\mathbb{C}\mathbf{P}^2$ cubic torus, none are explicitly known. In Table 1, we summarize this family of spaces.

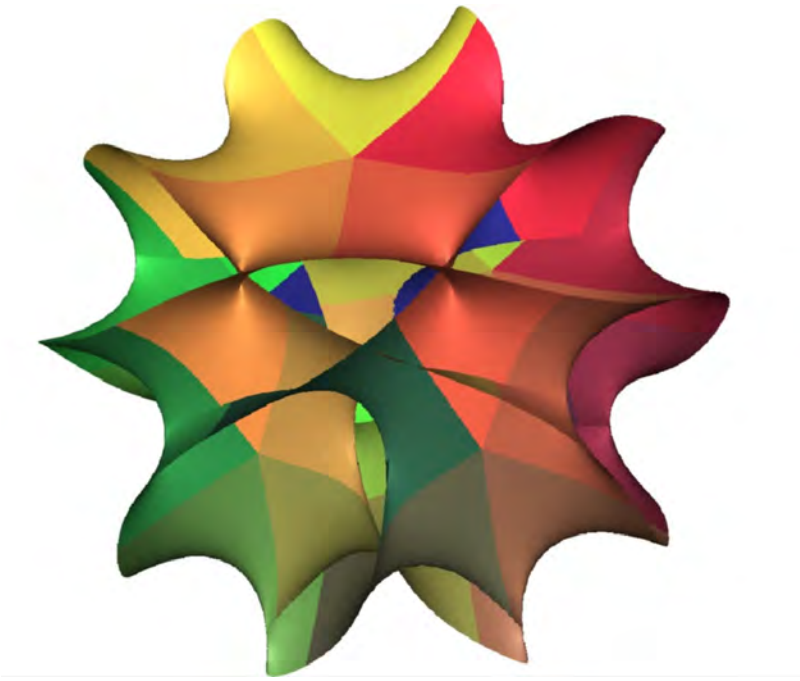


Figure 7 $n = 5$ in $\mathbb{C}\mathbf{P}^2$: The quintic is a section of the Calabi-Yau quintic, the 6-manifold proposed for the hidden dimensions of string theory.

Table 1 Road map of the simple homogeneous polynomial Calabi-Yau spaces.

N	CP	deg(f)	$\mathbb{C}$ dim	$\mathbb{R}$ dim	*Remarks*
1	$\mathbb{C}\mathbf{P}^1$	2	0	0	$z = \pm 1$, the 0-sphere $\mathbf{S}^0$
2	$\mathbb{C}\mathbf{P}^2$	3	1	2	flat torus $\mathbf{T}^2$
3	$\mathbb{C}\mathbf{P}^3$	4	2	4	K3 surface
4	$\mathbb{C}\mathbf{P}^4$	5	3	6	CY String Theory quintic
N	$\mathbb{C}\mathbf{P}^N$	N+1	N-1	2(N-1)	Solution of $\sum_{i=1}^{N} (z_i)^{N+1} = 1$

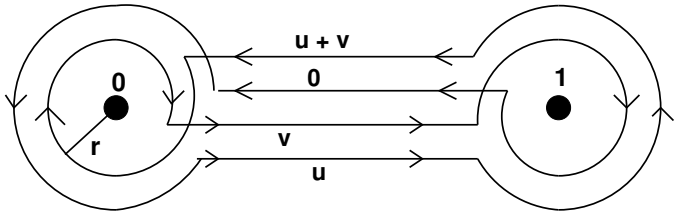


Figure 8 The Euler Beta Function has this remarkable *unshrinkable* contour representation due to Pochhammer.

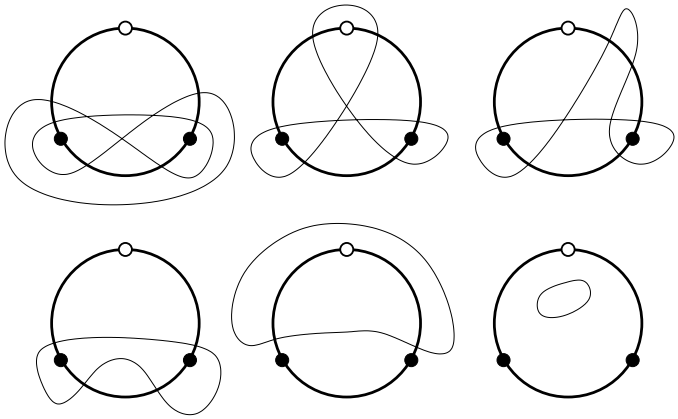


Figure 9 When $u + v = 0, -1, \ldots$, deformation through ∞ has no obstruction: one can simply unloop the contour and pass through the $[0, 1]$ branch line, resulting in a contour that *shrinks to zero.*

2.5 Visualizing Infinite Riemann Surfaces: the Pochhammer Contour

The next challenge is to consider the problems of visualizing complex functions that, unlike the square root and its analogs, may have infinite Riemann surfaces. There is a classic example from the 19th century that provides all the features relevant to this problem. The Euler Beta Function can be represented as the improper integral

$$B(u, v) = \int_0^1 x^{u-1}(1-x)^{v-1}\, dx \tag{29}$$

with the analytic continuation

$$B(u, v) = \frac{\Gamma(u)\Gamma(v)}{\Gamma(u+v)}\ . \tag{30}$$

Now, if one considers the integrand of Eq. (29) as a branched complex function $z^{u-1}(1-z)^{v-1}$ defining a Riemann surface, one finds branch points at $z = 0$ and at $z = 1$ with possibly infinite branchings if u or v should be irrational. However, in 1890 Pochhammer was clever enough to see this not as a problem but as an opportunity to define a new kind of contour integral that was not sensitive to infinite branchings ([5]). In Figure 8 we show the usual planar sketch of Pochhammer's Contour, from which a little analysis allows us to compute its value as

$$\epsilon(u, v) = (1 - e^{2\pi i u})(1 - e^{2\pi i v}) \int_0^1 x^{u-1}(1-x)^{v-1}\, dx \tag{31}$$

$$= (1 - e^{2\pi i u})(1 - e^{2\pi i v}) B(u, v)\ . \tag{32}$$

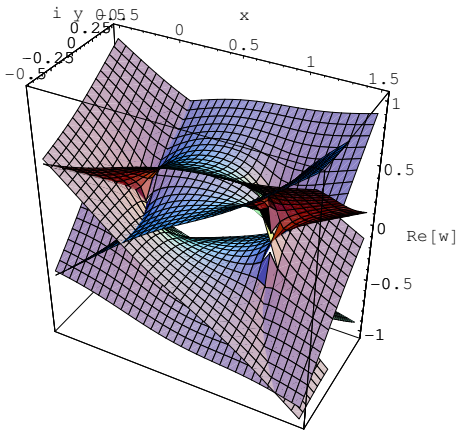


Figure 10 Sample Riemann surface for the B_4 integrand – multiple branch coverings spiral to infinity.

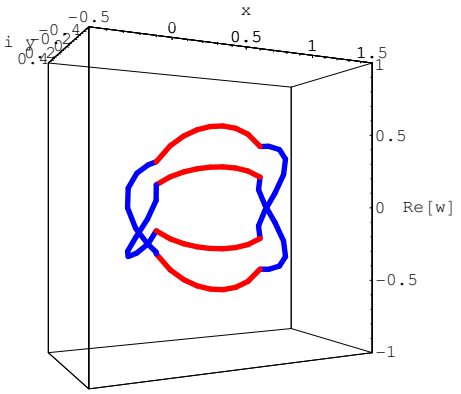


Figure 11 The corresponding embedded Pochhammer contour is a *commutator*, encircling each branch point *twice*.

Through an interesting trick of complex analysis, one can determine the zeroes of the function directly to occur when $u+v=0,-1,\ldots$: at these values, the contour can be "pulled over" the point at infinity as shown in Figure 9 and deformed to an equivalent vanishing loop. These zeroes can be confirmed explicitly from the analytic continuation Eq. (30).

Finally, we can explicitly create a function representing the Riemann surface of the Euler Beta function integrand.

$$\beta(z;\,u,v)=z^{u-1}(1-z)^{v-1} \tag{33}$$

and create a 3D projection with, e.g., the vertical axis given by $\mathrm{Re}\,\beta(z;\,u,v)$. Figures 10 and 11 show a section of a branched covering that could in principle spiral indefinitely, along with the closed Pochhammer loop that can be traced on the Riemann surface, no matter how complex. Figure 12 superimposes these on the same space to illustrate the context.

3 Extending $\mathbb{C}\mathrm{P}^2$ Visualization Methods to $\mathbb{C}\mathrm{P}^3$

Our next objective is to see how we can create some basic images of the K3 surface that give more global information than the 2D slice representation that we saw in Figure 6. We recall that K3 can be represented in general as a homogeneous quartic polynomial in $\mathbb{C}\mathbf{P}^3$ in the form

$$z_0{}^4+z_1{}^4+z_2{}^4++z_3{}^4=0\ , \tag{34}$$

which reduces after division by $z_0\neq 0$ (or equivalently, after division by z_1, z_2, or z_3) to Eq. (27). Even though this is a 4-manifold (after division, 6 real variables and two real constraint equations), we can pick out some of its global, non-slicing, properties by considering what amounts to the "real subspace" of the parameterization.

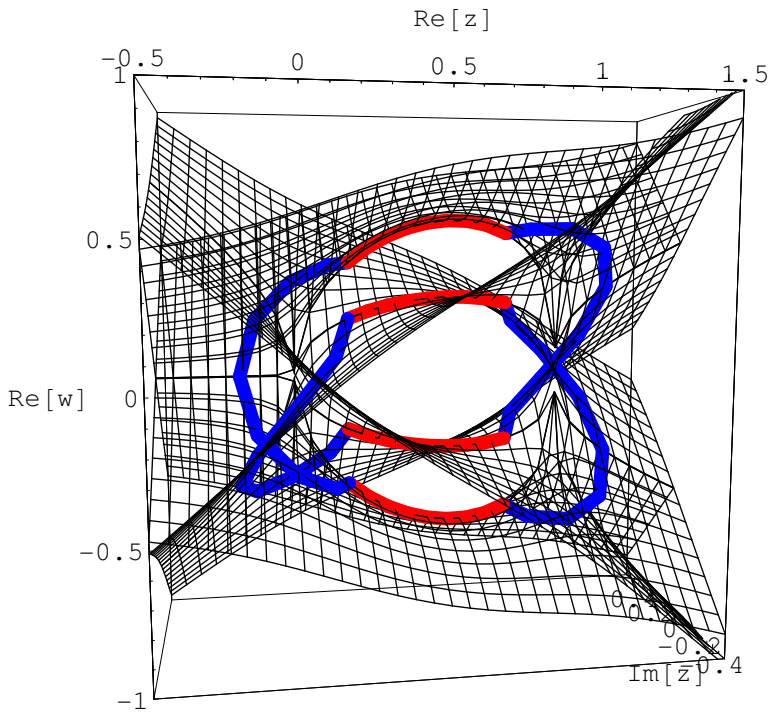


Figure 12 Pochhammer contour plotted directly on the Riemann surface.

3.1 $\mathbb{C}\mathbf{P}^2$ Example

We can see an example of a dimensional reduction in the $\mathbb{C}\mathbf{P}^2$ case by remembering that in the n^2 patch method ([3]), each patch is parameterized as

$$\begin{aligned} z_1 &= (\cos(\theta + i\xi))^{2/n} \\ z_2 &= (\sin(\theta + i\xi))^{2/n} \end{aligned}$$

with $0 \leq \theta \leq \pi/2$, so, with the imaginary part of the argument approaching zero ($\xi \approx 0$), the complex variables (z_1, z_2) become purely real and describe a circular quarter arc from $z_1(\theta = \pi/2) = 0$ to $z_2(\theta = 0) = 0$. Since, at $z_1 = 0$, multiplying by the z_1 phase $\exp(2\pi i k_1/n)$ leaves zero as a fixed point, there will be n copies of the circular arc fanning out from $z_1 = 0$ to n copies of the intersection point $z_2 = 0$. (Note: these may be understood simply as the roots of $(z_1)^n + (z_2)^n = 1$ with either $z_1 = 0$ or $z_2 = 0$.)

In Figure 13, we show the $z_1 = 0$ intersection points as green cubes, the $z_2 = 0$ intersection points as red cubes, and let ξ have a finite range to show the surface shape near the core. In Figure 14, we set $\xi \approx 0$ to expose the n^2 "real core" curves forming the graph of the surface skeleton in this local inhomogeneous coordinate system. Note that this is not quite a topologically symmetric structure because infinity has been treated specially in this coordinate system, but a great deal of the structure, e.g., the degree of the polynomial, is clearly exposed.

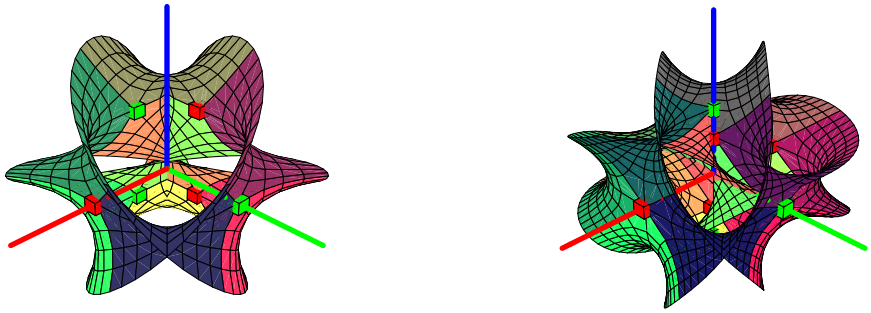


Figure 13 Left: $n = 3$ cubic. Right: $n = 4$ quartic. Red and Green points represented as small cubes indicate where the zeros of z_1 and z_2 pass through the surface; each of $z_2 = 0$ Red points is connected to all n copies of the $z_1 = 0$ Green points, and vice versa, through the "real" central line of each of the n^2 patches.

Figure 14 Shrinking the complex extent of the surface parameterization so that only the "real core" curves remain shows a connected graph of n^2 arcs connecting the $2n$ nodes.

3.2 The K3 "Real Core"

The representation of the K3 surface as a fourth-degree homogeneous polynomial in $\mathbb{C}\mathbf{P}^3$, like the general case of the $\mathbb{C}\mathbf{P}^2$ polynomials described earlier, can be solved in a variety of ways. Here we will focus on generalizing the n^2 patch method ([3]) to $\mathbb{C}\mathbf{P}^3$, which turns out to lead naturally to n^3 patches of dimension 4 (and for $\mathbb{C}\mathbf{P}^N$, to n^N patches of dimension $2(N-1)$). Thus the basic equation for which we seek a 4-parameter parametric form is

$$(z_1)^4 + (z_2)^4 + (z_3)^4 = 1 \ . \tag{35}$$

Following the complexified circle ($\mathbf{S}^1$) method used for $\mathbb{C}\mathbf{P}^2$, we arrive at a 4-manifold parameterization based on the complexified sphere $\mathbf{S}^2$, namely

$$(\cos\theta \sin\phi, \sin\theta \sin\phi, \cos\phi) \tag{36}$$

with $0 \le \phi \le \pi$, $0 \le \theta < 2\pi$. For the full 4D patch, we would complexify the angular variables as $\theta \to \theta + i\xi$, $\phi \to \phi + i\rho$ to get exponential growth towards infinity. To retain the "real skeleton" surface analogous to the network of edges shown in Figure 14 for $\mathbb{C}\mathbf{P}^2$ polynomials, all we need to do is recast the real equation (36) in the form

$$\begin{aligned} w_1 &= (\cos\theta \sin\phi)^{2/n} \\ w_2 &= (\sin\theta \sin\phi)^{2/n} \\ w_3 &= (\cos\phi)^{2/n} \ , \end{aligned}$$

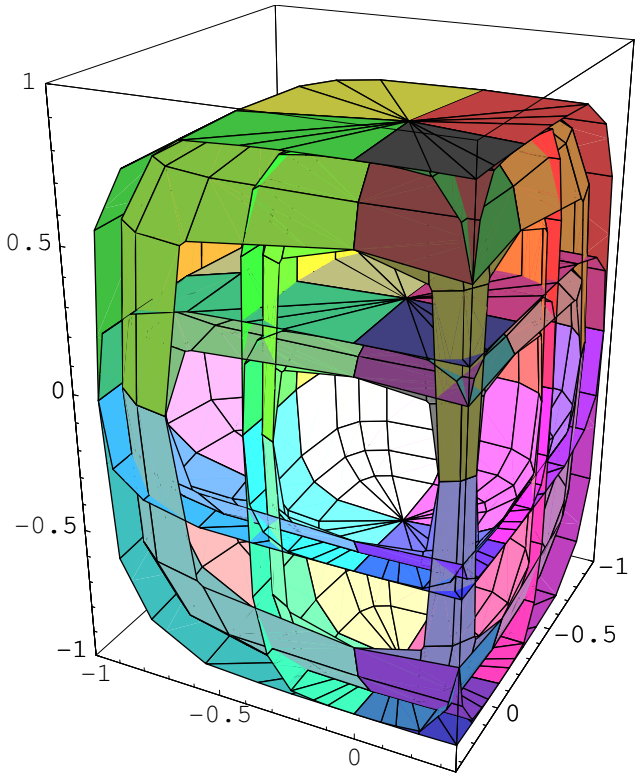


Figure 15 The "real core" of the quartic K3 Calabi-Yau space, delimited by $4^3 = 64$ spherical triangles. Each spherical triangle is bounded by the intersections of the zeros of the three local complex variables with the K3 surface (a 4-manifold).

with $n = 4$ selecting the K3 fundamental domain ($2/n = 2/4 = 1/2$ so each term is a square root). The remaining patches are found by making phase transformations on (z_1, z_2, z_3) until the entire surface is covered; all the patches are then labeled by the $n^3 = 64$ integers (k_1, k_2, k_3), where $k_i = 0, 1, 2, 3$ and the parametric solutions of the equations become

$$\begin{aligned} z_1 &= \exp(2\pi i k_1/4) w_1 \\ z_2 &= \exp(2\pi i k_2/4) w_2 \\ z_3 &= \exp(2\pi i k_3/4) w_3 \ . \end{aligned}$$

The result is a collection of octants of the sphere, spherical triangles that fan out four at a time from the curves where $z_1 = 0$, $z_2 = 0$, and $z_3 = 0$ intersect the manifold. The result is depicted in Figure 15, where part of the shape is cut away so that the interior "fanning out" is made visible. We remark that just as Figure 14 appears to have non-manifold triple or quadruple intersections, but in fact, when complexified, yields the smoothed continuous surface of Figure 13, we need to imagine that in 4D, Figure 15 also extends completely smoothly away from the 4-way fan-out junctions.

3.2.0.4 Regular global tessellations

The representations shown here are local and are limited in their effectiveness for exposing the overall topology of the polynomials in $\mathbb{C}\mathbf{P}^2$ and $\mathbb{C}\mathbf{P}^3$, etc. Extending these limited local representations to global tessellations with maximal symmetries is a subject of ongoing research.

4 Two Complex Variables and the Dodecahedron

Finally, we review our approach to creating visualizations for problems arising in many-complex-variable analysis, outlining the two-complex-variable case as our main example ([4]). We begin with the N-particle bosonic scattering amplitude of the original dual model, the precursor to string theory, which is given by the integral

$$B_N = \int \cdots \int_{\text{vol}} \left[\prod_{ij} {u_{ij}}^{\alpha_{ij}-1} \right] \frac{\prod_k du_{1k}}{(1-u_{14}) \cdots (1-u_{1,N-1})}$$

The N-point cross-ratios u_{ij} obey a set of constraints that is non-linear except for the 4-particle case B_4, which in fact is the Euler Beta function treated earlier:

$$u_{ij} = 1 - \prod_{m=i+1}^{j-1} \prod_{n=j+1}^{i-1} u_{mn} \ .$$

These constraints define manifolds in $\mathbb{C}\mathbf{P}^{N(N-3)/2}$ that provide new insight into the nature of the analytic continuation of these integrals.

The $N = 5$ case is the simplest non-trivial example that we can work out explicitly; the corresponding improper integral is two-dimensional,

$$\begin{aligned} B_5(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5) = \\ \\ = \int_0^1 \int_0^1 s^{\alpha_1 - 1} t^{\alpha_2 - 1} (1-s)^{\alpha_3 - 1} (1-st)^{\alpha_4 - \alpha_3 - \alpha_5} (1-t)^{\alpha_5 - 1} \, ds \, dt \\ \\ = \iint (z_1)^{\alpha_1 - 1} (z_2)^{\alpha_2 - 1} (z_3)^{\alpha_3 - 1} (z_4)^{\alpha_4 - 1} (z_5)^{\alpha_5 - 1} \, ds \, dt / (1-st) \ , \end{aligned} \tag{37}$$

and thus must eventually be treated using two complex variables.

The B_5 cross-ratio constraints are quadratic,

$$\begin{aligned} 1 - z_1 - z_3 z_4 &= 0 \\ 1 - z_2 - z_4 z_5 &= 0 \\ 1 - z_3 - z_5 z_1 &= 0 \\ 1 - z_4 - z_1 z_2 &= 0 \\ 1 - z_5 - z_2 z_3 &= 0 \ , \end{aligned} \tag{38}$$

and, in these variables, there are twelve different possible integration domains of Eq. (37) that we can initially represent as in Figure 16. Using $\mathbb{C}\mathbf{P}^5$ cross-ratio variables, we would properly represent these equations as ${z_0}^2 - z_0 z_1 - z_3 z_4 = 0$, etc., but we will omit the details here.

Each individual region, when plotted using Eq. (38), is not simply a square or triangle as one might guess from the Cartesian variable plot in Figure 16, but is actually a pentagon, as shown in Figure 17

The resulting figure is a "blown-up" dodecahedral manifold formed from twelve pentagons, but this dodecahedron, shown in Figure 18, is quite different topologically from the familiar Platonic dodecahedron, and in fact has Euler characteristic $\chi = -3$, so it is a surface

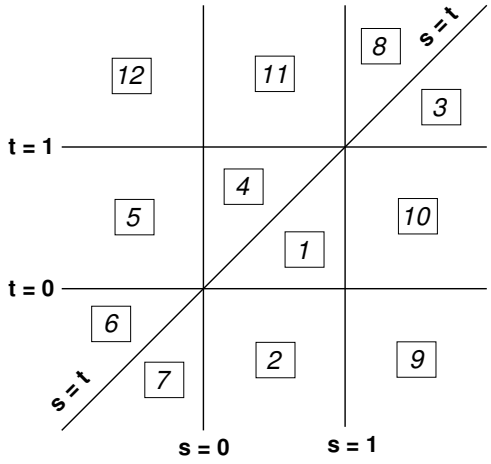


Figure 16 The 12 connected components of the domain of parameters for the set of 5-point cross-ratios.

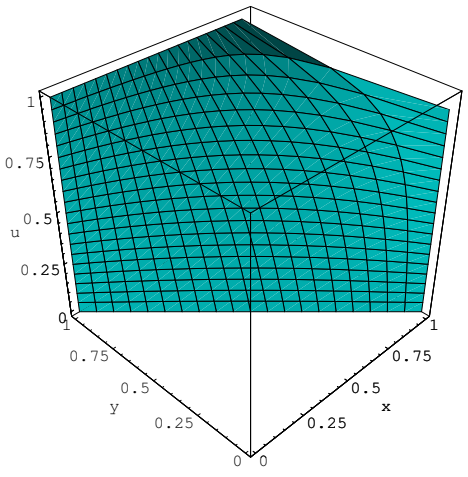


Figure 17 A direct plot of the solutions (e.g., $(x, (1-x)/(1-xy), y)$) shows "stretched" limits turning squares or triangles in (s,t) coordinates into pentagons in $[z_1, z_2, z_3, z_4, z_5]$-space.

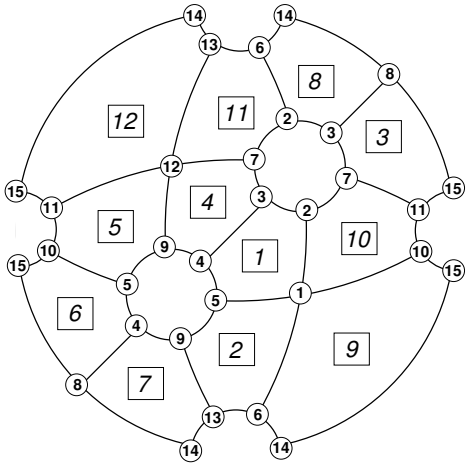


Figure 18 The 12 B_5 connected components without singularities.

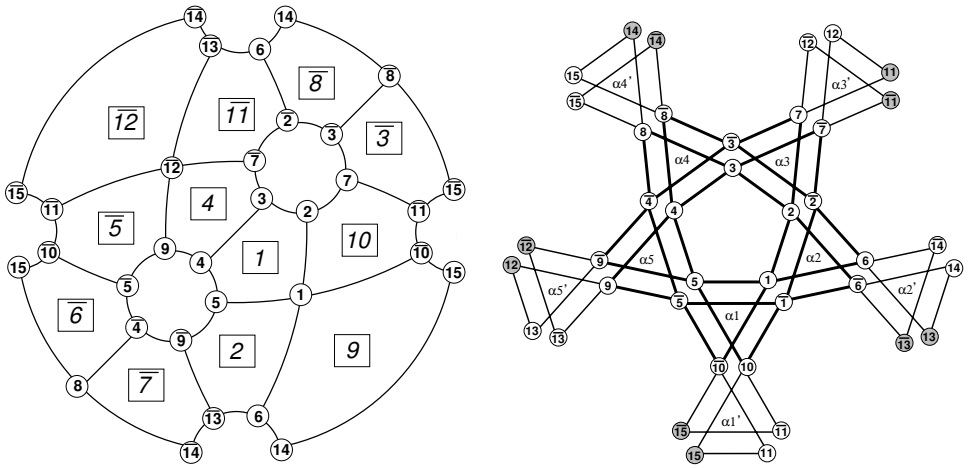


Figure 19 (a) The 12 B_5 connected components with double covering extensions noted by barred numbers. (b) Graph showing each vertex and the 10 hexagons corresponding to the double-covered $\mathbb{C}\mathbf{P}^1$ branch lines in the B_5 integrand.

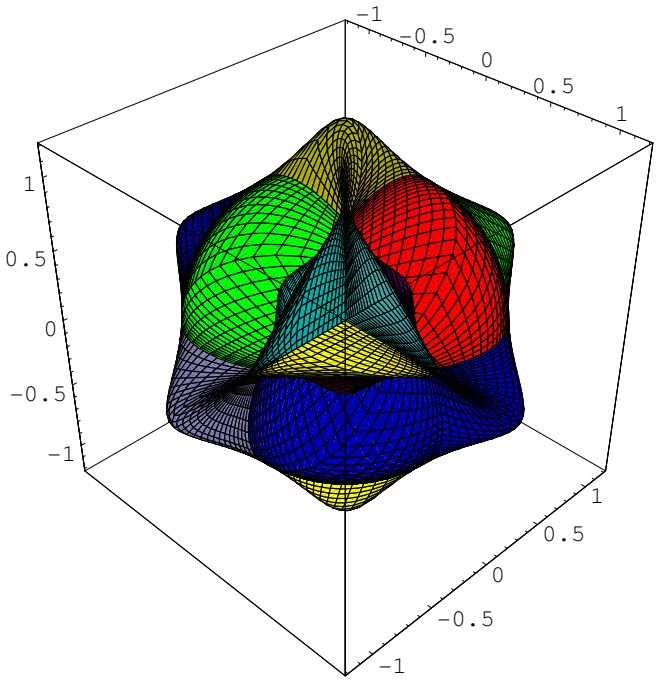


Figure 20 Genus 4 double cover of five-cross-cap surface projected from $\mathbb{R}^6$.

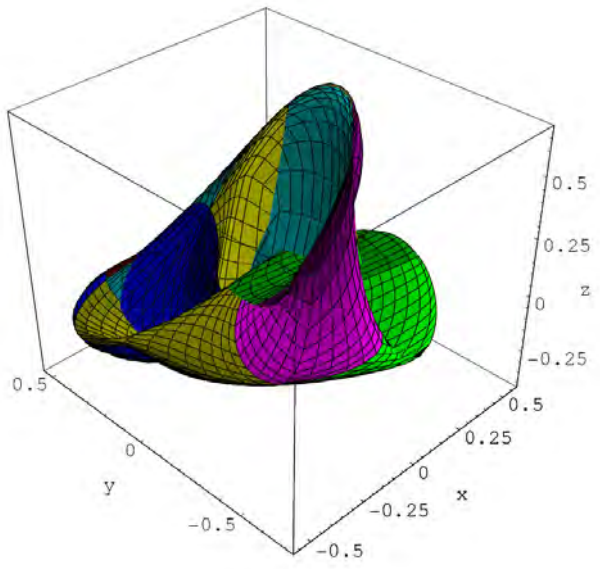


Figure 21 Projected embedding of the five-cross-cap surface itself.

corresponding to a sphere with *five cross-caps*. Figure 19 shows the solutions of the cross-ratio constraints, which double cover the five-cross-cap surface, and Figure 20 finally shows the visualization of the full double-covered surface embedded in $\mathbb{R}^6$. Creating an identified embedding of the five cross-cap surface itself (the single cover, not the double cover) employing the methods of ([4]) leads to the example image in Figure 21.

Much remains to be done to create further informative visualizations of these families of surfaces.

5 Discussion and Remarks

We have reviewed a family of basic problems in the complex analysis of one and many variables, and presented visualizations of a number of the manifolds that naturally arise. A new method, the use of cross-ratio variables instead of the expected $\mathbb{C}\mathbf{P}^N$ variables for the analysis of N complex variables, shows promise and is essential for the treatment of certain N-dimensional integrals. Among the surprising and unexpected aspects of this investigation was the discovery of a relationship to a family of objects known as the Stasheff associahedra ([7, 6, 2]). The explicit geometric embeddings that we discovered as part of our explicit graphics-oriented approach are in fact new *geometric* realizations of these *topological* objects.

Complex analysis is a fertile proving ground for developing and testing mathematical visualization methods. Here we have reviewed a variety of basic approaches that can be used for visualization in complex analysis, focusing on several families of complex algebraic equations and their geometry. Much remains to be done.

Acknowledgments

This research was supported in part by NSF grant numbers CCR-0204112 and IIS-0430730. Our thanks to Philip Chi-Wing Fu and Sidharth Thakur for assistance with graphics tools.

References

1 Élie Cartan. *The Theory of Spinors*. Dover, New York, 1981.

2 Satyan L. Devadoss. Tessellations of moduli spaces and the mosaic operad. In *Homotopy invariant algebraic structures. (Baltimore, MD, 1998)*, volume 239, pages 91–114, Providence, RI, 1999. Amer. Math. Soc.

3 A.J. Hanson. A construction for computer visualization of certain complex curves. *Notices of the Amer.Math.Soc.*, 41(9):1156–1163, November/December 1994.

4 Andrew J. Hanson and Ji-Ping Sha. A contour integral representation for the dual five-point function and a symmetry of the genus four surface in R^6. *Journal of Physics A: Mathematics and General*, 39:2509–2537, 2006. Web version at http://www.iop.org/EJ/abstract/0305-4470/39/10/017/ or on the arXiv at http://arxiv.org/abs/math-ph/0510064.

5 L.A. Pochhammer. Zur theorie der Euler'schen integrale. *Math. Ann.*, 35:495–526, 1890.

6 James Stasheff. Homotopy associativity of H-spaces. *Trans. Amer. Math. Soc.*, 108:275–292, 1963.

7 James Stasheff. *H-spaces from a Homotopy Point of View*. Lecture Notes in Mathematics 161. Springer Verlag, New York, 1970.

8 S.-T. Yau. On the Ricci curvature of a compact Kähler manifold and the complex Monge-Ampere equation. *Commun. Pure and Appl. Math.*, 31:339–411, 1978.

Tensor Field Reconstruction Based on Eigenvector and Eigenvalue Interpolation

Ingrid Hotz[1], Jaya Sreevalsan-Nair[1], Hans Hagen[2], and Bernd Hamann[1]

1 **Institute for Data Analysis and Visualization (IDAV)**
Department of Computer Science, University of California
Davis CA 95616, USA
{ihotz,sreevals,hamann}@cs.ucdavis.edu

2 **Computergrafics Lab, University of Kaiserslautern, Germany**
hagen@informatik.uni-kl.de

Abstract

Interpolation is an essential step in the visualization process. While most data from simulations or experiments are discrete many visualization methods are based on smooth, continuous data approximation or interpolation methods. We introduce a new interpolation method for symmetrical tensor fields given on a triangulated domain. Differently from standard tensor field interpolation, which is based on the tensor components, we use tensor invariants, eigenvectors and eigenvalues, for the interpolation. This interpolation minimizes the number of eigenvectors and eigenvalues computations by restricting it to mesh vertices and makes an exact integration of the tensor lines possible. The tensor field topology is qualitatively the same as for the component wise-interpolation. Since the interpolation decouples the "shape" and "direction" interpolation it is shape-preserving, what is especially important for tracing fibers in diffusion MRI data.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Tensor Field, Eigenvector, Eigenvalue, Interpolation

Digital Object Identifier 10.4230/DFU.SciViz.2010.110

1 Introduction

Typically visualization deals with discrete data; nevertheless, many visualization methods are designed to generate a continuous data representation. We need an appropriate interpolation or approximation method, which must be chosen from a large number of possibilities, which can influence the visualization result significantly. This choice is often guided by two - sometimes conflicting - goals: (i) the interpolation should be "simple," meaning it should simplify our computations; and (ii)the interpolation should be "natural," meaning that it should represent the real data as well as possible without introducing too many artifacts. Often losses due to simple interpolation schemes are accepted to simplify the computations.

In many cases a method accomplishing simplicity well uses a linear interpolation schema on a triangulated or tetrahedrized domain. For scalar fields, a unique piecewise linear interpolation is defined by a given triangulation. For vector fields, there are two obvious linear interpolations schemas: one based on the interpolation of the vector field components, and the other one based on or alternatively the direction and length of the vectors. In most cases, both approaches lead to similar results. Considering tensor fields, there are even more ways to interpolate linearly. The most common approach is a linear interpolation of the tensor components [4, 9]. But since the entities we are mostly interested in are not the tensor

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 110–123

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

components but tensor invariants, which are in general not linear, the interpolation is not really simple. There are other problems related to this interpolation, e.g. the sign of the determinate is not preserved.

There has been some work done in the area of tensor field interpolation based on the tensor components. Besides the linear approaches more advanced interpolation methods based on components have been developed with goals of noise reduction or feature preservation [1, 6, 8].

There are also some papers using direction interpolation. in context of diffusion MRI data with the goal of tracing anatomical fibers [5, 2]. Most of these approaches are specific to diffusion MRI data with the goal of tracing anatomical fibers. Such approaches often focus on regions with high anisotropy where no degenerated points exist and the issue of direction assignment is not so important. These interpolation lead to linear direction field, but are not consistent in regions containing degenerated points.

We introduce a linear interpolation schema for symmetrical tensor fields that combines the advantages of linear interpolation of components, which delivers a consistent field, with the advantages of eigenvector and eigenvalue-based interpolation generating a simple direction field and being shape preserving. It is based on eigenvectors and eigenvalues guided by the behavior of eigenvectors for the component-wise interpolation. It is "shape preserving" and minimizes the number of locations where we have to compute the eigendirections. We discuss its properties and compare it to the standard interpolation, i.e., linear interpolation of tensor components.

2 Outline – Idea

Our goal is the interpolation of the tensor field based on the linear interpolation of eigenvalues and eigenvectors. Since there are two eigenvalues and two eigendirections with each two possible orientations, there exists a variety of linear interpolations based on eigenvectors and eigenvalues, see Figure 1. To specify the interpolation uniquely we first have to assign the eigenvalues to each other and then assign orientations to the eigenvectors. When using vectors for the interpolation, we have to consider the fact that not all structures occurring in tensor fields can be simulated by global vector fields, e.g., winding numbers of half integers. This means that an assignment of directions is not possible globally, but it is possible for simply connected regions, that do not contain degenerate points. The situation is complicated by the fact that for discrete data the existence of a degenerate point inside a cell depends on the chosen interpolation.

We discuss one possible eigenvector eigenvalue interpolation schema which simulates the topology of the component-wise interpolation but decouples the interpolation of "shape," represented by the eigenvalues, and direction. Our orientation assignment for eigenvectors is guided by the behavior of the eigenvectors in case of a component-wise interpolation such that the resulting field topology is qualitatively the same as for the component-wise interpolation. This means that we obtain the same number and types of degenerate points in each triangle. The exact position of the degenerate points and separatrices varies slightly.

In the following sections, we describe first the criterion for the assignment of eigenvalues to each other. Then we discuss the behavior of the eigenvectors for the linear interpolation of tensor components, which is essential for the orientation assignment to the eigendirections. Since the existence of degenerate points inside the triangle influences the interpolation, the next step is to define degenerate points on the basis of eigenvectors at the vertices. The results from these discussions are finally used to define the interpolation of eigenvalues and eigenvectors.

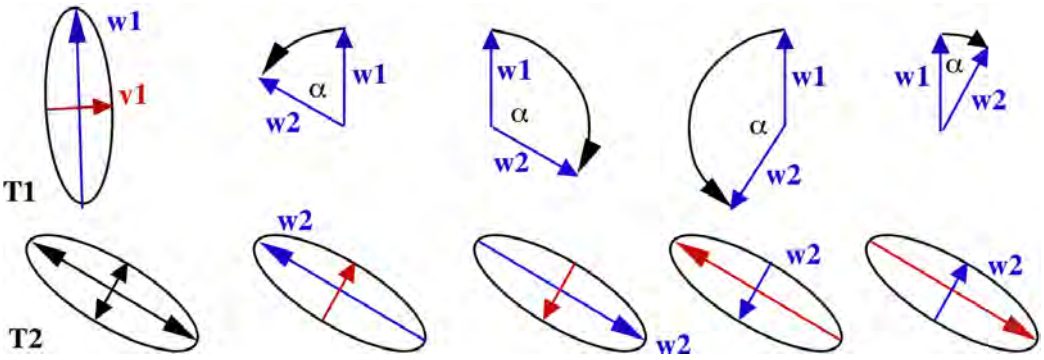


Figure 1 Symmetric tensors can uniquely be represented by their eigendirections and eigenvalues, here represented by the ellipses. Using the eigenvalues and eigendirections for interpolation we have several possibilities for grouping them and assigning orientations to eigendirections. If we restrict rotation angles to values smaller than π, there are four different assignments resulting in four different rotation angles.

3 Basics and Notation

We use the term tensor field for symmetric 2D tensors of second order defined on a triangulated two-dimensional domain. Using a fixed coordinate basis, each tensor can be expressed as a 2×2 matrix, given by four independent scalars. A tensor $\mathbf{T}$ is called symmetric if, for any coordinate basis, the corresponding array of scalars is symmetric. The symmetric part of the tensor is defined by three independent scalars and is represented by a symmetric matrix. We use the following notation:

$$\mathbf{T} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} d+\Delta & F \\ F & d-\Delta \end{pmatrix}, \tag{1}$$

where $\Delta = \frac{E-G}{2}$ and $d = \frac{E+G}{2}$.

A tensor $\mathbf{T}$ is characterized by its *eigenvalues* λ and μ and corresponding *eigenvectors* $\pm\mathbf{v}$ and $\pm\mathbf{w}$. For symmetric tensors, the eigenvalues are always real and the eigenvectors mutually orthogonal. We call the eigenvector with the larger eigenvalue major eigenvector and the smaller eigenvalue minor eigenvector. Further, $\Delta = (\lambda + \mu)/2$. In most points the two eigenvectors of a tensor are defined uniquely, each assigned to one eigenvalue. This is no longer the case for points where both eigenvalues are the same, i.e., $\lambda = \mu$, the so-called *degenerate points*. In tensor field topology the degenerate points play a similar role as zeros (critical points) in vector fields [3, 7]. Independently of the eigenvalues, an isolated degenerate point can be defined by the number of windings an eigenvector performs when moving on a closed line around the degenerate point. The undirected eigenvector field allows winding-numbers to be multiples of one half. The rotational direction provides us with additional information about the characteristics of the degenerate point.

To describe the points inside a triangle with vertices P_1, P_2, and P_3 we use barycentric coordinates β_1, β_2 and β_3, $0 \leq \beta_i \leq 1$, where

$$P(\beta_1, \beta_2, \beta_3) = \beta_1 P_1 + \beta_2 P_2 + \beta_3 P_3, \qquad \sum_{i=1}^{3} \beta_i = 1. \tag{2}$$

The triangle edges are named e_i, $i = 1, 2, 3$, where the index i is chosen according to its opposite vertex P_i. Table 1 summarizes the notation we use for the eigenvalues and

Table 1 Notation for eigenvectors and eigenvalues in triangle P_1, P_2, and P_3. The variables λ and μ are chosen such that $\lambda \geq \mu$. The functions $\lambda(t)$ and $\mu(t)$ are continuous but not necessarily differentiable everywhere. The functions $\mathbf{v}(t)$ and $\mathbf{w}(t)$ might not be continuous. The "$\pm$" in front of the eigenvectors allude to the fact that the eigenvectors are bidirectional and have no orientation.

Point	Tensor	Eigenvalues	Eigenvectors
P_1	T_1	λ_1	$\pm\mathbf{v}_1 = (v_{11}, v_{12})$
		μ_1	$\pm\mathbf{w}_1 = (w_{11}, w_{12})$
P_2	T_2	λ_2	$\pm\mathbf{v}_2 = (v_{21}, v_{22})$
		μ_2	$\pm\mathbf{w}_2 = (w_{21}, w_{22})$
P_3	T_3	λ_3	$\pm\mathbf{v}_3 = (v_{31}, v_{32})$
		μ_3	$\pm\mathbf{w}_3 = (w_{31}, w_{32})$
$P(\beta_1, \beta_2, \beta_3)$	$T(\beta_1, \beta_2, \beta_3)$	$\lambda(\beta_1, \beta_2, \beta_3)$	$\pm\mathbf{v}(\beta_1, \beta_2, \beta_3) = (v_1, v_2)$
		$\mu(\beta_1, \beta_2, \beta_3)$	$\pm\mathbf{w}(\beta_1, \beta_2, \beta_3) = (w_1, w_2)$

eigenvectors at the vertices P_i, $i = 1, 2, 3$, and in the interior of the triangle. We always use "$\pm\mathbf{v}$" and "$\pm\mathbf{w}$" when referring to eigenvectors to allude to the fact that the eigenvectors are bidirectional and have no orientation. We use $\mathbf{v}$ and $\mathbf{w}$ when referring to vectors representing the eigenvectors with an arbitrary but fixed direction for $\mathbf{v}$, e.g., considering the way they were generated by a numerical computation. The direction of $\mathbf{w}$ is defined in a way such that $\mathbf{v}$ and $\mathbf{w}$ is a right-handed system. The assignment of the names λ and μ to the eigenvalues is critical for the interpolation based on eigenvalues and eigenvectors. For component-wise interpolation it is given implicitly. The requirement for the assignment to be continuous resolves this ambiguity: assigning the same name to all major eigenvalues and to all minor eigenvalues. Thus, we can define the variable names in a way that $\lambda_i \geq \mu_i$ for all i, without loss of generality. If there is a degenerate point inside a cell the eigenvalue might not be differentiable there.

4 Linear Interpolation of Tensor Components

The most commonly used interpolation scheme for tensors is linear interpolation of tensor components. It is a consistent approach and produces a globally continuous tensor field approximation. We use this field to guide the orientation assignment to the vector field. To be able to do so we start with a detailed analysis of the eigenvector behavior for a component-wise tensor interpolation.

For linear interpolations, the value at a point P is already uniquely specified by the values in two points P_1 and P_2 whose connection passes trough P. Therefore, many properties can already be observed when considering the linear interpolation in-between two points:

$$P(t) = (1-t)\,P_1 + t\,P_2, \quad T(t) = (1-t)\,T(P_1) + t\,T(P_2), \quad t \in [0,1]. \tag{3}$$

The eigenvalues at $P(t)$ are given by

$$\lambda(t) = d(t) + \sqrt{F^2(t) + \Delta^2(t)} \text{ and } \mu(t) = d(t) - \sqrt{F^2(t) + \Delta^2(t)}\,. \tag{4}$$

There exists a degenerate point in $P(t_0)$ when both eigenvalues $\lambda(t_0)$ and $\mu(t_0)$ are the same.

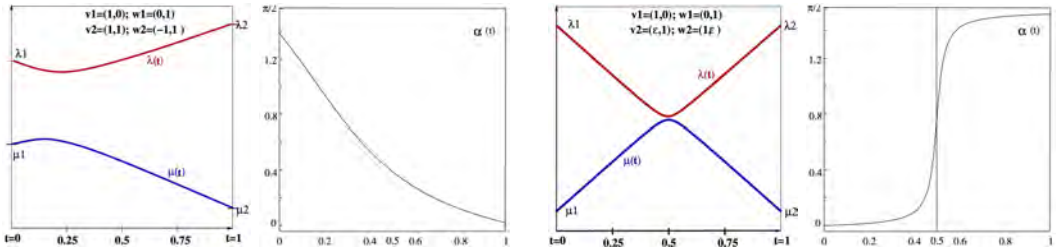


Figure 2 Two examples of the behavior of the eigenvalues and eigenvectors for component-wise linear interpolation when moving along the edge. The images on the right show an example where the rotation angle is almost $\pi/2$. The rotation takes place in a very small region where the eigenvalues reach their extrema.

This is equivalent to $F(t) = \Delta(t) = 0$.

Observation 1
For linear interpolation of tensor components, there exists only a degenerate point on the connection of two points if the following three conditions are satisfied:

$$\begin{array}{lrcl} (a) & F_1 \cdot F_2 & \leq & 0 \\ (b) & \Delta_1 \cdot \Delta_2 & \leq & 0 \\ (c) & F_1\Delta_2 - \Delta_1 F_2 & = & 0 \end{array} \tag{5}$$

If $F_1 = F_2 = 0 = \Delta_1 = \Delta_2$, the entire connection consists of degenerate points. In all other cases, there exists an isolated degenerate point located in $P(t_0)$, where

$$t_0 = \frac{F_1}{F_1 - F_2} = \frac{\Delta_1}{\Delta_1 - \Delta_2}. \tag{6}$$

If the denominators are equal to zero, the entire edge is degenerate. The eigenvalue in the degenerate point is $E(t_0) = G(t_0)$.

If we assume that P_1 is not a degenerate point and use the eigenvectors of T_1 as coordinate basis, then $F_1 = 0$ and $\Delta_1 \neq 0$. Thus, the third condition (5c) reduces to $F_2 = 0$, meaning that the eigenvectors of T_1 are eigenvectors of T_2 as well. The second condition (5c) implies that the corresponding eigenvectors are rotated about $\pi/2$.

Observation 2
For linear interpolation of tensor components, there exists a degenerate point on the connection of two not-degenerate points P_1 and P_2 if and only if

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = 0 = \mathbf{w}_1 \cdot \mathbf{w}_2. \tag{7}$$

Thus, the existence of degenerate points on an edge is independent from the eigenvalues on the vertices.

We now consider the behavior of eigenvectors when moving along an edge without degenerate points. The eigenvectors are-well defined everywhere and change continuously. The change can be expressed by the angle $\alpha(t)$ formed by the eigenvector $\mathbf{v}(t)$ and the x-axis, see Figure 3. The special choice of the angle does not influence the result for the change of the angle. Geometrical considerations using the tensor components to express the eigenvector

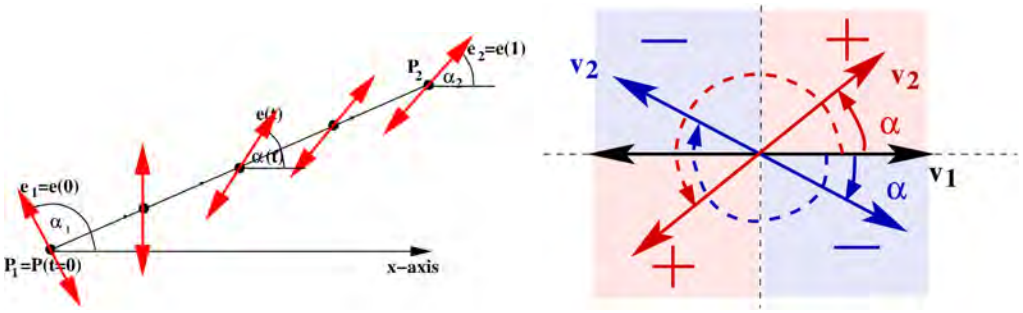


Figure 3 Change of eigenvector along the connection of two points. The rotation direction of the eigenvectors depends on the quadrant in which the second eigenvector lies. Only one possible rotation angle exists with absolute value smaller than $\pi/2$.

allow us to obtain after simplification:

$$\frac{\mathrm{d}\alpha(t)}{\mathrm{d}t} = \frac{1}{2}\frac{\Delta(t)\dot{F}(t) - F(t)\dot{\Delta}(t)}{\Delta^2(t) + F^2(t)}. \tag{8}$$

For linear interpolation of tensor components, this results in

$$\frac{\mathrm{d}\alpha(t)}{\mathrm{d}t} = \frac{1}{2}\frac{F_2\Delta_1 - F_1\Delta_2}{\Delta^2(t) + 4F^2(t)}. \tag{9}$$

Since the denominator is always greater than zero the sign, and thus the rotational direction, is determined by the numerator. Integrating Equation 9 shows that the absolute rotation angle is smaller than $\frac{3}{4}\pi$.

If there exists a degenerate point $D = P(t_0)$ on the edge, it can easily be seen that $\mathbf{v}(t) = \mathbf{v}_1$, $\mathbf{w}(t) = \mathbf{w}_1$ for $0 \leq t < t_0$, and $\mathbf{v}(t) = \mathbf{v}_2$, $\mathbf{w}(t) = \mathbf{w}_2$ for $t_0 < t \leq 1$.

Observation 3
When moving along an edge the rotation angle is limited to an absolute value of $\pi/2$. The direction of the rotation is given by

$$F_2\Delta_1 - F_1\Delta_2.$$

If this expression is smaller than zero, the eigenvector is rotated clockwise; if it is larger it is rotated counter-clockwise; if it is equal to zero, then either both points have the same eigenvectors or there exists a degenerate point on the edge, the rotation is $\pi/2$ and the rotation direction is undetermined.

5 Interpolation Based on Eigenvectors and Eigenvalues

We use the observations from the last sections to define a tensor interpolation inside a triangle based on an eigenvector and eigenvalue interpolation. We use the notations as defined in Table 1.

There are some issues we have to take care in order to use a vector interpolation applied to a tensors. First, we have to define a local criterion for the assignment of an orientation to the eigenvectors $\pm\mathbf{v}_1$ and $\pm\mathbf{w}_1$. Second, we have to show that the rotation angle is smaller than π such that it can be represented by a vector interpolation. Third, we have to show that the interpolated vectors are orthogonal everywhere and thus are valid eigenvectors. The

second point was already shown to be satisfied in the last section. The third point can be shown to hold by performing a simple scalar product calculation. The most critical point, the first one, is discussed in the next sections.

Among all possible assignments, we chose an assignment that reflects the continuous change of the eigenvectors defined by the component-wise interpolation. It is important to differentiate between triangles containing a degenerate point and those that do not, since the eigenvector behavior of triangles with a winding number of half integers cannot be simulated by a simple vector interpolation.

5.1 Edge Labeling

Since a consistent global orientation assignment is not possible, we first define arbitrarily directed eigenvectors $\mathbf{v}_i$ and $\mathbf{w}_i$. Instead of changing directions we label the edges according to the behavior of the eigenvectors when moving along the edge e_i:

$$l(e_i) = \begin{cases} 0 & \text{if there exists a degenerate point on the edge, meaning } \mathbf{v}_j \cdot \mathbf{v}_k = 0 \\ -1 & \text{if the directions of } \mathbf{v}_j \text{ and } \mathbf{v}_k \text{ do not match the direction propagation, meaning } \mathbf{v}_j \cdot \mathbf{v}_k < 0 \\ 1 & \text{if the directions of } \mathbf{v}_j \text{ and } \mathbf{v}_k \text{ match the direction propagation, meaning } \mathbf{v}_j \cdot \mathbf{v}_k > 0 \end{cases} \tag{10}$$

where $i, j, k \in \{1, 2, 3\}$ are cyclic indices.

5.2 Interpolation in Triangles without Degenerate Points

The tensor inside the triangle is defined by its eigenvectors $\mathbf{v}$ and $\mathbf{w}$ and eigenvalues λ and μ. We use the edge labelling defined in the last paragraph to define the interpolation of the eigenvectors, see Figure 4,

$$\begin{aligned} \mathbf{v}(\beta_1, \beta_2, \beta_3) &= \beta_1 \mathbf{v}_1 + \beta_2 l(e_3) \mathbf{v}_2 + \beta_3 l(e_2) \mathbf{v}_3, \\ \mathbf{w}(\beta_1, \beta_2, \beta_3) &= \beta_1 \mathbf{w}_1 + \beta_2 l(e_3) \mathbf{w}_2 + \beta_3 l(e_2) \mathbf{w}_3, \end{aligned} \tag{11}$$

and eigenvalues λ and μ,

$$\lambda(\beta_1, \beta_2, \beta_3) = \sum_{i=1}^{3} \beta_i \lambda_i, \qquad \mu(\beta_1, \beta_2, \beta_3) = \sum_{i=1}^{3} \beta_i \mu_i. \tag{12}$$

5.3 Existence of Degenerate points

We distinguish three cases: (i) isolated degenerate points; (ii) degenerate lines; and (ii) degenerate triangles. The interesting case is the isolated degenerate point. For linear interpolation, there only two types of isolated degenerate points exist, *wedge points* with a winding number of $1/2$ and *trisector points* with a winding number of $-1/2$. Using our edge

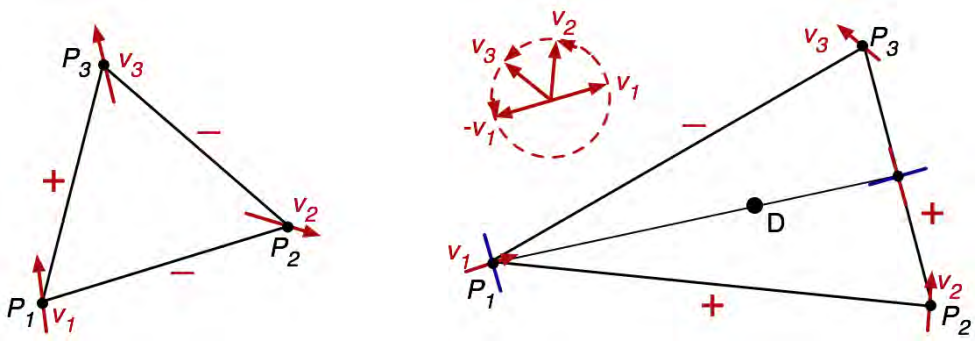


Figure 4 The left figure shows an example of a triangle without a degenerate point. The right figure shows an example of a triangle containing a degenerate point. The edges of the triangle are labelled with a "+" if the direction definition of the two adjacent eigendirections matches the assignment defined by the interpolation of tensor components. They are labelled with "-" if one of the directions must be reversed to obtain a consistent assignment.

labelling convention we obtain a criterion for the existence of a degenerate point inside the triangle.

Observation 4
Let $e_i, i = 1, 2, 3$ be the edges of the current triangle, and let $l(e_i)$ be the edge labelling as defined by Equation 10. The product of the labels of the triangle edges take on the following values:

$$\prod_{i=1,2,3} l(e_i) = \begin{cases} 0 & \text{if there exists a degenerate point on at least one of the edges. If there exist two degenerate edges, we have a degenerate line. If there exist three degenerate edges, the entire triangle is degenerate.} \\ 1 & \text{if there is no degenerate point inside the triangle.} \\ -1 & \text{if there exists an isolated degenerate point inside the triangle.} \end{cases}$$

5.4 Location of Degenerate Points

Since degenerate points at vertices can be detected easily, we restrict our considerations, in this section, to triangles without degenerate behavior at the vertices. Initially, we also assume that there is no degenerate point along the edges, thus the product of the edge labels is -1. Figure 4 shows an example of a triangle containing a degenerate point. We know that the existence of a degenerate point on a line connecting any two points A and B is equivalent to the eigendirections $\mathbf{v}_A$ and $\mathbf{v}_B$ being perpendicular, see Section 4, thus $\mathbf{v}_A \cdot \mathbf{v}_B = 0$. We use this fact to determine the location of the degenerate point in a triangle. First, we define an eigenvector field on the triangle boundary by linear interpolation. For each edge e_i, we define

$$\mathbf{v}(t) = (1 - t) \cdot \mathbf{v}_j + t \cdot l(e_i) \cdot \mathbf{v}_k, \qquad t \in]0, 1[, \tag{13}$$

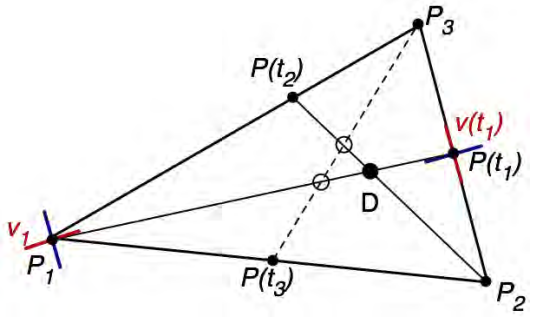


Figure 5 To show that the location of a degenerate point is well-defined we have to show that the three lines connecting the vertices and their opposite points intersect in one point.

where $i, j, k \in \{1, 2, 3\}$ are cyclic indices. Even though the vector field $\mathbf{v}$ on the boundary is not continuous at all vertices, the corresponding un-oriented direction field $\pm\mathbf{v}$, is continuous defining a continuous rotation angle varying from zero to π or $-\pi$.

The mean value theorem implies that there exist three parameters $t_i \in]0, 1[, i = 1, 2, 3$, for each vertex one, such that $\mathbf{v}_i \cdot \mathbf{v}(t_i) = 0$. Thus, for every vertex there exists a point on the opposite edge with rotation angle $\pi/2$. We call this point the opposite point of the vertex, see Figure 5. The following equation defines the parameters t_i:

$$\mathbf{v}_i \cdot ((1 - t_i)\mathbf{v}_j + l(e_i)t_i\mathbf{v}_k) = 0, \tag{14}$$

where $i, j, k \in \{1, 2, 3\}$ are cyclic indices. Using these definitions we finally define the location of the degenerate point in the following way:

Definition
The location of the degenerate point is defined as the intersection of the connections of the triangle vertices and their opposite points. Using barycentric coordinates, the location of the degenerate point D is defined by :

$$\beta_i = \left| \frac{(P(t_j) - P_j) \times (P_j - P_i)}{(P(t_j) - P_j) \times (P(t_i) - P_i)} \right|, \tag{15}$$

with cyclic indices $i, j, k \in \{1, 2, 3\}$ where

$$P(t_i) = (1 - t_i)P_j + t_iP_k \qquad \text{and} \qquad t_i = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\mathbf{v}_i \cdot (\mathbf{v}_j - l_i\mathbf{v}_k)}.$$

To prove that the point D is well-defined we have to show that the resulting point does not depend on the choice of i and j. From the definition of t_i, $i = 1, 2, 3$, Equation 14, it follows that

$$t_1t_2t_3 = (1 - t_1)(1 - t_2)(1 - t_3). \tag{16}$$

This is exactly the condition that three lines through the vertices and points on the opposite edge defined by parameters t_i intersect in one point. Since all other eigenvectors result from linear interpolation of the vectors at the vertices it can be seen that also all other connectors of points with their opposite point intersect in the same point. Thus, the point D is well-defined and can be use to define the location of the degenerate point. The location of

the degenerate point depends only the eigendirection at the triangle vertices, it is independent of the eigenvalues.

If the degenerate point lies on an edge, we cannot use this definition since the three connecting lines degenerate to one line. In this case we use the eigenvalues at the vertices to determine the degenerate point.

5.5 Eigenvalue Definition in the Degenerate Point

A linear interpolation of the eigenvalues at the tree vortices would not lead to multiple eigenvalues. Instead, we interpolate the mean eigenvalue $d = 1/2(\lambda_i + \mu_i)$ and set the deviator $\Delta = 1/2(\lambda_i - \mu_i)$ to zero. If β_i, $i = 1, 2, 3$, are the coordinates of the degenerate point inside the triangle, its eigenvalue is defined by

$$\nu = \frac{1}{2} \sum_{i=1}^{3} \beta_i (\lambda_i + \mu_i) \tag{17}$$

5.6 Interpolation of Triangles with Degenerate Point

For the interpolation of triangles containing degenerate points, we subdivide them by inserting an additional vertex D in the degenerate point. It is connected to the three triangle vertices. The tensor in the new point is defined as the degenerate tensor with eigenvalue ν as defined by Equation 17. Each new triangle is interpolated separately. The eigenvectors, which are not defined in the degenerate point, are set to zero, in correspondence to vector field singularities. The final interpolation of the eigenvectors is performed in the new triangle with vertices P_i, P_j, and D. Let $P = P(\beta_1, \beta_2, \beta_3) := \beta_i P_i + \beta_j P_j + \beta_k D$, using cyclic indices i, j, k. Then the eigenvectors in P are defined by

$$\begin{aligned} \mathbf{v}(\beta_i, \beta_j, \beta_k) &= \beta_i \mathbf{v}_i + \beta_j l(e_k) \mathbf{v}_k \text{ and} \\ \mathbf{w}(\beta_i, \beta_j, \beta_k) &= \beta_i \mathbf{w}_i + \beta_j l(e_k) \mathbf{w}_k, \end{aligned} \tag{18}$$

Thus, the eigenvectors are independent from the coordinate β_k. The eigenvalues are defined in the non-degenerated case as described in Section 5.2.

5.7 Classification of the Degenerate Point

The neighborhood of the degenerate point is characterized by segments separated by radial tensor lines. In our case, the radial tensor lines are straight lines and are defined by their intersection $P(t_r)$ with the triangle boundary:

$$\mathbf{v}(t) \times (P(t_v) - D) = 0, \qquad \mathbf{w}(t) \times (P(t_w) - D) = 0, \qquad t_v, t_w \in [0, 1]. \tag{19}$$

These are quadratic equations for each edge with a maximum of two solutions per edge and eigenvector field. For the entire triangle, one or three solutions per eigenvector field are possible.

Trisector Point

The trisector point has a winding number of $-1/2$. It is characterized by three separatrices and three hyperbolic sectors for each eigenvector field, see Figure 6.

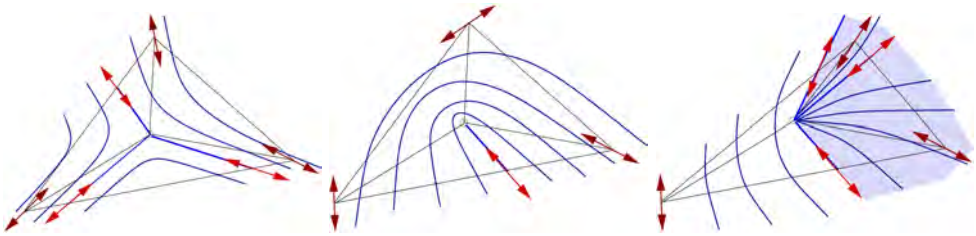


Figure 6 The left figure is an example of a trisector point for one eigenvector field. The middle and right figure are two examples showing wedge points, with one and three radial tensor lines, respectively.

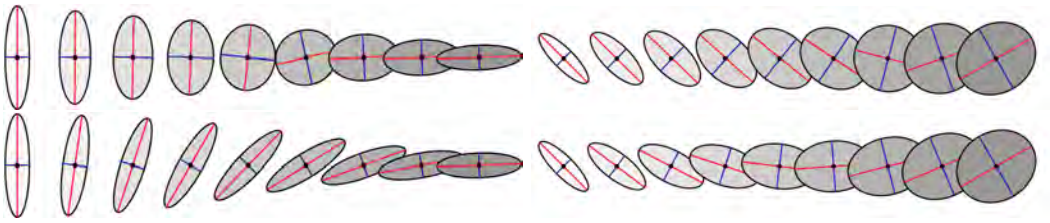


Figure 7 Two examples comparing the results of linear component-wise tensor interpolation (top row) and linear interpolation of eigenvectors and eigenvalues (bottom row). The second interpolation is much more shape-preserving, and the change of directions is much more uniform.

Wedge Point

A wedge point has a winding number of 1/2. It is characterized by one to three radial tensor lines. These radial lines define either one hyperbolic sector or one hyperbolic and two parabolic sectors, see Figure 6.

6 Results

We provide some examples to illustrate the basic differences of the two interpolation methods. In the figures we use ellipses to represent tensors. The half axes are aligned to the eigenvector field, and the radii represent eigenvalues. The half axes are defined as

$$r_1 = 1 + c\lambda \qquad \text{and} \qquad r_2 = 1 + c\mu, \tag{20}$$

where c is a normalization factor. This approach allows us to represent non positive definite tensors as well as ellipses.

The first example shows interpolations on a line, see Figure 7. It illustrates, that the decoupling of eigenvalue and eigenvector interpolation preserves shape. The change of the direction is much more continuous than for component-wise interpolation. Similar observations can be made for the interpolation inside a triangle without degenerate point, see Figure 8. Figure 9 shows two examples of triangles with degenerate points, one with wedge point and one with trisector point. Since interpolation of the eigenvectors is only piece wise linear, due to the subdivision, it allows a more general structure when compared with component-wise interpolation. The qualitative structure of the mesh generated by integrating the eigenvector fields are the same. The position of the degenerate point varies slightly, but the type of the degenerate point is always the same. It can happen that, for eigenvector interpolation the wedge point has two more radial tensor lines, resulting in an additional parabolic, sector, see 9(b).

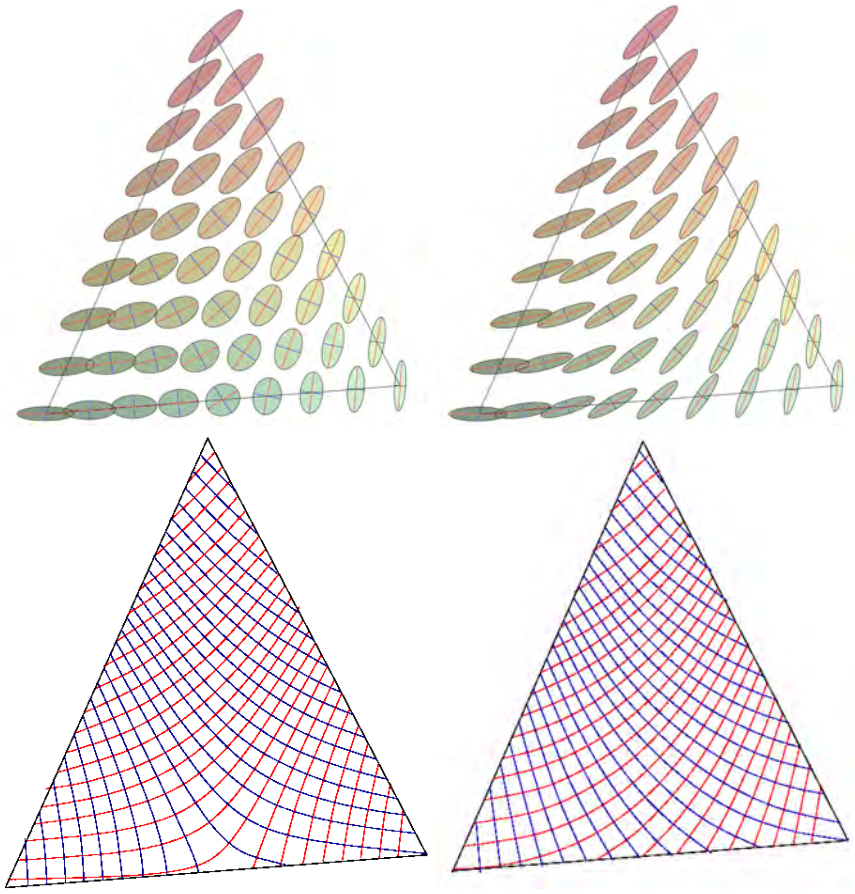


Figure 8 Comparison of component-wise (left) and eigenvalue, eigenvector-based (right) tensor interpolation inside a triangle without degenerate point. The upper row shows the tensors represented as ellipses, and the second row shows the mesh resulting form integrating the eigenvector fields.

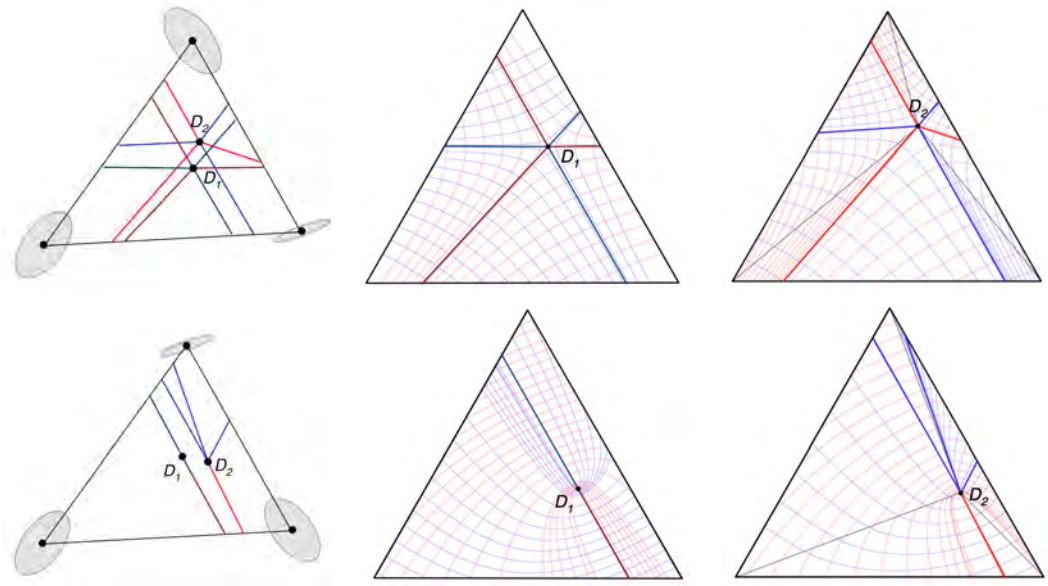


Figure 9 Comparison of component-wise (middle) and eigenvalue, eigenvector-based (right) tensor interpolation inside a triangle with degenerate point. The upper row shows a triangle with trisector point, and the bottom row a triangle with wedge point. The triangles on the left compare the separatrices for both interpolations. For the wedge point case, there exist two more radial lines with two additional parabolic sectors for the eigenvector interpolation.

7 Acknowledgements

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSS-DSV) program under contract ACI 9982251, and a large Information Technology Research (ITR) grant; the National Institutes of Health under contract P20 MH60975-06A2, funded by the National Institute of Mental Health and the National Science Foundation; and the U.S. Bureau of Reclamation. We thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis.

References

1 A. Aldroubi and P. Basser. Reconstruction of vector and tensor fields from sampled discrete data. *Contemporary Mathematics*, 247:1–15, 1999.

2 O. Coulon, Daniel C. Alexander, and S.R. Arridge. Tensor field regularisation for dt-mr images. In *MIUA01, British. Conference on Medical Image*, pages 21–24, Birmingham, UK, 2001.

3 Lambertus Hesselink, Thierry Delmarcelle, and James L. Helman. Topology of second-order tensor fields. *Computers in Physics*, 9(3):304–311, May/June 1995.

4 Gordon Kindlmann, D. Weinstein, and D. Hart. Strategies for direct volume rendering of diffusion tensor fields. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):124–138, 2000.

5 M. Martin-Fernandez, C.-F. Westin, and C. Alberola-Lopez. 3d bayesian regularization of diffusion tensor MRI using multivariate gaussian markov random fields. In *7th In-*

ternational Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'04), Lecture Notes in Computer Science, pages 351–359, Rennes - Saint Malo, France, September 2004.

6 Maher Moakher and Philipp G. Batchelor. Symmetric positive-definite matrices. In Joachim Weickart and Hans Hagen, editors, *Visualization and Image Processing of Tensor Fields*, Mathematics and Visualization, pages 285–297. Springer, 2006.

7 Xavier Tricoche. *Vector and Tensor Field Topology Simplification, Tracking and Visualization*. PhD thesis, University of Kaiserslautern, April 2002.

8 Joachim Weickert and Martin Welk. Tensor field interpolation with pdes. In Joachim Weickart and Hans Hagen, editors, *Visualization and Processing of Tensor Fields*, Mathematics and Visualization, pages 315–324. Springer, 2005.

9 Leonid Zhukov and Alan H. Barr. Oriented tensor reconstruction: Tracing neural pathways from diffusion tensor MRI. In Robert Moorhead, Markus Gross, and Kenneth I. Joy, editors, *Proceedings of the 13th IEEE Visualization 2002 Conference (VIS-02)*, pages 387–394, Piscataway, NJ, October 27– November 1 2002. IEEE Computer Society Press.

Tracking Lines in Higher Order Tensor Fields

Mario Hlawitschka[1] and Gerik Scheuermann[1]

1 **Department of Computer Science, University of Leipzig**
Postfach 100920, 04009 Leipzig, Germany
{hlawitschka,scheuermann}@informatik.uni-leipzig.de

Abstract

While tensors occur in many areas of science and engineering, little has been done to visualize tensors with order higher than two. Tensors of higher orders can be used for example to describe complex diffusion patterns in magnetic resonance imaging (MRI). Recently, we presented a method for tracking lines in higher order tensor fields that is a generalization of methods known from first order tensor fields (vector fields) and symmetric second order tensor fields. Here, this method is applied to magnetic resonance imaging where tensor fields are used to describe diffusion patterns for example of hydrogen in the human brain. These patterns align to the internal structure and can be used to analyze interconnections between different areas of the brain, the so called tractography problem. The advantage of using higher order tensor lines is the ability to detect crossings locally, which is not possible in second order tensor fields. In this paper, the theoretical details will be extended and tangible results will be given on MRI data sets.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Tensor Field, Line Tracking

Digital Object Identifier 10.4230/DFU.SciViz.2010.124

1 Introduction

Tensors are mathematical objects that are used in physics and engineering for measuring natural quantities or for describing derived quantities such as the vector derivative which is a second order tensor. While second order tensors, especially symmetric second order tensors are well studied and many visualization techniques exist, little has been done to visualize tensors of order higher than two. Higher order tensor occur for example in mechanical engineering as the fourth order material tensor but despite of painting complex glyphs, no method exists for analyzing the structure of higher order tensor fields.

Magnetic resonance tomography (MRT) is an imaging technique used in medicine that is more sensitive to tissue structures than computer tomography (CT). Diffusion weighted MRT is a variant where diffusion of hydrogen bound in molecules is measured along gradient directions of an applied magnetic field. As the magnetic field gradient can be changed, the diffusion can be sampled using a three dimensional sampling pattern. If six different directions on a sphere $g^{(i)}$ are acquired leading to six signals $s^{(i)}, i \in \{1 \dots 6\}$ measured in addition to a base image $s^{(0)}$, a second order diffusion tensor can be reconstructed by solving the system of six equations

$$s^{(i)} = s^{(0)} e^{-bT_{jk}\mathbf{g_j^{(i)}}\mathbf{g_k^{(i)}}} \tag{1}$$

describing a symmetric second order tensor[1] $T_{jk} = T_{kj}$. Here $s^{(i)}$ is the signal intensity in presence of a magnetic field gradient and $s^{(0)}$ is the baseline image which is the signal intensity

[1] We use Einstein's summing convention in all equations where variables are summed up over same indices

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 124–135
Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

in absence of diffusion-sensitizing field gradients to which the remaining measurements are related. The parameter b is called b-factor or diffusion weighting factor which will be assumed to be a constant here. The influence of the b-value to the measurement has been studied for example by Frank [5] and Jones [9]. Usually, more gradient directions are used to smooth the data and equation 1 is solved using least squares fitting.

In addition to this approach, other techniques have been introduced to handle the additional information gained by sampling using more than six points, among these are q-Space imaging, higher angular resolution diffusion (tensor) imaging HARD(T)I and q-Ball imaging. While q-Space imaging [1] is difficult to measure and is prone to artefacts [13], q-Ball imaging needs a high number of gradient directions (about 120 to 300) [13]. HARDI is a technique using higher order tensors to represent diffusion patterns using higher angular resolution than second order diffusion tensor imaging while only a reasonable small amount of gradient directions is needed. The number of gradient directions is important because of its linear dependence on the measuring time. In clinical environments only ten to twenty minutes of scanning time are available resulting in six to thirty gradient directions using two or three images for averaging.

Concerning visualization, the ellipsoidal glyph is a rather simple but the best known visualization technique. It is an ellipsoid spanned by the scaled eigenvectors of the symmetric, positive definite second order tensor and may be interpreted as an isosurface of the density function of particles placed in a fluid after a certain diffusion time. In addition many other glyphs exist like the superquadric glyph presented by Kindlmann [10]. It presents a better technique providing a direction independent interpretation by reducing errors of introduced by visual artifacts. Unfortunately, the physical interpretation partly vanishes.

Tensor lines are a well known and widely used method for visualizing symmetric second order tensor fields which are applied in many different settings like mechanical engineering [3, 8] and medical imaging [14]. In MRT of the human brain, neural fibers hinder diffusion perpendicular to their course. Therefore tensor lines approximate the neural fiber structures found in the white matter of the brain. While second order tensors can represent only a single direction (the direction of the major eigenvector discarding its orientation), higher order functions are able to represent a higher angular resolution and thus a higher amount of directions inside the same volume element. This makes it possible to extract a higher amount of information from the scanned data compared to simple second order tensor approaches. Usually crossings are detected by looking at the neighboring voxels which reduces the absolute resolution of the data. As current diffusion tensor images still have a relatively low resolution (usually larger than $1 \times 1 \times 1mm^3$ compared to the fiber structures which are at a resolution of micrometers in diameter) and many fiber tracts span only across two or three voxels, it is important to work with the highest resolution of data available. This implies that analysis of data at voxel resolution or beyond is crucial for analyzing MRT data.

on one side of the equation. Free indices or indices on different sides of the equation lead to a system of equations.

2 Higher Order Tensors

A tensor of order (rank)[2] r and dimension d is a multilinear form mapping r d-dimensional vectors to a scalar:

$$T : (\mathbb{R}^d)^r \to \mathbb{R} \tag{2}$$

$$\left(\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(r)}\right) \to T_{i_1 \ldots i_r} \cdot \mathbf{v}^{(1)}_{i_1} \cdots \mathbf{v}^{(r)}_{i_r}. \tag{3}$$

When using the same normalized direction vector $\mathbf{g}$ (we call it gradient vector as used in the MRT nomenclature) a tensor can be interpreted as a scalar function defined on the sphere

$$f_T(\mathbf{g}) = T_{j_1 \ldots j_r} \cdot \mathbf{g}^{(i)}_{j_1} \cdots \mathbf{g}^{(i)}_{j_r} \tag{4}$$

There is an analogy between symmetric even order tensors and the symmetric spherical harmonics approach presented by Frank [6] which has been pointed out by Özerslan et al. [12]. This can be used as an alternative to the tensor representation. A discussion of the use of spherical harmonics in detail can also be found in Hlawitschka et al. [7]. Here we only want to emphasize the similarity of the spherical harmonic transform to the Fourier transform. Thus, higher order spherical harmonic basis functions contain higher frequency components. As spherical harmonics and the tensor functions f_T of same order r describe the same function space, the frequency information is contained in the tensors, too.

We compute a higher order tensor using the raw information $s^{(0)}$ and $s^{(n)}$ and map it to a tensor T of order r using

$$s^{(n)} = s^{(0)} e^{-bT_{i_1 \ldots i_r} \cdot \mathbf{g}_{i_1} \cdots \mathbf{g}_{i_r}} \tag{5}$$

To display this information, we use a generalization of Reynold's stress glyph that is defined by the surface

$$\mathbf{S} = \{\mathbf{p} \in \mathbb{R}^3 : \mathbf{p}_j = T_{i_1 \ldots i_r} \cdot \mathbf{v}_{i_1} \cdots \mathbf{v}_{i_r} \cdot \mathbf{v}_j \quad \wedge \quad \mathbf{v} \in S^2\}. \tag{6}$$

The usual color map for medical imaging indicating the direction of the largest expansion can be applied to this, too, as shown in Fig. 1. Fading out the color by anisotropy values is difficult for higher order tensors because anisotropy measures of higher order tensors can not be compared to those of second order tensors. This is due to the fact that higher order components are independent of lower order components. Color mapping on the surface function can be applied to strengthen the shape of the glyph. In addition, arrows can be drawn to improve visibility of local maxima as shown in Figures 2–5 and 8.

3 Higher Order Tensor Lines

Despite of the fact that glyphs give a good impression of the properties of a tensor defined at a certain position, no information is shown about its neighborhood. Therefore stream lines and

[2] We are ignoring the difference between contravariant and covariant indices here and use a fixed orthonormal coordinate system for the sake of readability. Therefore all tensor indices are lower indices and vector indices should be interpreted as upper indices to preserve mathematical correctness but are written as lower indices to simplify notation and to allow upper indices to be reused differently where needed.

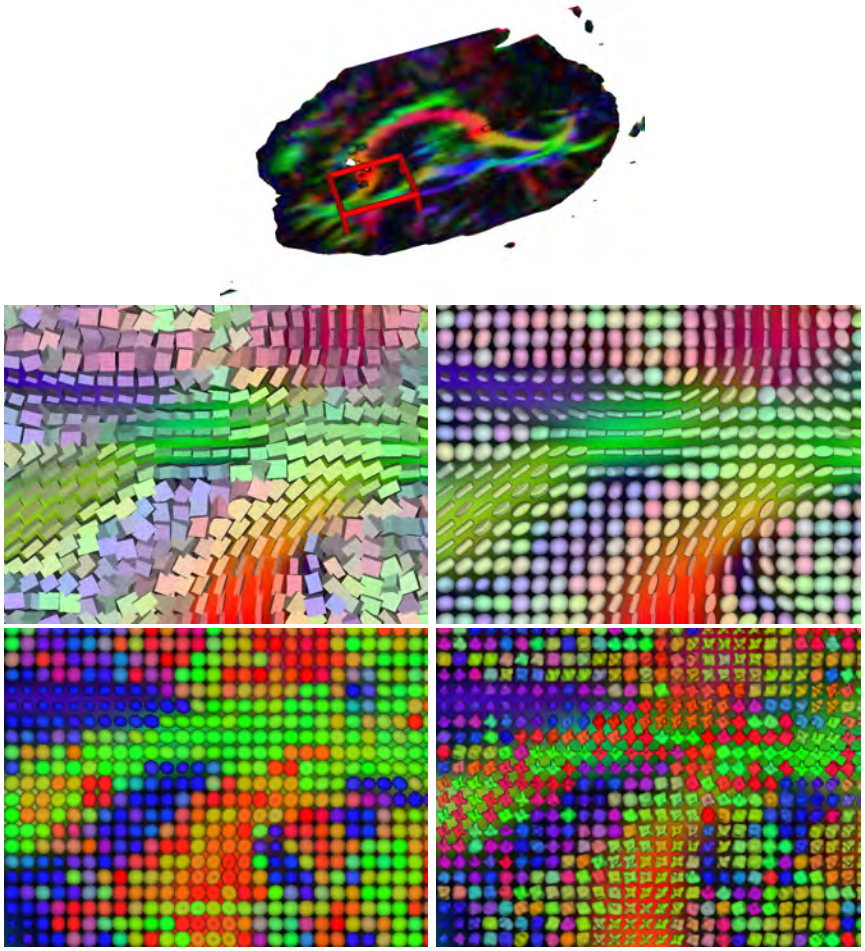


Figure 1 A comparison of different tensor glyphs in the area marked by the red box including parts of the forceps minor and the pyramidal tract. Middle: Box glyphs aligned to the eigenvectors (left) and Superquadric tensor glyphs (right). Bottom: second order Reynold's glyph (left) and its fourth order modification (right).

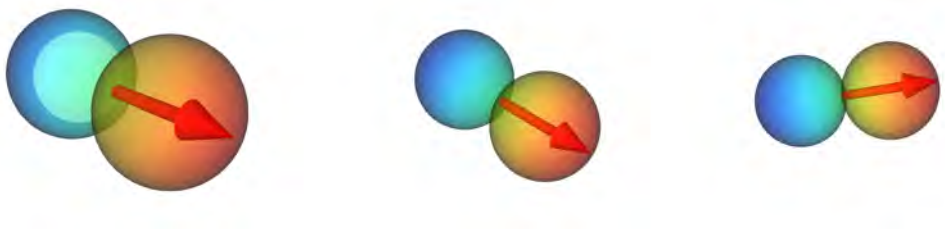


Figure 2 First order tensor are defined by the scalar product of the direction vector and the vector sampling the surface. It is obvious that lines can be drawn and that they correspond to streamlines. (The scaling of the sampling vector has been normalized for this drawing, blue values are negative.)

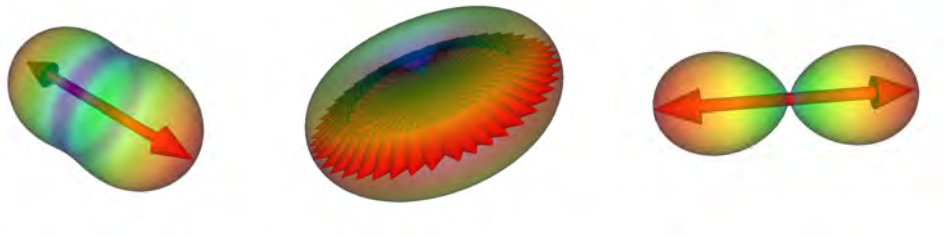


Figure 3 Second order tensor lines in general have one major eigendirection. They also are a subset of fourth order tensors.

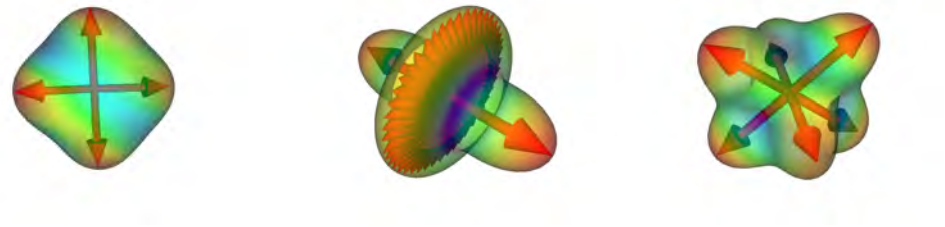


Figure 4 Symmetric, fourth order tensors may have more directions and have special degenerated cases like the one in the middle where a single major direction still exists.

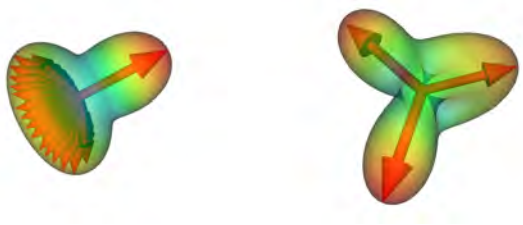


Figure 5 Our approach of tracking lines is also applicable to mixed order tensors (here zeroth and third order), but this is not investigated any further due to the lack of application.

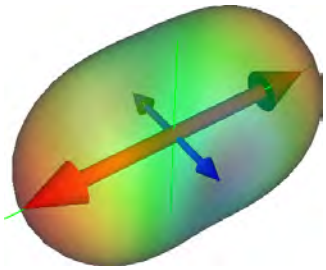


Figure 6 A second order Reynold's glyph with maxima and minima indicated by red and blue arrows resp. and its eigenvector decomposition (thin green lines).

tensor lines have been introduced to depict the information present in a certain neighborhood around a point of interest or – when using randomly seeded lines or lines seeded by complex algorithms determining good seed points – global information about the behavior of the vector or tensor field. Therefore, tensor lines have been used in many fields of visualization. Even though tracking of second order tensor lines reveals a reasonable visualization for medical images on the first view, it does not handle crossings of fibers at all due to its nature and underlying model of gaussian diffusion. Thus, the interconnections between the inner part of the corpus callosum and the outer areas of the corpus callosum are not handled correctly because they seem to be divided by the pyramidal tract. For a detailed description about the commissural fibers (neofibrae commissurales) from a neurological/topical point of view refer to Duus [4].

In second order tensor fields these techniques are based on the eigenvector decomposition of the matrix representation of second order tensors, especially symmetric, positive definite tensors which have three positive eigenvalues and three orthogonal eigenvectors. As there is no such decomposition for higher order tensors [11], we introduce a technique leading to the same results for stream lines and tensor lines that is applicable to higher order tensor fields, too.

A non degenerated symmetric second order tensor as shown in Fig. 6 has two distinct maxima, two minima and two saddle points which correspond to its eigenvector directions. We use this property to define a major tensor line as a line following these maxima.

Let Σ be the set of tensors of arbitrary order r and let

$$\mathbf{T} : \mathbb{R}^3 \supseteq U \rightarrow \Sigma \tag{7}$$

$$\mathbf{p} \rightarrow T(\mathbf{p}) \tag{8}$$

be a C^2 continuous tensor field. In the following, we study the corresponding function $f_{T(\mathbf{p})}$ at each position, i.e.

$$f_{T(\mathbf{p})} : S^2 \rightarrow \mathbb{R} \tag{9}$$

$$(\theta, \phi) \rightarrow f_{T(\mathbf{p})}(\theta, \phi) \tag{10}$$

so we have a function on the sphere at every position.

▶ **Definition 1.** We call a position $\mathbf{p} \in U$ *degenerated* if there is a position $(\theta, \phi) \in S^2$ where

$$\nabla_{S^2} f_{T(\mathbf{p})}(\theta, \phi) = \mathbf{0} \tag{11}$$

and

$$\det |\nabla^2_{S^2} f_{T(\mathbf{p})}(\theta, \phi)| = 0. \tag{12}$$

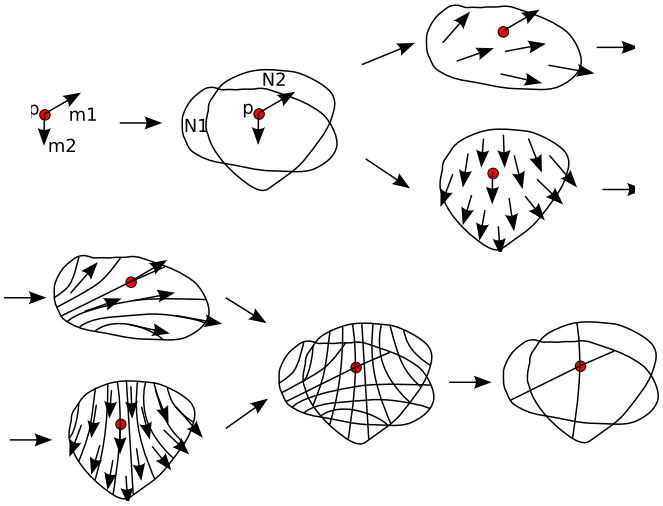


Figure 7 A tensor T at a position p in the data set with two maxima m1 and m2 provides two neighborhoods N1 and N2 around p. On both neighborhoods a C^1 continuous vector field is defined. Because the area contains no critical points, streamlines can be integrated everywhere in the neighborhood. Both sets of lines are combined to areas containing two, one or no streamlines. Usually only lines going through the point p are of interest.

(The name is well chosen because some tensor lines are not uniquely defined at these positions in accordance to the usual notion of degenerate points introduced by Delmarcelle and Hesselink [2]. Fig. 8 gives a visual impression of some of those tensor glyphs.) Usually, testing higher derivatives would lead to a more restrictive definition of degenerated points. There are special instable cases in which study of higher order derivatives would reveal, that what we call degenerated is not degenerated. For simplicity, we ignore these very rare cases in the following sections.

At a *regular* point (i.e. a point that is not degenerated) $\mathbf{q} \in U$, we have a finite number M of isolated maxima $m_1, \ldots, m_M = (\theta_1, \phi_1), \ldots, (\theta_M, \phi_M)$ of $f_T(\mathbf{q})$. Using the implicit function theorem, we obtain neighborhoods $U_1, \ldots, U_M \subset U$ and unique C^1 continuous functions

$$\mathbf{w}_m : N_m \rightarrow S^2 \tag{13}$$

$$\mathbf{p} \rightarrow \mathbf{w}_m(\mathbf{p}) = (\theta_m(\mathbf{p}), \phi_m(\mathbf{p})) \tag{14}$$

that parametrize the maxima $(\theta_m(\mathbf{p}), \phi_m(\mathbf{p}))$ in the neighborhood N_m, i.e. we can extract M C^1-continuous vector fields around $\mathbf{p}$ as shown in Fig. 7. Using these vector fields on N_m we define major arbitrary order tensor lines.

▶ **Definition 2.** A unique *(major) tensor line* in the tensor field $\mathbf{T}$ through the point $\mathbf{q}$ following a maximum m is a curve

$$\mathbf{x}_m : I_m \rightarrow N_m \tag{15}$$

$$t \rightarrow \mathbf{x}_m(t) \tag{16}$$

with

$$\mathbf{x}_m(0) = \mathbf{q} \tag{17}$$

and

$$\frac{\partial \mathbf{x}_m}{\partial t}(t) = \mathbf{w}_m(\mathbf{x}_m(t)). \tag{18}$$

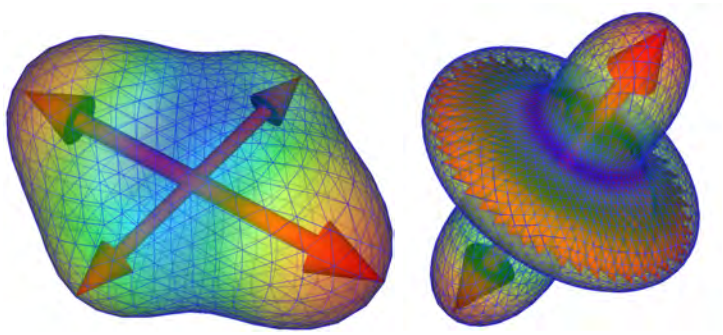


Figure 8 A fourth order tensor glyph showing two main directions of diffusion (left) and a degenerated fourth order glyph with a single main direction and a "degenerated ring" (right).

4 Properties of Higher Order Tensor Lines

Similar definitions can be given for minor tensor lines following the minima and medium tensor lines following the directions of saddle points of the functions f_T. This provides us with a framework of lines in arbitrary order tensor fields where crossing is a valid behavior of lines following different maxima in overlapping areas. Every parameterized line is uniquely defined by a position and an initial oriented direction. In symmetric tensor fields, two lines having the same direction but different orientation differ in their parameterization only by the relation of their parameters $t_1 = -t_2$. In the following sections, we will deal only with symmetric tensors. As the number of maxima on the function f_T is even because of its symmetry, we will speak of one (two, three...) direction when having two (four, six...) maxima of f_T.

Despite of the fact that our definition of higher order tensor lines is closely related to the definition of second order tensor lines it is independent of the order of the tensor. Thus, this definition can be applied to zeroth, first and second order tensors, too. Obviously, for a zeroth order tensor f_T is constant which is also the most important degenerate case in higher order tensor fields i.e. a completely isotropic tensor.

4.1 First Order Tensors

First order tensors (vectors) are simply build by the scalar product of the vector direction and the sampling direction v.

$$f_T = T_i v_i =< T, v >= \|T\|\|v\| \cos\gamma, \tag{19}$$

thus its maximum in the non degenerated case is in the same direction as the vector direction and our higher order tensor lines correspond to streamlines. A special case about asymmetric functions f_T is that there exists a minimum that can be tracked leading to an reverse parameterized line. In the case of first order tensor fields, there is only one minimum which describes backward integration of the streamline. The only degenerated case here is $T_i = 0$, thus $f_T = 0$.

4.2 Symmetric Second Order Tensors

For symmetric second order tensors, the maximum is in the direction of the largest eigenvector. This can be seen by decomposing T into $R^{-1}DR$ where R is a rotation of the eigenvector

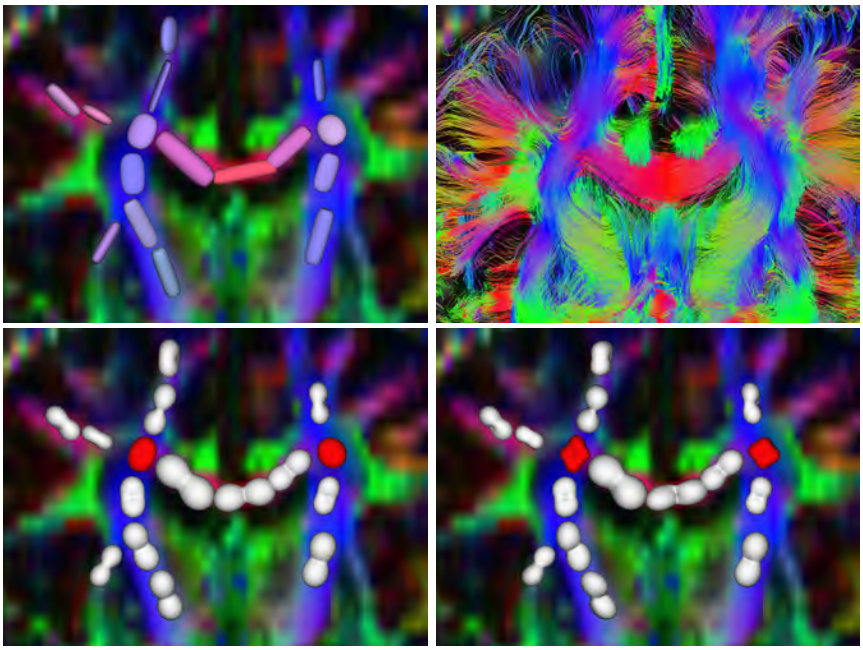


Figure 9 Slice through a human brain. Top left: Kindlmann's superquadric glyphs, right: Full brain tracking of second order tensor lines colored by direction. Bottom left: second order tensors displayed by surface glyphs. Bottom right: Fourth order glyphs shown in the same dataset. The red glyphs indicate crossings where the fourth order glyph reveals much more information than the corresponding second order glyph in the same data set. The data set has been provided by Max-Planck-Institute for Human Cognitive and Brain Sciences Leipzig.

basis of the tensor to the cartesian basis vectors and $D_{ii} = \lambda_i$ a diagonal tensor containing the eigenvalues.

For a vector $e^{(k)}$ along the k-th eigendirection of T, f_T can be written as

$$f_T(e^{(k)}) = T_{ij} e_i^{(k)} e_j^{(k)} = \lambda_1 e_j^{(k)} e_j^{(k)} = \lambda_1 \|e^{(k)}\|^2. \tag{20}$$

For any other direction, the vector v can be projected into the basis of the tensor which can be expressed by the rotation $\tilde{\mathbf{v}} = R\mathbf{v}$. Let $e^{(k)}$ now be the normalized eigenvectors of the tensor then

$$\begin{aligned} f_T(\mathbf{v}) &= f_T(\tilde{\mathbf{v}}_1 e^{(1)} + \tilde{\mathbf{v}}_2 e^{(2)} + \tilde{\mathbf{v}}_3 e^{(3)}) \\ &= T_{ij} \tilde{\mathbf{v}}_1 e_i^{(1)} \tilde{\mathbf{v}}_1 e_j^{(1)} + T_{k\ell} \tilde{\mathbf{v}}_2 e_k^{(2)} \tilde{\mathbf{v}}_1 e_\ell^{(2)} + T_{mn} \tilde{\mathbf{v}}_3 e_m^{(3)} \tilde{\mathbf{v}}_1 e_n^{(3)} \\ &= \lambda_1 \tilde{\mathbf{v}}_1^2 + \lambda_2 \tilde{\mathbf{v}}_2^2 + \lambda_3 \tilde{\mathbf{v}}_3^2 \end{aligned} \tag{21}$$

Obviously the maximum of f_T for a constant length of $\mathbf{v}$ is reached at $\tilde{\mathbf{v}}_1 \to max$ which is achieved by turning the vector $\mathbf{v}$ in the direction of the eigenvector of the largest eigenvalue λ_1. This shows that higher order tensor lines are the same lines as second order tensor lines.

Due to the fact that the neighborhoods are areas of smooth behavior, their borders have to be degenerated. Even though this seems obvious their calculation and mathematical analysis is still an open topic and will be a subject of further research.

5 Application to Real World Data Sets

We applied the method to several measured data sets of healthy volunteers. 36 gradient directions are measured in addition to the base image. The data has been converted to symmetric tensors using equation 4. No additional filtering or smoothing has been applied. The data set consists of a rectilinear grid of size $128 \times 128 \times 44$ with a voxel size of approximately $1.7 \times 1.7 \times 3.0 mm^3$. Lines are seeded at every grid point inside the brain or randomly in an region of interest. We tracked several lines in areas where crossings are assumed in medical literature, e.g. in Duus [4], most important the area close to the corpus callosum.

5.1 Second Order Tensor Lines

Second order tensor lines are used to compare our results to previous results. We seeded a second order tensor line at every position of the grid inside the brain where the fractional anisotropy is larger than 0.2. All lines were stopped when the FA reaches the threshold of 0.15. The resulting lines were filtered by a maximal length of about the diameter of the brain and maximal number of steps to prevent loops and a minimal length of 30mm to remove visual clutter from too short lines. Results of the tracking can be seen in Fig. 9. We further magnified the area where the corpus callosum and the pyramidal tract meet as described by Duus. This area is shown in Fig. 10. Due to limitations of second order tensors tensor lines from the bottom to the top (blue) split the image into two parts. This separatrix cannot be crossed by other lines which can be clearly seen in the figure.

5.2 Higher Order Tensor Lines

The higher order approach has been applied to the same data set. Again, no filtering has been applied. Fourth order tensors are reconstructed as described previously. Random seedpoints have been selected in a region of interest which is approximately the area of the assumed crossing and are marked by different colors than the lines themselves. Due to simplicity, a simple Euler approach with adaptive stepsize control has been used for the integration of lines. The result can be seen in Fig. 11. A comparison to the second order tensor lines that can be seen in Fig. 10 shows that the knowledge of physicians is much better represented by the higher order approach as a crossing of lines can be detected and a smaller amount of lines of the corpus callosum is deflected by the influence of the diffusion pattern of the pyramidal tract and the corona radiata.

6 Conclusion and Future Work

The theoretical basis of tracking higher order tensor lines has been presented. Proofs of equality to first order and symmetric second order lines have been indicated. Furthermore, we have shown that higher order tensor lines can be applied to noisy medical data sets acquired using diffusion weighted magnetic resonance imaging with a relatively small amount of gradient directions. There, well known crossings of the pyramidal tract and corpus callosum have been reconstructed that are not visible in second order tensor fields. A comparison to images found in medical literature reveals many similarities, like the crossing structure and their directions that can not be present in second order tensor fields. Further investigations have to be done relating the influence of noise in second and higher order tensor fields, describing the reliability of the tracking. Application to other higher order tensor data such as complex fourth order material tensors is still an open topic.

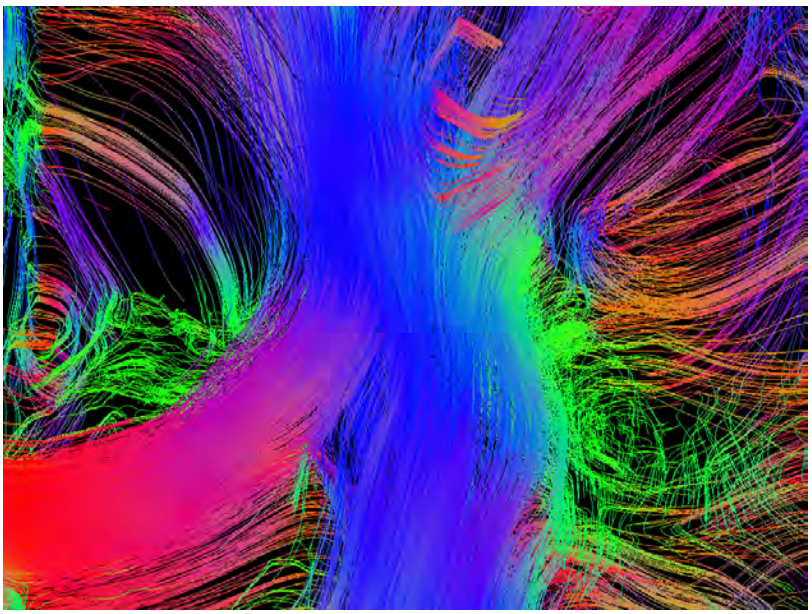


Figure 10 Crossings of the pyramidal tract and the corpus callosum shown using second order tensor lines in a measured data set of a young healthy volunteer. Even though a crossing is expected, second order tensors are not capable of displaying crossing behavior. Data set provided by Max-Planck-Institute for Human Cognitive and Brain Sciences Leipzig.

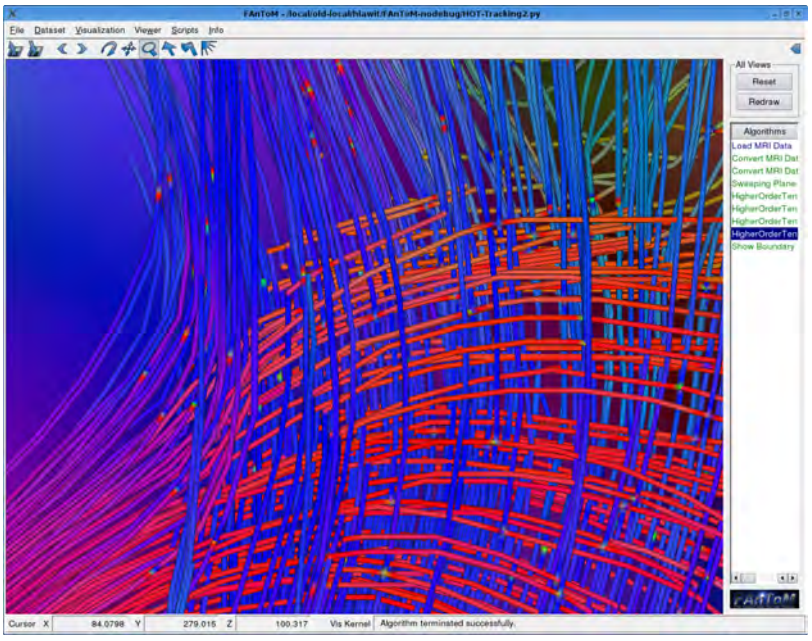


Figure 11 Similar view of the same data set as in Fig. 10. Crossings of the pyramidal tract and the corpus callosum painted as tubes. Colored points indicate the random seeding points of the lines. The data set has been provided by the Max-Planck-Institute for Human Cognitive and Brain Sciences Leipzig.

Acknowledgements

We thank our cooperation partners from the Max-Planck-Institute for Human Cognitive and Brain Sciences Leipzig especially Marc Tittgemeyer, Alfred Anwander, Thomas Knösche and Harald Möller for providing the data sets, for the fruitful discussions and for answering our questions relating MRT and neuro sciences in general. Further thanks go to Enrico Kaden and Davide Imperati for their useful hints relating the data sets. Finally we thank the ***FAnToM*** team for providing the framework for our implementation.

References

1 Paul T. Callaghan, D. MacGowan, K.J. Packer, and F.O. Zelaya. High resolution q-space imaging in porous structures. *Journal of Magnetic Resonance*, 177, 1990.

2 Thierry Delmarcelle and Lambertus Hesselink. Visualization of second order tensor fields and matrix data. In *Proceedings of IEEE Visualization 1992*, pages 316–323, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

3 Thierry Delmarcelle and Lambertus Hesselink. Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics and Application*, 13 (4)(4):25–33, July/August 1993.

4 Peter Duus. *Neurologisch-Topische Diagnostik*, volume 7. Thieme, Stuttgart, 2001.

5 Lawrence R. Frank. Anisotropy in high angular resolution diffusion-weighted MRI. *Magnetic Resonance in Medicine*, 45:935–939, 2001.

6 Lawrence R. Frank. Characterization of anisotropy in high angular resolution diffusion-weighted MRI. *Magnetic Resonance in Medicine*, 47:1083–1099, 2002.

7 Mario Hlawitschka and Gerik Scheuermann. HOT-lines — tracking lines in higher order tensor fields. In Cláudio T. Silva, Eduard Gröller, and Holly Rushmeier, editors, *Proceedings of IEEE Visualization 2005*, pages 27–34, October 2005.

8 Boris Jeremic, Gerik Scheuermann, Jan Frey, Zhaohui Yang, Bernd Hamann, Kenneth I. Joy, and Hans Hagen. Tensor visualization in computational geomechanics. *Int. J. Numer. Anal. Meth. Geomech.*, 26:925–944, 2002.

9 Derek K. Jones and Peter J. Basser. Squashing peanuts and smashing pumpkins: How noise distorts diffusion-weighted MR data. *Magnetic Resonance in Medicine*, 52:979–993, 2004.

10 Gordon Kindlmann. Superquadric tensor glyph. *Joint EUROGRAPHICS – IEEE TCVG Symposium on Visualization*, 2004.

11 Carla D. Moravitz Martin. Tensor decompositions workshop discussion notes. American institude of Mathemattics (AIM), July 2004.

12 Evren Özarslan and Thomas H. Mareci. Generalized diffusion tensor imaging and analytical relationships between diffusion tensor imaging and high angular resolution diffusion imaging. *Magnetic Resonance in Medicine*, 50:955–965, 2003.

13 David Solomon Tuch. Q-ball imaging. *Magnetic Resonance in Medicine*, pages 1358–1372, 2004.

14 Song Zhang, Ćagatay Demiralp, and David H. Laidlaw. Visualizing diffusion tensor MR images using streamtubes and streamsurfaces. *IEEE Transactions on Visualization and Computer Graphics TVCG*, 9:454–462, October-December 2003.

Illustrative Focus+Context Approaches in Interactive Volume Visualization

Stefan Bruckner[1], M. Eduard Gröller[1], Klaus Mueller[2], Bernhard Preim[3], and Deborah Silver[4]

1 **Institute of Computer Graphics and Algorithms, Vienna University of Technology**
{bruckner,groeller}@cg.tuwien.ac.at
2 **Center for Visual Computing, Computer Science, Stony Brook University**
mueller@cs.sunysb.edu
3 **Department of Simulation and Graphics, Otto-von-Guericke University of Magdeburg**
preim@isg.cs.uni-magdeburg.de
4 **Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey**
silver@ece.rutgers.edu

Abstract

Illustrative techniques are a new and exciting direction in visualization research. Traditional techniques which have been used by scientific illustrators for centuries are re-examined under the light of modern computer technology. In this paper, we discuss the use of the focus+context concept for the illustrative visualization of volumetric data. We give an overview of the state-of-the-art and discuss recent approaches which employ this concept in novel ways.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, J.3 Life and Medical Sciences

Keywords and phrases Illustrative Visualization, Volumetric Data

Digital Object Identifier 10.4230/DFU.SciViz.2010.136

1 Introduction

A considerable amount of research has been devoted to developing, improving and examining visualization techniques for scientific volume data. It has been shown that volume rendering can be successfully used to explore and analyze volumetric data sets in medicine, biology, engineering, and many other fields.

A recent trend in volume visualization is that researchers tend to use traditional illustrations as an inspiration for their work. As the domain of scientific illustration is based on centuries of experience in the depiction of complex volumetric structures, it represents a valuable source for visualization researchers. A common technique found in many traditional illustrations (see Figure 1) is referred to as focus+context in visualization literature. As there is often not enough space available to display all information in sufficient detail, the general idea is to emphasize regions of particular interest (focus) without completely removing other information important for orientation (context). Moreover, focus+context visualizations are not only motivated by space limitations but also by human visual perception. People are capable of simultaneously perceiving both local detail and global context [46]. Focus+context methods make it possible to show more detailed or targeted information and at the same

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 136–162

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

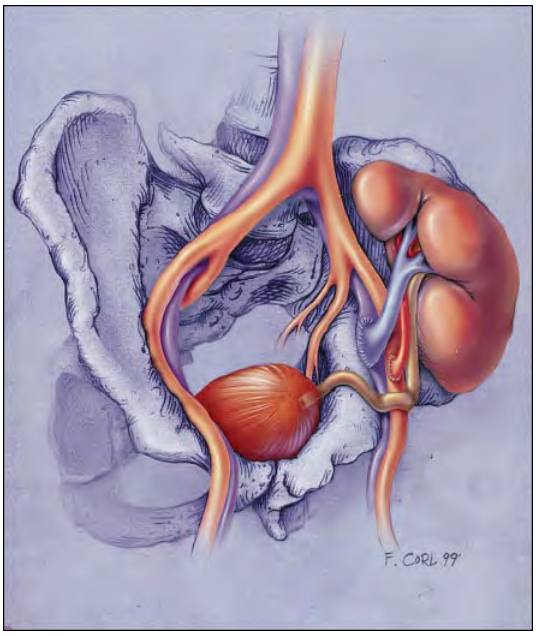


Figure 1 Medical illustration using focus+context – the image shows the position of a kidney after transplantation into the pelvic region. Bones are displayed in a stylized way while focus is emphasized by a more realistic rendering style (image courtesy of Frank M. Corl [17]).

time give users a sense of where in the data the zoomed-in, more detailed, or pointed out information is.

In connection with the advanced interaction possible in computer-based visualization the focus+context concept offers additional advantages. As the focus can be modified interactively, it serves as a means to explore complex data. This paper reviews recent approaches which employ different kinds of focus+context techniques for improved illustrative visualization of volumetric data. Section 2 gives an overview of previous work in the area. In Section 3 we discuss a framework for distortion-based focus+context volume visualization. It employs traditional (i.e. physically plausible) as well as arbitrary distortions for highlighting structures in volumetric data. Section 4 presents deformation techniques which allow a user to manipulate the data for improved comprehension in Section 4. Next, in Section 5, we focus on concepts used in the design of a volume-based illustration system which aims to produce visualizations with the aesthetic appeal of traditional illustrations. In Section 6 we discuss how illustrative focus+context visualization can be employed in medical applications for surgery planning. Finally, the paper is concluded in Section 7.

While distinct approaches, the individual parts of this paper have many similarities. They all employ the concept of focus+context to prevent information overload and allow the user to concentrate on certain structures of interest. Traditional scientific illustration is used as a source of inspiration and adapted to the additional degrees of freedom offered by computer-based visualization. Finally, all approaches use the capabilities of graphics hardware to allow interactive navigation and interaction.

2 Related Work

Focus+context approaches have been heavily employed in information visualization. Many approaches are based on spatial distortion, for instance fish-eye views [21], hyperbolic trees [31] or the document lens [44], and others [29] (see [34] for a more detailed overview). Viewpoint-dependent distortion of 3D data [8, 9] highlights regions of interest by dedicating more space to them. Other methods, for example tool glass and magic lenses [3], allow the display of additional data dimensions on demand. Cue methods [30] enhance the visualization by assigning visual cues to certain objects so that they are more prominent to the viewer without hiding the context.

While the area of non-photorealistic rendering [25, 49] is more concerned with imitating artistic styles in an automated way, illustrative visualization goes one step further and tries to apply these techniques selectively to enhance visual comprehension. Illustrative visualization can be seen as a fusion of the focus+context concept and non-photorealistic rendering. The visual abstraction is realized at two basic levels: stylized depiction (low-level abstraction) deals with how objects should be presented, while smart visibility (high-level abstraction) is concerned with what should be visible and recognizable.

The inherent complexity of volumetric data has lead to a considerable amount of research in illustrative techniques for volume visualization. Most approaches tend to combine both levels of abstractions and do not have an explicit steering mechanism. Levoy [35] was the first to propose modulation of opacity using the magnitude of the local gradient. This is an effective way to enhance surfaces in volume rendering, as homogeneous regions are suppressed. Based on this idea, Rheingans and Ebert [43] present several illustrative techniques which enhance features and add depth and orientation cues. They also propose to locally apply these methods for regional enhancement. Using similar methods, Lu et al. [37] developed an interactive volume illustration system that simulates traditional stipple drawing. Csébfalvi et al. [20] visualize object contours based on the magnitude of local gradients as well as on the angle between viewing direction and gradient vector using depth-shaded maximum intensity projection. Lum and Ma [38] present a hardware-accelerated approach for high-quality non-photorealistic rendering of volume data. They also suggest the use of lighting transfer functions [39] for object enhancement. The concept of two-level volume rendering, proposed by Hauser et al. [26], allows focus+context visualization of volume data. Different rendering styles, such as direct volume rendering and maximum intensity projection, are used to emphasize objects of interest while still displaying the remaining data as context. Zhou et al. [60] use a focal-region-based rendering approach which depicts context data using a different rendering technique. They also propose the use of distance to emphasize and de-emphasize different regions [59]. The distance from a focal point is used to directly modulate the opacity at each sample position. Tapenbeck et al. [51] employ distance-based transfer function based on the distance to an object (rather than a focal point). Levoy and Whitaker [36] perform adaptive resolution volume rendering based on gaze direction. Cignoni et al. [13] provide the MagicSphere metaphor to visualize 3D data with the MultiRes filter. Wei et al. [58] apply fisheye views to particle track volume data using nonlinear magnification functions. LaMar et al. [33] integrate a 3D magnification lens with a hardware-texture based volume renderer. Cohen and Brodlie [15] magnify features by generating a new volume using inverse distortion functions.

Some recent approaches explicitly distinguish between low- and high-level abstraction. Viola et al. [54, 55] map an importance function which specifies the relevance of different structures within the volume data to appropriate levels-of-spareness which control object

appearance. Different importance compositing strategies control object visibility. Svakhine et al. [50] employ illustration motifs to control the appearance of objects at varying degrees of complexity.

3 Distortion-based Focus Enhancement Using Volumetric Lenses

Distortion-based focus enhancement with lenses has a long history, with the first reported uses going all the way back to the Greeks, Arabs, and Romans. Greek philosopher Aristophanes (448 BC-380 BC) already knew that glass could be used as a magnifying lens, but it was not until roughly 150 AD that Ptolemy discovered the basic rules of light diffraction and wrote extensively on this subject. Back then, magnifying glasses were mainly used as a reading aid by the literate class. For example, Roman tragedian Seneca (4 BC-AD 65) is said to have read "all the books in Rome" by looking through a glass globe of water. A thousand years later, far-sighted monks employed segments of glass spheres which they laid against reading material to magnify the letters, and the development of these "reading stones" was based on the theories of the Arabic mathematician Alhazen (roughly 1000 AD). This basic invention was then later refined by Venetian glass blowers, who constructed lenses that could be held in a frame in front of the eye instead of directly on the reading material. While these first spectacles were intended for use by one eye only, the idea to frame two ground glasses with wood or horn into a single binocular unit was introduced in the 13th century. Eventually, in 1268, Roger Bacon made the first known scientific commentary on lenses for vision correction, and he is generally credited with the invention of the magnifying glass, and perhaps with the foundation of the field of optics as a whole. Since then these basic lens optics have experienced a great revolution, on a vast order of scales, ranging from the magnification of individual biological cells to the scanning of the far-out cosmos. In many applications, and definitely in the case of spectacles, preserving context is a strong necessity, in order to maintain and provide ease of navigation. Preserving context usually means that the resolution of the visual information presented is highest in the foveal center (the focus), and then falls off towards the periphery in some smooth fashion, without performing any clipping within the viewing area. This is usually the case in lens-based physical optics. In more recent years, the laws of lens optics have also found application in the virtual world, using computers to implement these general concepts. In the beginning, the emphasis was on realistic simulations of the physical laws, and a great number of computer graphics papers were written to that effect [47]. However, while computer graphics was mainly concerned with the realistic simulation of optical effects in possibly very complex scenarios, the field of information and data visualization has been more in line with the original intentions of the ancient "reading stones" concept, that is, the magnification of objects of interest for better perception of their detail. It was quickly discovered that virtual (computerized) lenses are no longer constrained by the physical laws of optics, allowing the liberal use of these concepts in creative ways.

The approach summarized in this section of the paper (see [57] for more detail) seeks to generalize distortion functions and to make them interactive via implementation on graphics hardware. The latter allows their use within an engaging volume exploration tool, where a real-time response to user actions is a must – a property that is also expected, and in fact taken for granted, in physical lenses. As was mentioned above, in addition to physics-based lens optics, software lenses also allow the derivation and implementation of functionalities that do not have counterparts in physical optics, or at least are hard to fabricate. We provide volumetric lenses in both categories. In the former, we devise a set of lenses that tune their

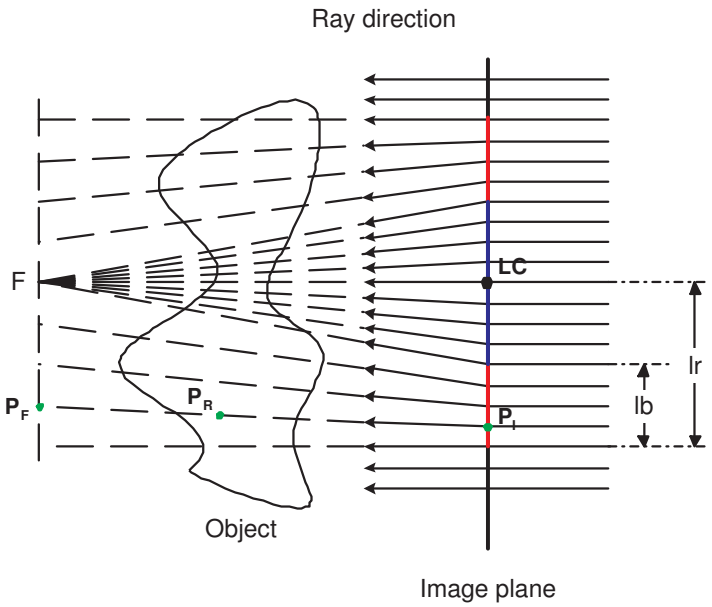


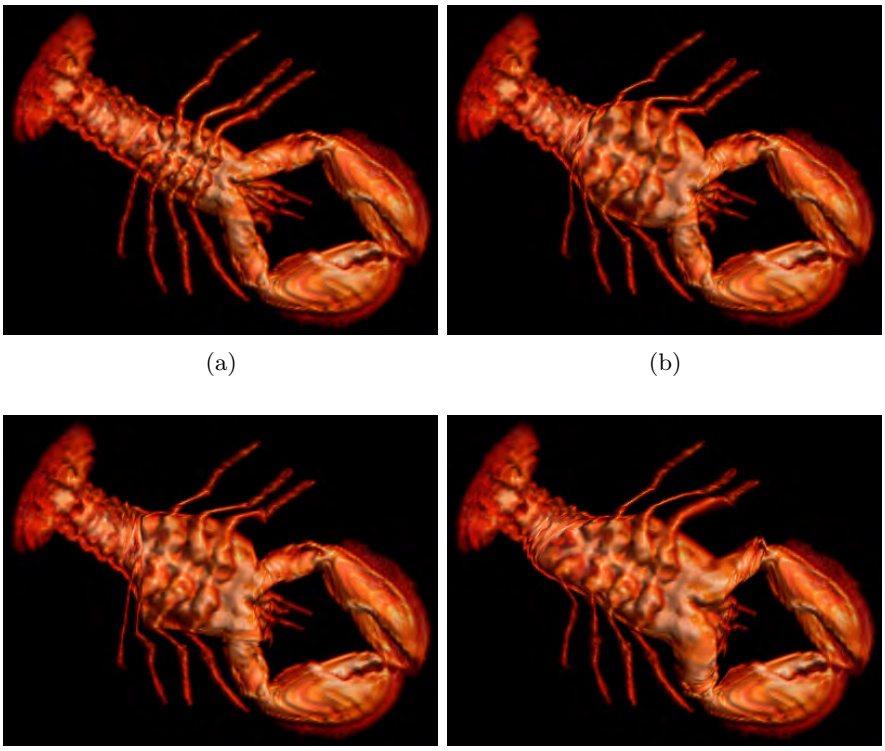
Figure 2 Illustration of the general lens mechanism.

geometry to the underlying feature semantics, while in the latter we allow the user to design the distortion function with a free-hand tool. In the following sections, we will describe the key components of our framework, and we conclude with pointers to possible extensions of our framework.

3.1 Virtual Lens Optics in a Volumetric Environment

Our lens aims to provide smooth transitions between focal and peripheral regions, with no clipping. Further, it aims to keep the lens effects local. Thus, the additional screen area dedicated to the focal (magnified) regions must be taken away from the peripheral (minified) regions. In sampling theory terms, this means that the focal volume regions are oversampled, while the peripheral volume regions are undersampled. The latter requires proper anti-aliasing during image generation. Figure 2 illustrates these concepts, using a raycasting rendering paradigm. Here, the blue line segment on the image plane represents the magnification part of the lens, LC is its center point and F is the virtual focal point. When orthogonal incident rays hit the image plane, in the area of the focal region, the ray directions are modified and go through F. Therefore, a ray cone is formed between the lens and F, and the object regions within this cone are rendered in an enlarged area on the image plane. The peripheral regions (to the left and right of the focal region) are represented by the red line segments on the image plane with width lb and are rendered at reduced resolution, while image regions outside $+-lr$ (the radius of the lens) appear at normal magnification. Thus, the paths of the rays traversing the peripheral region form the smooth transition between normal and focal region.

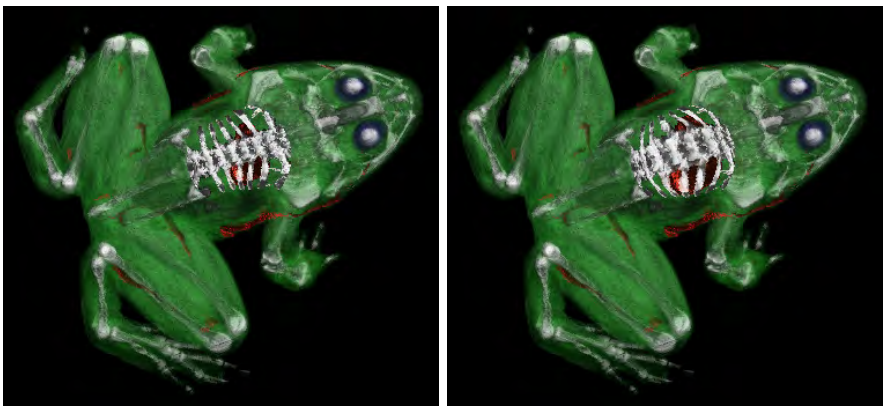
This general framework enables the design of lenses with arbitrary shapes. Figure 3a shows the original volume rendering obtained with no lens, and Figure 3b-d are renderings obtained using a circular, square, and arbitrary-shaped lens, respectively.



(a) (b)

(c) (d)

Figure 3 Magnifier volume renderings with (a) No lens, (b) Circular lens, (c) Square lens, (d) Arbitrary-shaped lens.



(a) (b)

Figure 4 Magnifier volume renderings for the bone feature in a segmented frog dataset. (a) and (b) are renderings without and with magnification under a circular lens.

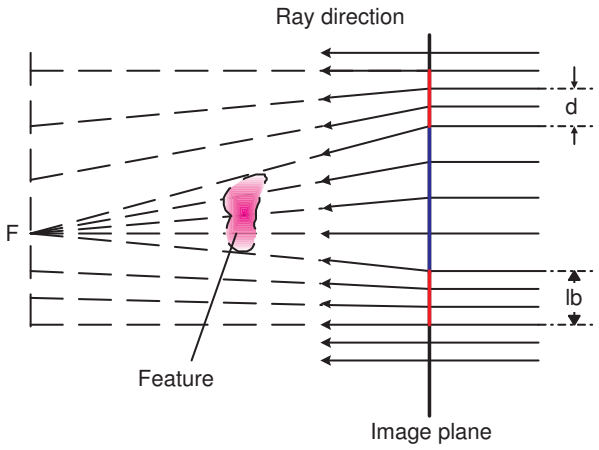


Figure 5 Feature-based lens illustration.

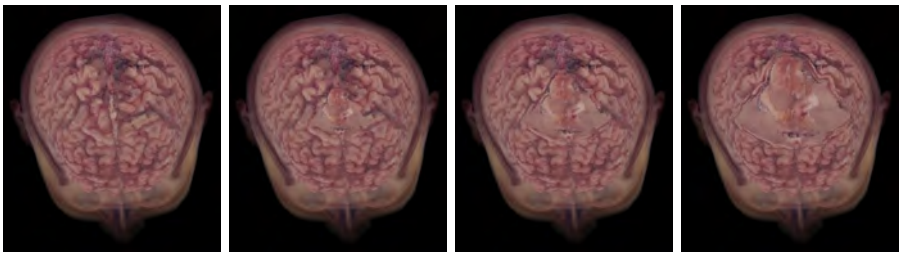
A segmentation of the dataset enables magnification with semantics. Figure 4 was generated by allowing the rays under the lens to penetrate the frog's skin, magnifying the skeleton underneath.

3.2 Feature-centric Lens

The feature-centric lens provides users a means to highlight portions of interest in volume objects by dedicating more screen area to them. This also promotes a more accurate and differentiated understanding of these features, since fine detail is enlarged. In this lens, the shape of its magnification portion (and that of the surrounding transition region) is defined dynamically by the shape of the features (represented by available segmentation information) in the dataset (see Figure 5 for an illustration). Here, whether an incident ray changes direction depends on the distribution of the feature. Thus the direction of each ray needs to be determined dynamically. Transition regions are also used here to retain the spatial context of the features. For each ray orthogonally incident upon the image plane, the new direction is computed as follows. Assuming all rays changed directions to the focal point F,

- if a ray passes through the feature, then its new direction is pointing to F.
- if the ray does not pass through the feature but is inside the transition region on the image plane, the distance d (see Figure 5) from its entry point to the boundary of the feature-projected area is calculated. This distance is used to compute the new direction.
- otherwise, the ray continues along its original direction.

The transition region is determined by a boundary of certain width around the feature. For this we first project the feature onto the image plane and then fill all interior points with a constant value. This region will be magnified using the over-sampling scheme defined before. The transition region field can then be marked (on the image plane) using a distance transform and the values be used to determine the direction of the rays. Alternatively, one may also determine the ray direction vector by finding the distance of the starting position to the closest neighbor in the projected feature region. We have used a GPU-accelerated search circle approach for this, with the transition region width lb being the circle's maximal radius. Figure 6 shows some rendering results for a color volume dataset, in which a user-selected



(a) (b) (c) (d)

Figure 6 Feature-based lens volume renderings for a segmented human brain color volume dataset. (a) without specifying any feature of interest, (b) with a feature of interest, which is not magnified and appears too small to be seen clearly. From (c) to (d) the magnification factor increases.

feature is magnified and the other objects near that feature are compressed. Figure 6a shows the skin of the brain. Figure 6b shows an interior structure of the brain, without rendering other features which occlude this structure, while the magnified structures are shown in Figure 6c and d.

3.3 Free-form Lens Optics

Our framework also allows the design of arbitrary lens functions, using a free-form drawing tool. Figure 7 (top) shows an example. Here, the vertical axis (the height of the curve) indicates the (instantaneous) sampling rate, which is the reciprocal of the local sample distance on the image plane. The horizontal axis indicates the distance from the lens center. The higher the sampling rate, the greater is the number of rays per unit area and the magnification. Since magnification in one screen area (sampling rate > 0) must be balanced with minification in another (sampling rate < 0), the total curve integrals above and below the x-axis must approximately match. This is also illustrated in Figure 7 (bottom), where we observe that the rays shot into the object are denser in the center region of the lens and become coarser towards the boundary. Various sampling functions can be adopted to define various volumetric lenses and to achieve different volume rendering results.

Figure 8 shows some results object with the free-form lens, comparing it with the results obtained with no lens and a standard magnification lens, respectively. The toes of the foot are shown rendered with different magnification effects. The difference between Figure 8b and 8c is mainly caused by the different magnification factor distributions on the lenses. For the standard lens, the magnification factors for points projecting into the magnification region and having the same distance to the image plane are the same. Therefore, objects at the same depth are magnified uniformly. However, for the lens with cubic sampling function, the factor is the highest on the lens center and decreases gradually towards the lens boundary. Objects with projections closer to the lens center are magnified with higher magnification factors. Along any ray, the factor remains the same for different depth values.

3.4 Extensions

There are several extensions that fit well into the presented framework. For example, it would be relatively straightforward to extend the current mipmap-based zooming capabilities to more sophisticated multi-resolution data, where the data appearing under magnification comes from a different data source. This could either be a modality acquiring data at a

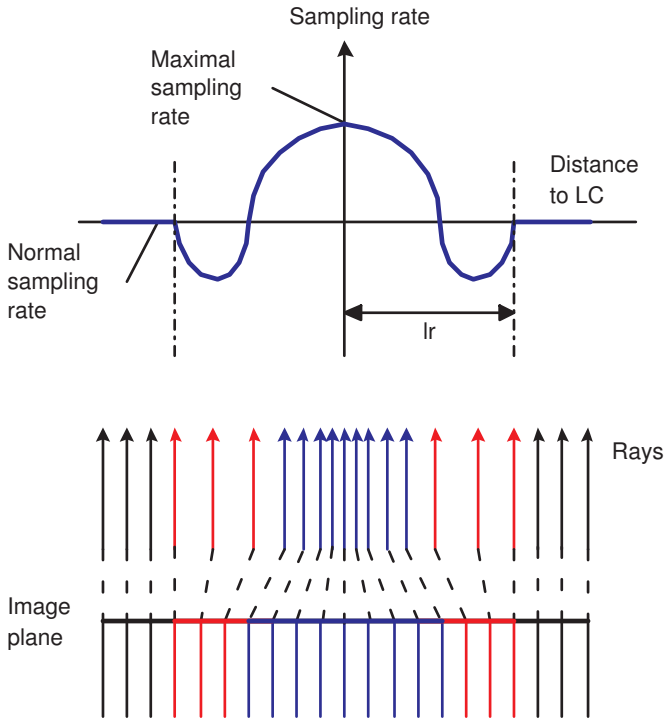


Figure 7 Sampling-rate-based lens illustration.

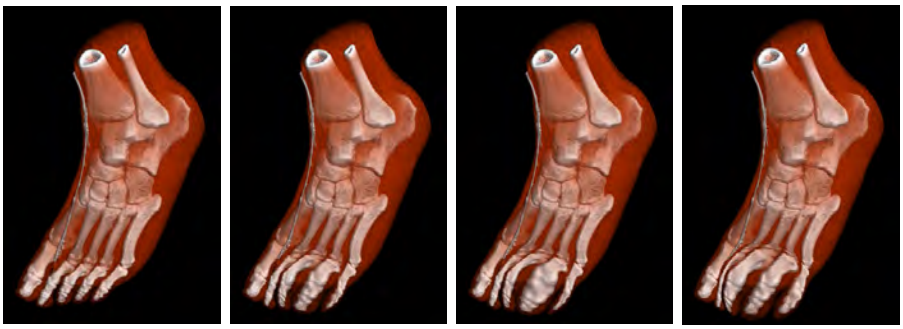


Figure 8 Comparing volume renderings with (a) no lens, (b) normal magnification lenses (c) cubic lens optics sampling function (maximal sampling rate/normal sampling rate = 3), and (d) an arbitrary lens optics sampling function.

resolution appropriate for the current local magnification rate, or a texture synthesis process that generates these data from high-resolution data swatches on the fly [56]. Finally, the lens may be generalized to provide Superman-vision capabilities – a magic lens to see the underlying uncertainties associated with the data, another channel in a multi-modal dataset, a segmentation result, or some semantic annotations that go with the data. It may fuse these views together, controlled by the user.

4 Interactive Manipulation of Volumetric Objects

The purpose of visualization is to "gain insight by using our visual machinery" [1] and "Underlying the concept of visualization is the idea that an observer can build a mental model, the visual attributes of which represent data attributes in a definable manner" [45]. The tools we currently have in 3D data visualization to help building the mental model include real-time rendering, rotation, slicing, transfer functions, segmentations and many novel focus+context renderings. However, the effective exploration of volumetric data is still a challenging task, especially for complex volumetric datasets with convoluted structures. With the prevalence of 3D imaging in all fields, such as for physical therapy, psychology, security and screening, archeology, e-commerce, etc. it is important to explore methodologies which can enhance our comprehension of the underlying structure.

Interestingly, the use of physical representations in rapid prototyping (layered manufacturing) has capitalized on this fact. Claims include: 2D screen displays do not always provide an intuitive representation of 3D geometry; unusual or deformed geometry may be hard to comprehend on-screen; the integration of different modalities is hard to visualize; and the planning of complex 3D manipulation from 2D images can be difficult [10]. As opposed to virtual prototypes (display), "physical prototypes bring in a completely new interactive modality – the sense of handling an object" [2]. It can be much easier to learn about complex 3D shapes by holding the actual objects.

In this section, we describe a more *active* approach to visualization which allows the user to manipulate the data. The purpose is to allow the viewer to explore the data for comprehension not necessarily to simulate reality. Techniques or operations on the data that exemplify this approach include bend, move/re-pose, peel, pull, sweep, roll/unroll, cut, retract, and split. All of these descriptions are verbs, symbolizing an action on the dataset. Some examples of this type of visualization are shown in Figure 9. This type of visualization is common in surgical education and simulation, medical illustration and other types of illustration. Below we briefly describe some of the different types of manipulations which are useful, and discuss how to achieve these effects on 3D datasets.

4.1 Spatial Transfer Functions

In medical illustration, one commonly sees peel-away effects simulating surgical cuts and other types of surgical procedures. Therefore, what is desired is a deformation-like procedure which can model cuts or splits in a volume. Surgical simulation packages can handle cuts, but they are usually specialized to one model. Other physically based deformation methodologies exist (see [11] for a review) and most focus on deforming the surface without modeling cuts. For visualization and illustration purposes it is not clear that the manipulations must be physically based. Most related is the work of [40] where surgical-like operators were defined on volumetric datasets.

A *spatial transfer function* [12, 27] defines a geometrical transformation of the scalar values of volumetric models to allow different effects such as splitting, squeezing and sweeping. The

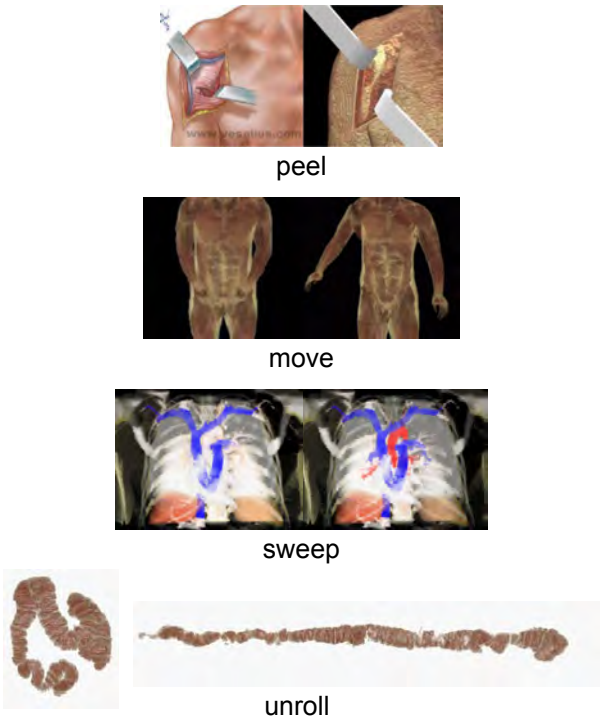


Figure 9 Different manipulations on a volume: peel, move, seep, unroll. Images are from [28, 48, 23, 19].

spatial transfer function is able to handle discontinuities in the rendering method which other more direct data manipulation methodologies have trouble with. The surgical cut shown in Figure 9 is rendered with a spatial transfer function. The volumetric model with the spatial transfer function is rendered by first computing the transformation on the bounding volume, and mapping back to find the actual volumetric values. These can then be composited using a standard ray casting approach. Details can be found in [28].

4.2 Curve-Skeleton Decomposition

To achieve the bend, move/re-pose, and sweep, a *proxy geometry* can be used. The proxy geometry allows the user to easily specify a transformation on the data. One such proxy geometry which is useful is a curve-skeleton. A curve-skeleton is a 1D line-like representation of the object (sometimes referred to as a centerline). A skeleton also is a natural decomposition of many objects and provides a simple way to specify a path (e.g., for virtual navigation). A skeleton of a volumetric object is a useful shape abstraction that captures the essential topology of an object. It also has a cognitive basis in shape comprehension. Motion is also traditionally specified in computer graphics using a skeleton. (Therefore, all of the motion capture available to computer graphics can be available for volumes as well.) The skeleton can be extracted by using a variety of methods (see [18] for more information). Once the skeleton has been obtained, the joints can be chosen [22]. The joints define the areas where bending can occur. Each skeletal segment defines a bounding cuboid about a logical segment of the volume, as seen in Figure 10. The width of the cuboid can be determined from a distance field computed on the volume. The subdivision of the cuboids can be determined

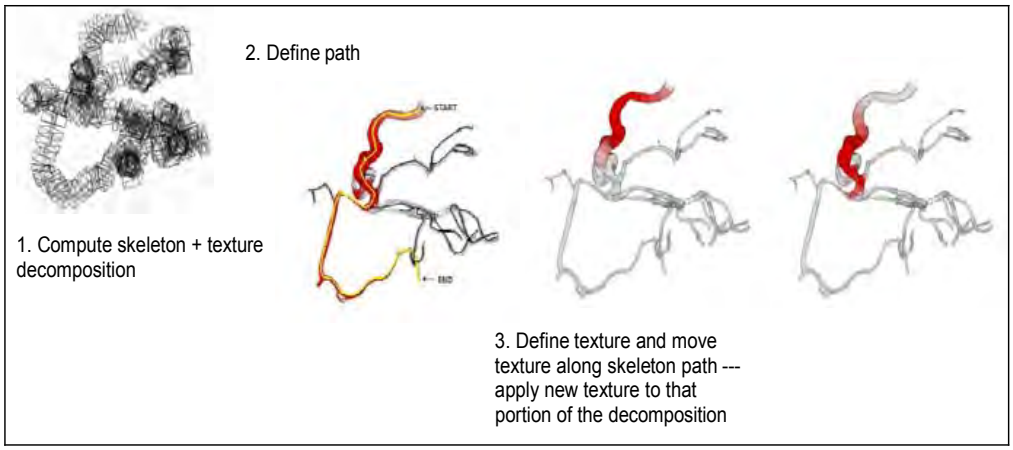
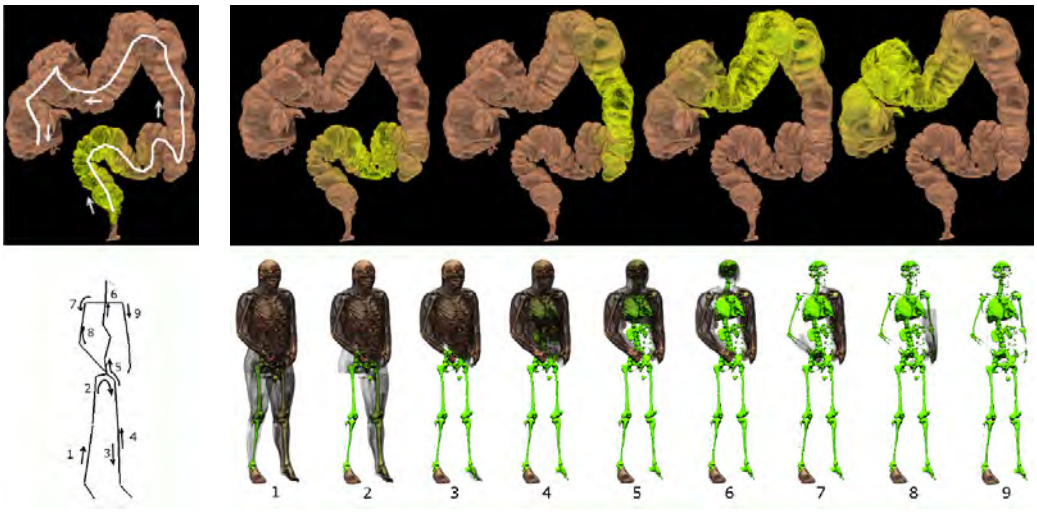


Figure 10 Block based decomposition of a volume along a skeleton path. The skeleton is divided into small cube-like regions. Each region can be sliced and then rendered. By specifying small regions, different colors, transfer functions, or general transformations can be applied independently. An effect simulating blood flow is shown by moving a texture along a specified path.

based upon the requirements of the visualization (described below). This geometry, though a coarse approximation to the actual shape, suffices for reconstructing most shapes and provides a simple geometry for fast rendering. The boxes are also good for fuzzy bounded volumes, which may not have a definite boundary. An example of the cuboid structure about a central axis is shown for the aneurysm dataset in Figure 10.

The skeleton/cuboids supports both kinematic manipulation and sweeping. One can grab the skeleton and move it as shown in two images from Figure 9. In the first, the hands of the visible man are moved away from the torso to allow better viewing of the torso area. In the second, a volumetric colon is unrolled (its skeleton is straightened). The manipulation can be performed by explicitly computing a new volume in the reposed position [23] or interactively using a technique which only renders the manipulated volume [48]. For interactive manipulation, the cuboids are transformed about the joint, and each cuboid is sliced along the viewport. The slices are mapped back to the original texture to determine the appropriate color value. The texture-mapped polygons are then composited back-to-front which volume renders the deformed volume. Interpolation is done between two end planes of the adjoining bounding boxes which essentially covers the joint area with a stretched volume. The advantage of using this approach is that it requires a minimum of geometric processing and is therefore very fast. If the volume is rotated, the underlying geometry has to be re-sliced and composited. Since each cuboid is sliced independently, it is necessary to sort the sliced polygons along the view direction from back to front. This is achieved in a two-pass rendering algorithm with the aid of a data structure that indexes polygons with respect to their depth coordinate.

The skeleton decomposition also supports selective rendering [48] and sweeps [19]. Selective rendering is where a portion of the dataset is rendered with a different transfer function. This allows one to highlight different parts of a volume for more effective focus+context viewing. By adding motion to the selective rendering, we can create a *swept* representation where a dataset is traversed along a specified path. This is commonly used in medical illustration to produce animations, highlight features or enhance the rendering of a dataset. Techniques for navigation of datasets are used in virtual colonoscopy, where the userís viewpoint is traversed



(a) (b)

Figure 11 Swept based volume rendering. The effect simulates flow through a volume and enables focus+context views. (a) shows the skeleton path (b) is the traversal. See [19] for more examples and a movie.

along a path. Such techniques can be described as inside-out visualizations of the dataset. Here, we are doing an outside-in visualization, where the exploration is enabled by moving a transfer function, in addition to independent control of the userís viewpoint. It is naturally a focus+context technique, as it focuses the viewers gaze on the area being swept while still showing the entire object. It is similar to using a highlighter pen to emphasize parts of an object. An example of this technique used in medical illustration to show blood flow can be seen in [5].

4.3 Dataset Traversal

Traversing (explicitly or mentally) a complex datasets seems to be an essential part of understanding 3D shape and a skeleton or sweep structure can aid in this process. When rendering volumes as 3D textures, transfer functions are usually applied as a lookup color table. This table defines the color and transparency associated with each density value of the volume. The skeletal-based block decomposition allows us to apply a different transfer function to each cuboid of the decomposition along a path at a particular time. A traversal path is selected, and each segment of the path is highlighted at a different time. For rendering, each cuboid of the decomposition is rendered as textured slices. When a slice is being rendered, the proper transfer function is found (i.e., if that slice is part of the segment being highlighted). Since all of the slices are composited together, any number of highlights can be used and the combinations are solved on-the-fly. This creates a feeling of "sweeping out the structure", similar to lighting the structure from within. Two examples of swept visualization can be seen in Figure 11. In the first, parts of a colon are highlighted, and in the second the visible man dataset is traversed while changing the rendering parameters (from bone to skin). Figure 10 shows a swept volume simulating blood flow. More details of the algorithm are given in [19].

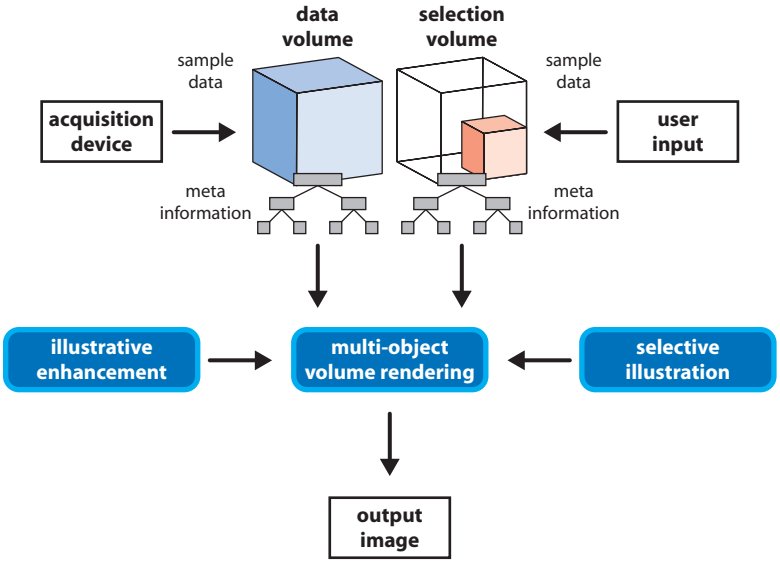


Figure 12 Conceptual overview of our direct volume illustration environment.

5 Interactive Design of Illustrations from Volume Data

In this section we discuss several concepts used in the design of VolumeShop [7], a system for interactive generation of illustrations directly from volume data. The advantages of such a system are manifold: Firstly, the whole process of creating an illustration is accelerated. Different illustration methods and techniques can be explored interactively. It is easy to change the rendering style of a whole illustration – a process that would otherwise require a complete redrawing. Moreover, the research process is greatly simplified. Provided that the object to be depicted is available as a volumetric data set, it can be displayed with high accuracy. Based on this data, the illustrator can select which features he wants to emphasize or present in a less detailed way. Illustration templates can be stored and reapplied to other data sets. This allows for the fast generation of customized illustrations which depict, for instance, a specific pathology. Finally, the illustration becomes more than a mere image. Interactive illustrations can be designed where a user can select different objects of interest and change the viewpoint.

The architecture of VolumeShop discriminates between two basic types of volumes: data volumes and selection volumes. A data volume stores the actual scalar field, for example acquired by a CT scanner. A selection volume specifies a particular structure of interest in a corresponding data volume. It stores real values in the range [0,1] where zero means "not selected" and one means "fully selected". While both multiple data and selection volumes can be defined, only one pair is active at a time. At the heart of the system lies a multi-object volume rendering algorithm which is responsible for the concurrent visualization of multiple user-defined volumetric objects. It makes use of illustrative enhancement methods and selective illustration techniques defining the visual appearance of objects. A conceptual overview of the system is given is Figure 12.

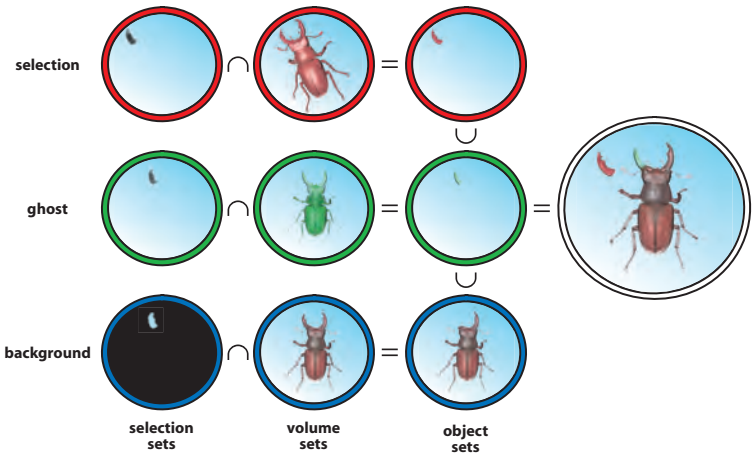


Figure 13 Overview of the basic multi-object combination process for background, ghost, and selection: the intersection between selection sets and volume sets results in object sets which are then combined.

5.1 Multi-Object Volume Rendering

When illustrating a volumetric data set, we want to enable interactive selection and emphasis of specific features. The user should be able to specify a region of interest which can be highlighted and transformed, similar to common image editing applications. We also want to permit arbitrary intersections between objects and control how the intersection regions are visualized.

VolumeShop's approach identifies three different objects for the interaction with a volumetric data set: a *selection* is a user-defined focus region, the *ghost* corresponds to the selection at its original location, and the *background* is the remaining volumetric object. A transformation can be applied to the selection, e.g., the user can move, rotate, or scale this object. While the concept of background and selection is used in nearly every graphical user interface, ghosts normally exist, if at all, only implicitly. The approach uses fuzzy set arithmetic to derive the selection, ghost, and background objects (objects sets) as an intersection of the selection volumes (selection sets) and the opacity transfer function specified for the data volumes (volume sets), as illustrated in Figure 13.

Additionally, the user is supplied with control over the appearance of regions of intersection. Frequently, for example, illustrators emphasize inter-penetrating objects when they are important for the intent of the illustration. Two dimensional intersection transfer functions are employed for this purpose. An intersection transfer function specifies the color and opacity at a resample location based on the scalar volume of the volumetric objects present at that location. Per definition background and ghost never intersect. The selection, however, can intersect either the background, the ghost, or both. The intersection transfer functions can be used to control the color and opacity in the region of intersection between two objects based on the scalar values of both objects. VolumeShop provides a default setting which is an opacity-weighted average between the one-dimensional color transfer functions of the two respective objects (background and selection, or ghost and selection). Furthermore, there are several presets where the opacity is computed from the one-dimensional opacity transfer functions by one of the compositing operators derived by Porter and Duff [42]. The color can be specified arbitrarily. Additionally, the user can paint on the two-dimensional function

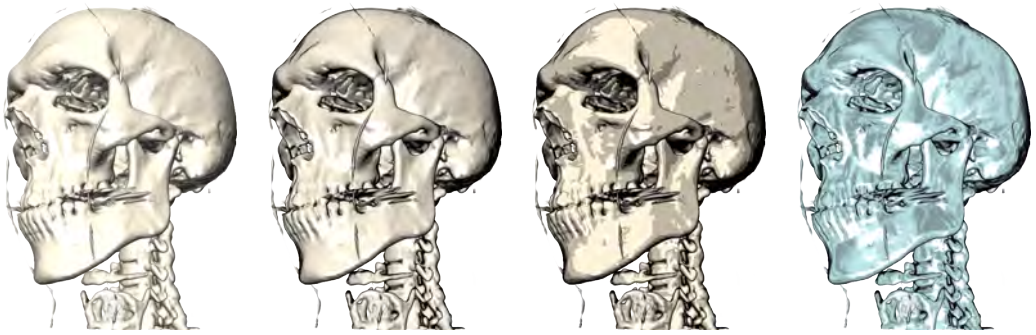
Figure 14 Using intersection transfer functions to illustrate implant placement in the maxilla. As the selection (green) is moved into the ghost (faint red), the intersection transfer function causes it to be displayed in blue.

using a gaussian brush to highlight specific scalar ranges. Figure 14 shows an example where the ghost/selection intersection transfer function is used to illustrate the placement of an implant in the maxilla. This kind of emphasis is not only useful for the final illustration, but can act as a kind of implicit visual collision detection during its design.

5.2 Illustrative Enhancement

Illustration is closely related to non-photorealistic rendering methods, many of which attempt to mimic artistic styles and techniques. VolumeShop uses a simple approach which integrates several presented models and is thus well-suited for a volume illustration system. Most illumination models use information about the angle between normal, light vector and viewing vector to determine the lighting intensity. In volume rendering, the directional derivative of the volumetric function, the gradient, is commonly used to approximate the surface normal. Additionally, the gradient magnitude is used to characterize the "surfaceness" of a point; high gradient magnitudes correspond to surface-like structures while low gradient magnitudes identify rather homogeneous regions. Numerous distinct approaches have been presented that use these quantities in different combinations to achieve a wide variety of effects. In order to integrate many of these models, VolumeShop uses a two-dimensional lighting transfer function for shading objects. The arguments of this function are the dot product between the normalized gradient $\hat{N}$ and the normalized light vector $\hat{L}$ and the dot product between the normalized gradient and the normalized half-way vector $\hat{H}$, where $\hat{H}$ is the normalized sum of $\hat{L}$ and the normalized view vector $\hat{V}$. A two-dimensional lookup table stores the ambient, diffuse, and specular lighting contributions for every $\hat{N} \cdot \hat{L}$ and $\hat{N} \cdot \hat{H}$ pair. Additionally, a fourth component used for opacity enhancement is stored.

We use the terms "ambient", "diffuse", and "specular" to illustrate the simple correspondence in case of Phong-Blinn lighting. However, the semantics of these components are defined by the model used for generation of the lighting transfer function. Thus, a lighting transfer function might use these terms to achieve effects completely unrelated to ambient, diffuse, and specular lighting contributions. This approach allows different illustrative shading models to be evaluated at constant costs. For example, contour lines are commonly realized by using a dark color where the dot product between gradient and view vector $\hat{N} \cdot \hat{V}$ approaches zero, i.e., these two vectors are nearly orthogonal. We can thus create a lighting transfer function where we set ambient, diffuse and specular components to zero where $\hat{N} \cdot \hat{L} \approx 2(\hat{N} \cdot \hat{H})$. Other methods, such as cartoon shading [14] or metal shading [24] can be realized in a straight-forward manner and combined with effects like contour enhancement. Figure 15 shows an image rendered using four different lighting transfer functions.



(a) (b) (c) (d)

Figure 15 The same data set rendered with four different lighting transfer functions (the lighting transfer function for each image is displayed in the lower left corner - ambient, diffuse, specular, and opacity enhancement components are encoded in the red, green, blue, and alpha channel, respectively). (a) Standard Phong-Blinn lighting. (b) Phong-Blinn lighting with contour enhancement. (c) Cartoon shading with contour enhancement. (d) Metal shading with contour enhancement.

5.3 Selective Illustration

Selective illustration techniques are methods which aim to emphasize specific user-defined features in a data set using visual conventions commonly employed by human illustrators. They are closely related to focus+context approaches frequently found in information visualization.

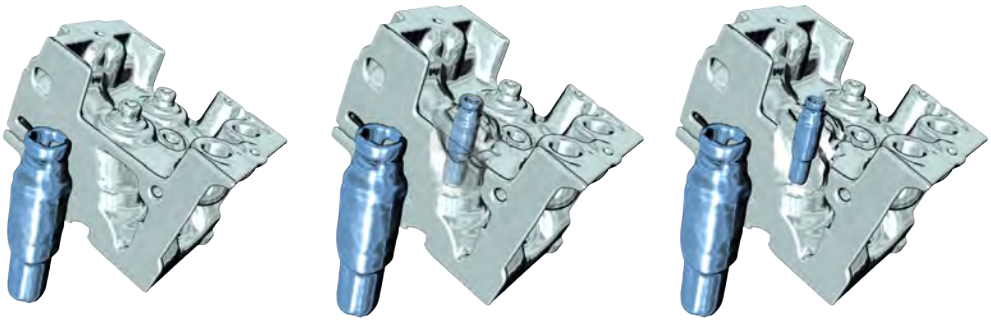
5.3.1 Cutaways and Ghosting

Cutaways (also referred to as cut-away views) are an important tool commonly employed by illustrators to display specific features occluded by other objects. The occluding object is cut out to reveal the structure of interest. Viola et. al. [54] introduced importance-driven volume rendering, a general framework for determining which object is to be cut by using an importance function. VolumeShop's simplified three-object setup allows a static definition of this importance function, which enables us to skip costly importance compositing and thus allows for an efficient implementation. Cutaways are only performed on the background and can be independently defined for ghost and selection.

Ghosting [6] refers to a technique which is frequently used in conjunction with cutaways. Instead of removing the occluding regions completely, opacity is selectively reduced in a way which attempts to preserve features such as edges. This tends to aid mental reconstruction of these structures and generally gives a better impression of the spatial location of the object in focus. The user can smoothly control the degree of ghosting from no ghosting (opacity is not reduced at all) to full cutaway view (occluding structures are completely suppressed) as shown in Figure 16.

5.3.2 Visual Conventions and Interaction

As the selection can undergo a user-defined transformation there are a number of possibilities for combining the effects of transfer functions, cutaways and ghosting, and spatial displacement. In its simplest form, this can be used to illustrate the removal or insertion of an object. Furthermore, "magic views" on a structure of interest can be generated, where the object is displayed using a different degree of detail, orientation, or rendering style. Illustrators commonly employ certain visual conventions to indicate the role of an object in their works.



(a) (b) (c)

Figure 16 Different degrees of ghosting - from no ghosting (a) to full cutaway (c).

(a) (b)

Figure 17 Using different artistic visual conventions. (a) Illustrating a tumor resection procedure using an automatically generated arrow. (b) Detailed depiction of a hand bone using a fan.

VolumeShop provides the user with different kinds of visual enhancements inspired by these conventions.

For example, arrows normally suggest that an object actually has been moved during the illustrated process (e.g., in the context of a surgical procedure) or that an object needs to be inserted at a certain location (e.g., in assembly instructions). Analogously, VolumeShop employs arrows to depict the translation between ghost and selection, i.e., the arrow is automatically drawn from the object's original position to its current location. To avoid very short arrows in case the selection and the ghost project to nearby positions in image space, we use the screen-space depth difference to control the curvature of the arrow. This leads to the kind of bent arrows frequently found in illustrations. Figure 17 (a) shows an example for the use of arrows.

Another metaphor used are "fans". A fan is a connected pair of shapes, such as rectangles or circles, used to indicate a more detailed or alternative depiction of a structure. It can be easily constructed by connecting the screen-space bounding rectangles of ghost and selection. In combination with cutaways and ghosting, this type of enhancement can lead to very expressive visualizations, depicting, for example, two different representations of the same object (see Figure 17 (b)).

Apart from controlling visual appearance, it is useful to provide different interaction types based on the role of an object in the illustration. For example, the user can "pin" down

the current selection, i.e. its on-screen location will remain static, but it is still affected by rotations. A rotation of the viewpoint causes the same relative rotation of the object. This can be used to generate a special view which always shows the part of an object facing away from the viewer in the background object.

5.4 Conclusion

The use of an optimized GPU volume rendering algorithm with multi-object capabilities (see [7] for more details) allows us to provide a responsive interface for interactive generation of volume-based illustration using the techniques described in this section. VolumeShop is an open architecture and supports extensions using a plug-in mechanism. For more information (including a downloadable version) see: `http://www.cg.tuwien.ac.at/volumeshop`.

6 Illustrative Visualization for Neck Dissection Planning

In this section, we discuss how conventional and illustrative rendering techniques might be employed to support a particular surgical intervention: neck dissection planning. Neck dissection planning poses challenging visualization problems due to the enormous density of crucial anatomic structures: Muscles, vascular structures, and nerves share the same small space. This discussion is based on an ongoing research project and the experiences with planning 20 neck dissections based on CT datasets ([32] and [53]).

6.1 Medical Background

Neck dissections are carried out for patients with malignant tumors in the head and neck region. These surgical procedures are necessary because the majority of the patients develops lymph node metastases in the neck region. The extent of the intervention depends on the occurrence and location of enlarged (and probably) malignant lymph nodes. In particular, the infiltration of muscles, nerves or blood vessels determine the surgical strategy. If for example the *A. carotis interna* is infiltrated, the patient is regarded as not resectable. Visualization techniques should be developed to support decisions regarding the surgical strategy.

6.2 Conventional Surgical Planning

Surgical planning is carried out by means of 2D slices. Computer support allows to browse quickly through the slices, to change brightness and contrast and to perform measurements. 3D renderings are rarely used and many surgeons are not convinced of the additional value of 3D renderings at all. This attitude has serious arguments: in 2D slices each and every voxel is visible – it can be selected and its intensity value can be inquired. Instead, 3D visualizations are primarily used to give an overview. Since conventional surgical planning relies on 2D slices, it is a good strategy to include 2D slices and the related manipulation techniques in advanced surgical planning systems. With this strategy, surgeons can plan their interventions as they did it before and can use the advanced techniques additionally. The most benefit can be achieved if 2D and 3D visualizations are carefully synchronized, e.g. with respect to the selection and emphasis of an object (Figure 18).

6.3 Advanced Surgical Planning

Advanced surgical planning requires reliable segmentation results. Segmentation of the relevant structures is a challenging task and an area of ongoing research. In our case study,

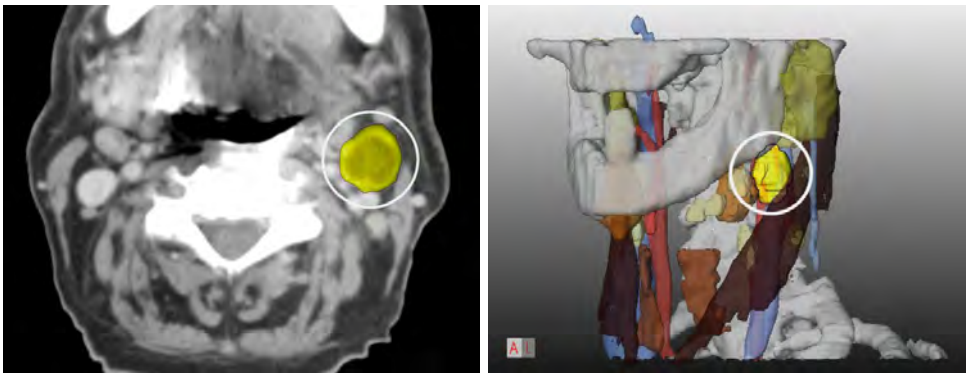


Figure 18 The lymph node emphasized in the 3D visualization is simultaneously emphasized in the original slices.

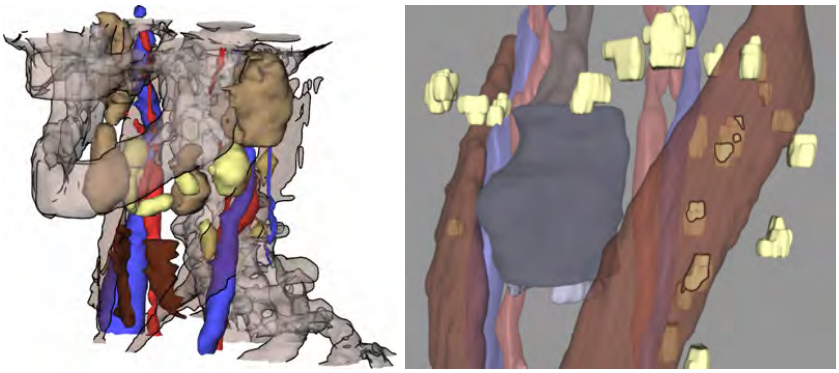


Figure 19 Left: Illustrative rendering for neck dissection planning. Silhouettes are generated for the bones which serve as anatomic context. Right: Silhouette rendering reveals lymph nodes which touch and potentially infiltrate a critical structure.

the segmentation of all relevant structures is accomplished with NeckVision, a dedicated software assistant [16].

Illustrative Rendering. Silhouette rendering is employed for two purposes. The obvious use is to indicate the context objects, such as bones (Figure 19, left). In addition, silhouettes may be used to discriminate two classes of objects; those which exhibit a certain feature (or are more “interesting") are rendered with silhouettes enabled whereas the remaining objects are drawn without silhouettes. In neck surgery planning, many lymph nodes have to be explored by the user. In particular, lymph nodes which are enlarged and touch a critical structure are essential and thus rendered with silhouettes (Figure 19, right). Surgical users regard this a substantial help since otherwise it is not recognizable whether the lymphnode is (only) close to a critical structure or touches it. The combination of silhouette-, surface- and volume rendering is accomplished with a scenegraph-based approach [52].

Approximative Rendering of Nervs. For some structures, such as nerves, a complete segmentation is not possible. Nerves are very small compared to the spatial resolution of the data. Therefore, a single voxel contains nerve tissue and other adjacent tissue resulting in an intensity value which is hard to distinguish from its surrounding. As a consequence, only in some slices a nerve could be identified at all. Nevertheless, the rough course of the nerves

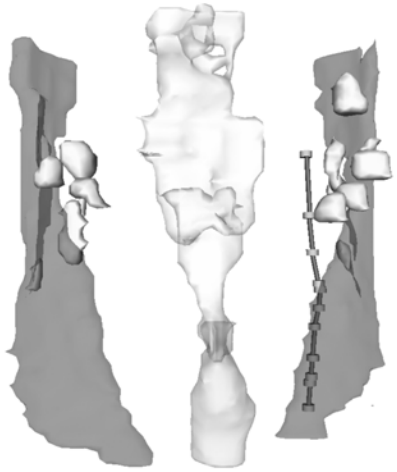


Figure 20 Approximate visualization of N. facialis for neck dissection planning. In the lower portion, the density of disks is higher which implies a more reliable visualization.

is essential for surgical planning. Anatomic experience shows that nerves proceed almost linearly and do not deviate strongly from the straight connection between positions found in some slices. Since it is important to prevent the injury of nerves, approximate visualizations should be generated where the segmented portions are emphasized and the part in between is reconstructed as linear connection. The emphasis of the segmented portions is carried out with small cylindrical disks (see Figure 20).

6.4 Enhancing Slice-based Visualizations

In the following, we describe how slice-based visualizations can be enhanced to give an overview on anatomic structures. This description is motivated by the importance of slice-based views for surgery planning and is based on [53].

For an overview of the segmented structures in a slice-based visualization, it is essential to present the relative position of structures in the current slice as well as their positions within the whole set of slices. In the following, we refer to the slice number as the z-dimension. The visualization problem that occurs here is similar to time scheduling. In this area, various techniques have been developed to visualize data entries and their temporal relations. Graphical overviews should present appointments distinguishable from each other and the temporal relations between them (see for example, the LifeLines project [41]). Translated to slice-based visualization, the interval of slices (z_{min}, z_{max}) of the segmented structures corresponds to the lengths of appointments.

Similar to temporal overviews in time scheduling, we attach a narrow frame next to the cross sectional image that represents the overall extent of slices in the volume data set. The top and bottom boundary of the frame correspond to the top and bottom slice of the dataset. Each segmented structure is displayed as a bar at the equivalent vertical position inside this frame. The vertical extent of the bar represents the interval (z_{min}, z_{max}) for each structure. We refer to this combination of bars as LiftChart and regard it as a widget which provides interactive facilities to locate structures and slices. The LiftChart widget can be used for interaction and navigation. The horizontal slice indicator is operated like a normal scrollbar and moves through the slices. If the mouse is placed over a particular bar, information about

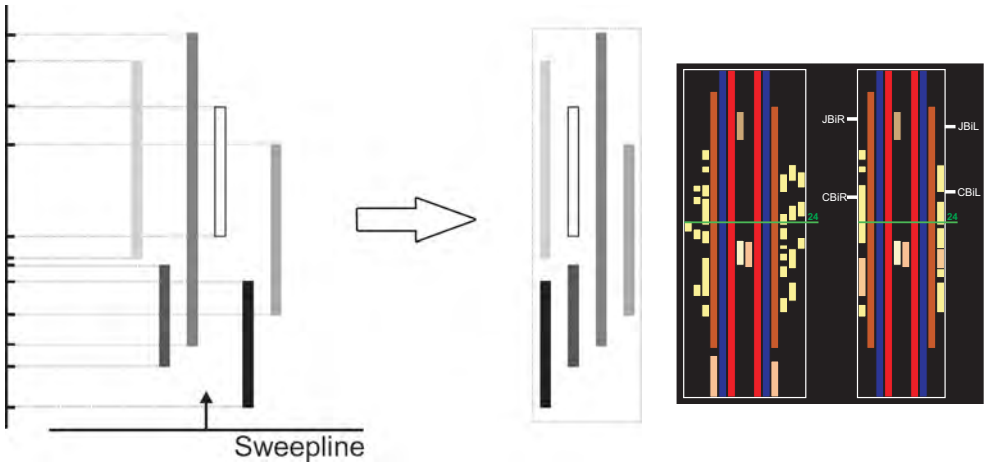


Figure 21 Left: The Sweepline moves from bottom to top and places the next available bar at the leftmost unused column. Middle: Each anatomic structure is represented by one bar. The LiftChart is divided in three parts: one part for structures on the left and on the right side each and one part for structures in the middle. Right: Lymphnodes are aggregated in one column. Additional landmarks serve as orientation aids.

the underlying anatomic structure is shown.

In order to optimize the LiftChart's use of screen space, the bars are ordered with a Sweepline algorithm. The slices are processed from bottom to top and for each anatomic structure a bar is drawn in the leftmost available column. If a structure ends, the respective column is freed again and can hold the bar of a new structure starting farther above (Figure 21, left).

The LiftChart enhances the recognition of relative positions of structures in the volume dataset. To simplify the correlation between the slice view and the LiftChart, the color and style of the bars should correspond with the color and style of the structures displayed in the slice view. The colors represent different categories: Lymph nodes (yellow), Muscles (brown), veins (blue), arteries (red) and the lung (skin-colored). The green line denotes the current slice. The currently displayed slice is depicted by a horizontal line and annotated with the slice number. To visualize not only the z-distribution of structures, but also their horizontal position, several arrangements of the bars have been explored (Figure 21). In the simplest form, each anatomic structure is represented by one bar. Since some anatomic structures have a defined side, the LiftChart may be divided in three parts: one part for structures on the left and on the right side each and one part for structures in the middle (see the left image of Figure 21). In particular, the separation in lymph nodes located at the left and right side is motivated by the surgical strategies (left and/or right-sided surgery).

It is also possible to group bars which belong to the same category of anatomical structures to minimize the horizontal extent of the widget (see Figure 21, right, where the lymph nodes are aggregated into one column). Furthermore, landmarks for orientation in the dataset may be displayed. As an example relevant for neck dissections, in the right image of Figure 21, the bifurcations of the *Vena Jugularis* (JBiL/JBiR) and *Arteria Carotis* (CBiL/CBiR) are indicated.

Slice-based Visualization of Safety Margins. Safety margins are essential for pre-operative planning and intraoperative navigation. To prevent damage to structures at risk, the distances of the surgical tool to such structures have to be carefully observed during

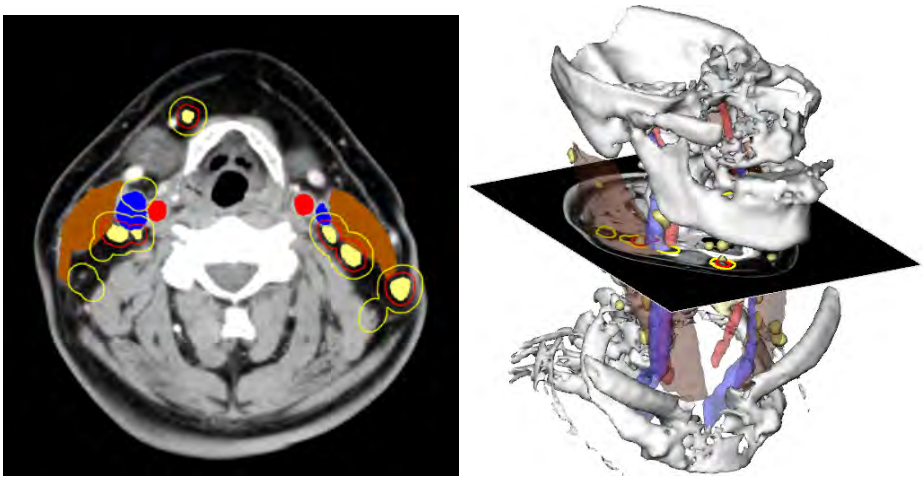


Figure 22 Depicting safety margins around pathologic lymph nodes as special halos on the current slice. Left and right are swapped in the slice view, because the viewing direction is from bottom to top. Left image: the safety margins of $2mm$ (red) and $5mm$ (yellow) are shown as the user would see them in the slice view. Right image: the position of the slice with the displayed structures is shown for comparison.

the surgery. Halos (in the original sense of the word) can convey this distance information. Therefore, for all structures at risk an Euclidean distance transform [4] is performed and the resulting distance information is overlaid on the slice image. We considered color-coding the distance information but rejected this idea, since a color map conveys too much information not relevant for the surgical strategy. Depicting important distance thresholds as halos by drawing isolines representing 2 and $5mm$ distances reduces the information to a few categories which are easy to interpret (Figure 22).

6.5 Discussion

From an applications point of view, illustrative renderings target only a portion of the overall problem. Whether or not all relevant lymph nodes are correctly delineated is probably more important than the details of their visualization. Valuable computer support for surgical planning requires high-quality image acquisition, reliable and fast image analysis techniques *and* comprehensible visualizations. The combination of 2D and 3D renderings is essential for the acceptance of computer-supported surgical planning systems.

The use of illustrative techniques for neck dissection planning is based on discussions with clinicians. Although illustrative techniques are not wide-spread in surgical planning, our research indicates, that they have a potential to improve surgical planning. The need for illustrative techniques will likely increase since more and more information is available preoperatively. The development of illustrative techniques should be directed to support the integrated visualization of these different sources of information. The great advantage of using illustrative techniques is the additional freedom to fine-tune visualizations with respect to task-specific needs. The major drawback is that additional effort is required to select appropriate techniques and parameters. These steps need to be strongly supported since the time for surgical planning remains severely restricted.

7 Conclusion

In this paper we have presented illustrative techniques for the visualization of volumetric data. We have shown that the concept of focus+context is a particularly useful metaphor for dealing volume data due to their inherent complexity.

Lens-based distortion is a powerful framework for the exploration of volume data even if no additional information is available. When segmentation data is present, feature-based lenses can be used to enhance fine details in an intuitive way. The "hands-on" approach to volume visualization presented in Section 4 is a natural way for examining volume data. Three-dimensional interaction allows users to examine objects in a similar way as he they would do in real life. Section 5 showed that the use of advanced volume visualization techniques makes it possible to interactively generate expressive illustrations based on real data rather than hand-made geometric models. And finally, in Section 6 we have seen that medical applications such as treatment planning can greatly benefit from illustrative visualization.

Illustrative visualization has generated a considerable amount of interest in the community. While we can learn a lot by studying the world of illustration, many new challenges arise when adapting traditional techniques to computer-based visualization. The aspect of interaction seems of particular importance in this context as traditional illustration does not feature any interaction capabilities.

References

1 T. A. DeFanti B. H. McCormick and M. D. Brown. Visualization in scientific computing. *Computer Graphics*, 21(6), 1987.

2 M. J. Bailey. The use of solid rapid prototyping in computer graphics and scientific visualization. In *ACM SIGGRAPH 1996 Course Note 37: The Use of Touch as an I/O Device for Graphics and Visualization*, 1996.

3 E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. DeRose. Toolglass and magic lenses: the see-through interface. In *Proceedings of ACM SIGGRAPH 1993*, pages 73–80, 1993.

4 G. Borgefors. Chamfering: A fast method for obtaining approximations of the euclidean distance in N dimensions. In *Procedings of the Scandinavian Conference on Image Analysis 1983*, pages 250–255, 1983.

5 M. Brierley. Arteriovenous malformations, 2000. `http://brainavm.oci.utoronto.ca/swf/intro.html`.

6 S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving volume rendering. In *Proceedings of EuroVis 2005*, pages 69–76, 2005.

7 S. Bruckner and M. E. Gröller. VolumeShop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization 2005*, pages 671–678, 2005.

8 M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Distortion viewing techniques for 3-dimensional data. In *Proceeding of the IEEE Symposium on Information Visualization 1996*, pages 46–53, 1996.

9 M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications*, 17(4):42–51, 1997.

10 K. L. Chelule, T. J. Coole, and D. G. Cheshire. Fabrication of medical models from scan data via rapid prototyping techniques. In *Proceedings of Time-Compression Technologies 2000*, pages 45–50, 2000.

11 M. Chen, C. Correa, S. Islam, M. W. Jones, P.-Y. Shen, D. Silver, S. J. Walton, and P. J. Willis. Deforming and animating discretely sampled object representations. In *Proceedings of Eurographics 2005 – State of the Art Reports*, pages 113–140, 2005.

12 M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In *Proceedings of the International Workshop on Volume Graphics 2003*, pages 35–44, 2003.

13 P. Cignoni, C. Montani, , and R. Scopigno. Magicsphere: an insight tool for 3d data visualization. *Computer Graphics Forum*, 13(3):317–328, 1994.

14 J. Claes, F. Di Fiore, G. Vansichem, and F. Van Reeth. Fast 3D cartoon rendering with improved quality by exploiting graphics hardware. In *Proceedings of Image and Vision Computing New Zealand 2001*, pages 13–18, 2001.

15 M. Cohen and K. Brodlie. Focus and context for volume visualization. In *Proceedings of Theory and Practice of Computer Graphics 2004*, pages 32–39, 2004.

16 J. Cordes, J. Dornheim, B. Preim, I. Hertel, and G. Strauß. Preoperative segmentation of neck CT datasets for the planning of neck dissections. In *Proceedings of SPIE Medical Imaging 2006*, pages 1447–1456, 2006.

17 F. M. Corl, M.R. Garland, and E. K. Fishman. Role of computer technology in medical illustration. *American Journal of Roentgenology*, 175(6):1519–1524, 2000.

18 N. Cornea, D.Silver, and P. Min. Curve-skeleton applications. In *Proceedings of IEEE Visualization 2005*, pages 95–102, 2005.

19 C. D. Correa and D. Silver. Dataset traversal with motion-controlled transfer functions. In *Proceedings of IEEE Visualization 2005*, pages 359–366, 2005.

20 B. Csébfalvi, L. Mroz, H. Hauser, A. König, and M. E. Gröller. Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum*, 20(3):452–460, 2001.

21 G. W. Furnas. Generalized fisheye views. In *Procedings of ACM SIGCHI 1986*, pages 16–23, 1986.

22 N. Gagvani and D. Silver. Parameter-controlled volume thinning. *Graphical Models and Image Processing*, 61(3):149–164, 1999.

23 N. Gagvani and D. Silver. Animating volumetric models. *Graphical Models and Image Processing*, 63(6):443–458, 2001.

24 A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of ACM SIGGRAPH 1998*, pages 447–452, 1998.

25 B. Gooch and A. Gooch. *Non-Photorealistic Rendering*. AK Peters, 2001.

26 H. Hauser, L. Mroz, G.-I. Bischi, and M. E. Gröller. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):242–252, 2001.

27 S. Islam, S. Dipankar, D. Silver, and M. Chen. Spatial and temporal splitting of scalar fields in volume graphics. In *Proceedings of the IEEE Symposium on Volume Visualization and Graphics 2004*, pages 87–94, 2004.

28 S. Islam, D. Silver, and M. Chen. Volume splitting and its applications. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):193–203, 2006.

29 T. Keahey and E. Robertson. Techniques for non-linear magnification transformations. In *Proceedings of the IEEE Symposium on Information Visualization 1996*, pages 38–45, 1996.

30 R. Kosara, S. Miksch, and H. Hauser. Focus+context taken literally. *IEEE Computer Graphics and Applications*, 22(1):22–29, 2002.

31 M. Kreuseler, N. Lopez, and H. Schumann. A scalable framework for information visualization. In *Proceedings of the IEEE Symposium on Information Vizualization 2000*, pages 27–38, 2000.

32 A. Krüger, C. Tietjen, J. Hintze, B. Preim, I. Hertel, and G. Strauß. Interactive visualization for neck dissection planning. In *Proceedings of EuroVis 2005*, pages 295–302, 2005.

33 E. LaMar, B. Hamann, and K. I. Joy. A magnification lens for interactive volume visualization. In *Proceedings of the Pacific Conference on Computer Graphics and Applications 2001*, pages 223–232, 2001.

34 Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

35 M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

36 M. Levoy and R. Whitaker. Gaze-directed volume rendering. *Computer Graphics*, 24(2):217–223, 1990.

37 A. Lu, C. J. Morris, D. S. Ebert, P. Rheingans, and C. Hansen. Non-photorealistic volume rendering using stippling techniques. In *Proceedings of IEEE Visualization 2002*, pages 211–218, 2002.

38 E. B. Lum and K.-L. Ma. Hardware-accelerated parallel non-photorealistic volume rendering. In *Proceedings of the International Symposium on Non-photorealistic Animation and Rendering 2002*, pages 67–74, 2002.

39 E. B. Lum and K.-L. Ma. Lighting transfer functions using gradient aligned sampling. In *Proceedings of IEEE Visualization 2004*, pages 289–296, 2004.

40 M. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Proceedings of the IEEE Visualization 2003*, pages 401–408, 2003.

41 C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. LifeLines: Visualizing personal histories. In *Proceedings of ACM SIGCHI 1996*, pages 221–227, 1996.

42 T. Porter and T. Duff. Compositing digital images. *Computer Graphics*, 18(3):253–259, 1984.

43 P. Rheingans and D. S. Ebert. Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):253–264, 2001.

44 G. G. Robertson and J. D. Mackinlay. The document lens. In *Proceedings of the ACM Symposium on User Interface Software and Technology 1993*, pages 101–108, 1993.

45 P. K. Robertson. A methodology for choosing data representations. *IEEE Transactions on Computer Graphics and Applications*, 11(3):56–68, 1991.

46 M. Sheelagh, T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Making distortions comprehensible. In *Proceedings of the IEEE Symposium on Visual Languages 1997*, pages 36–45, 1997.

47 P. Shirley. *Fundamentals of computer graphics*. AK Peters, 2005.

48 V. Singh, D. Silver, and N. Cornea. Real-time volume manipulation. In *Proceedings of the International Workshop on Volume Graphics 2003*, pages 45–51, 2003.

49 T. Strothotte and S. Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation*. Morgan Kaufmann, 2002.

50 N. Svakhine, D. S. Ebert, and D. Stredney. Illustration motifs for effective medical volume illustration. *IEEE Computer Graphics and Applications*, 25(3):31–39, 2005.

51 A. Tappenbeck, B. Preim, and V. Dicken. Distance-based transfer function design: Specification methods and applications. In *Proceedings of Simulation und Visualisierung 2006*, pages 259–274, 2006.

52 C. Tietjen, T. Isenberg, and B. Preim. Combining silhouettes, surface, and volume rendering for surgery education and planning. In *Proceedings of EuroVis 2005*, pages 303–310, 2005.

53 C. Tietjen, B. Meyer, S. Schlechtweg, B. Preim, I. Hertel, and G. Strauß. Enhancing slice-based visualizations of medical volume data. In *Proceedings of EuroVis 2006*, pages 123–130, 2006.

54 I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven volume rendering. In *Proceedings of IEEE Visualization 2004*, pages 139–145, 2004.

55 I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):408–418, 2005.

56 L. Wang, Y. Zhao, K. Mueller, and A. E. Kaufman. Generating sub-resolution detail in images and volumes using constrained texture synthesis. In *Proceedings of IEEE Visualization 2004*, pages 75–82, 2004.

57 L. Wang, Y. Zhao, K. Mueller, and A. E. Kaufman. The magic volume lens: An interactive focus+context technique for volume rendering. In *Proceedings of IEEE Visualization 2005*, pages 367–374, 2005.

58 X. Wei, A. E. Kaufman, and T. J. Hallman. Case study: visualization of particle track data. In *Proceedings of IEEE Visualization 2001*, pages 465–468, 2001.

59 J. Zhou, A. Döring, and K. D. Tönnies. Distance based enhancement for focal region based volume rendering. In *Proceedings of Bildverarbeitung für die Medizin 2004*, pages 199–203, 2004.

60 J. Zhou, M. Hinz, and K. D. Tönnies. Focal region-guided feature-based volume rendering. In *Proceedings of the International Symposium on 3D Data Processing, Visualization, and Transmission 2002*, pages 87–90, 2002.

Model-Based Visualization for Intervention Planning

Bernhard Preim[1]

1 **Otto-von-Guericke-University of Magdeburg, Dept. for Computer Science, Visualization Group, PO Box 4120, 39016 Magdeburg, Germany**
preim@isg.cs.uni-magdeburg.de

Abstract

Computer support for intervention planning is often a two-stage process: In a first stage, the relevant segmentation target structures are identified and delineated. In a second stage, image analysis results are employed for the actual planning process. In the first stage, model-based segmentation techniques are often used to reduce the interaction effort and increase the reproducibility. There is a similar argument to employ model-based techniques for the visualization as well. With increasingly more visualization options, users have many parameters to adjust in order to generate expressive visualizations. Surface models may be smoothed with a variety of techniques and parameters. Surface visualization and illustrative rendering techniques are controlled by a large set of additional parameters. Although interactive 3d visualizations should be flexible and support individual planning tasks, appropriate selection of visualization techniques and presets for their parameters is needed. In this chapter, we discuss this kind of visualization support. We refer to *model-based visualization* to denote the selection and parameterization of visualization techniques based on 'a priori knowledge concerning visual perception, shapes of anatomical objects and intervention planning tasks.

1998 ACM Subject Classification I.3.7 Three-Dimensional Graphics and Realism, J.3 Life and Medical Sciences

Keywords and phrases Model-based Visualization, Surface Visualization, Illustrative Rendering

Digital Object Identifier 10.4230/DFU.SciViz.2010.163

1 Introduction

Surgical interventions, radiotherapies and other local therapies require a precise understanding of the patient's anatomy. In particular, the location and extent of pathologic variations in relation to vital anatomic structures, such as major blood vessels, is essential to evaluate the resectability and to determine the surgical strategy. Interventions are planned by means of CT or MRI data. Planning involves a systematic exploration of the slices of radiological data. In order to support the mental preparation of surgeons, more and more 3d visualizations are generated. Oblique MPR (multiplanar reformation) slices for instance allow to assess the local cross section of vascular structures and volume rendering is employed to get an overview which is essential for example in case of complex fractures or rare anatomic variants.

Intervention planning can be supported even better if image analysis results, such as segmentation information concerning the relevant objects, are available. For an efficient segmentation, model-based segmentation approaches are often exploited. Statistical models, such as Active Shape Models and Active Appearance Models, employ 'a priori knowledge with respect to the expected shape and grey value distributions [8, 9]. With active contour models-another class of model-based segmentation techniques-deformable models are fitted

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 163–178

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany

to the segmentation target structure based on a flexible geometric representation such as B-Splines. The process of fitting the model to the target structure is guided by physical principles and constraints which restrict for example the curvature along the path (model assumptions) [24, 27, 39].

Based on image analysis results, visualization parameters can be locally adapted to individual objects or certain categories of anatomic structures, such as nerves or lymph nodes. Since visualizations should provide insights into spatial relations, there is an argument for visualization techniques which "idealize" anatomic structures to some extent to render them more comprehensibly.

The design of "idealized" visualizations requires assumptions or 'a priori knowledge with respect to geometric properties. This gives rise to the term *model-based visualization*. More general, model-based visualization refers to the automatic selection of appropriate visualization techniques. There is a variety of sources which can be exploited to derive such automatic selections. Similar to the model generation process in image segmentation, experience with the visualization of a variety of similar datasets is an essential source of information. In case of clinical applications, "idealized" visualizations must be shown to be "correct enough" to draw reliable conclusions. Therefore we discuss the validation of model-based visualization techniques.

Model-based visualization versus intelligent computer graphics. The (semi-)automatic selection and parameterization of visualization techniques – which we characterized as model-based visualization – might be considered as an instance of *knowledge-based* or *intelligent* computer graphics. However, the typical goals of knowledge-based computer graphics are considerably more ambitious: the automatic selection of appropriate viewpoints and perspectives, the computation of complex layouts, labeling of 3d models, the selection of appropriate levels of detail and the determination of movements of a virtual camera through complex virtual environments are among these goals [21, 12, 6, 16, 22, 36]. The general concept of knowledge-based computer graphics is to hierarchically decompose high-level intent-based specifications into more and more elementary specifications until they are precise enough to be rendered. The results are evaluated with respect to rules and constraints and backtracking mechanisms are employed to initiate new solutions if the initial solutions failed to generate an appropriate result. The goals of model-based visualization are at a lower and more elementary level. Since neither knowledge representations nor backtracking mechanisms are involved, model-based visualization should not be regarded as *intelligent graphics*.

Organization. A general problem for many intervention planning tasks is the generation of geometric models which represent the segmentation results. Due to the large variety of steps and algorithms, a model-based approach is needed for this problem. As a first step, we discuss the appropriateness of mesh smoothing algorithms for different categories of anatomic structures (Section 2). While flat and compact structures can be smoothed satisfactory with general methods, elongated and in particular branching structures require dedicated smoothing approaches. In Section 3, we therefore discuss model-based visualization of vascular structures. In Section 4, we describe the process of generating geometric models for illustrative visualization with a focus on silhouettes and feature lines. Illustration techniques, such as cut-away and ghostviews, and their application are discussed in Section 4. Finally, we provide a general discussion of 'a priori knowledge for visualization purposes in Section 5. In essence, this chapter should rather present a framework for the analysis and refinement of visualization techniques instead of presenting final results.

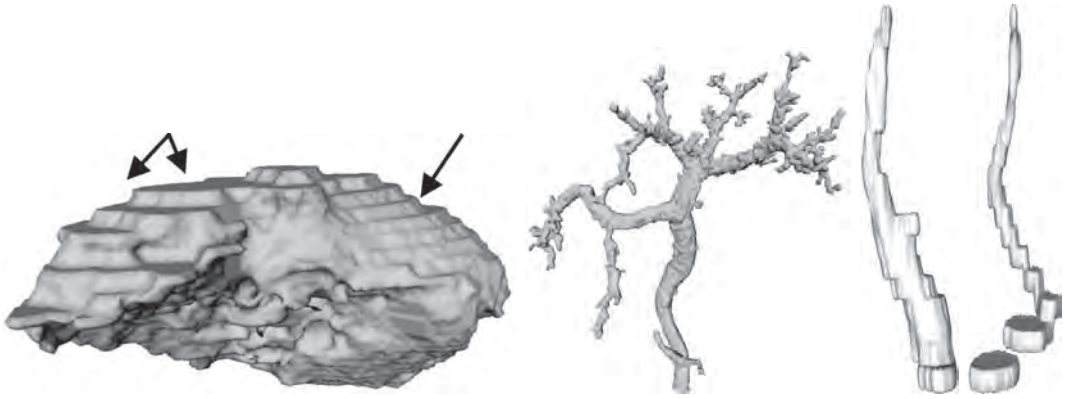


Figure 1 Typical problems of applying Marching Cubes to binary segmented volume data. From left to right: an organ (large compact), a vascular tree (elongated branching), and two individual vascular structures. The arrows in the left image relate to typical problems of visualizing compact objects from data with insufficient slice distance: plateaus and visible staircases arise. Note, that due the oblique course of the vascular structure on the very right, the segmentation results in adjacent slices do not overlap resulting in the generation of several surfaces. Images are courtesy of Jens Haase, University of Magdeburg.

2 Towards Model-Based Surface Extraction and Smoothing

For many rendering options, it is essential to transform segmentation results into (polygonal) surfaces. The usual representation of a segmentation is a binary volume with the same resolution as the original volume data (the set of "1" voxels represents a particular anatomic structure). The common surface extraction technique is the Marching Cubes-algorithm (or one of its refinements which handle ambiguities in a more sophisticated manner). The problem with this general strategy is that Marching Cubes leads to jaggy surfaces if it is applied to binary volume data, in particular if the slice distance is high (e.g. ≥ 5 mm). In some cases, for example when the segmentation results in adjacent slices do not overlap, Marching Cubes would not even generate a connected polygonal surface (Figure 1, right).

A variety of techniques have been developed and discussed to improve the surfaces either before, during or after surface extraction. Before surface extraction, image processing filters may be applied to convert the binary volume in a multivalue volume. In particular, with morphologic filters a good trade-off between accuracy and smoothness can be achieved [28]. Another technique which is applied before surface extraction is the interpolation of intermediate slices. The surface extraction itself may be improved by gradually refining the initial Marching Cubes result when it is strongly discontinuous [7]. Most research however tackles the question how an existing polygonal surface may be smoothed [38].

Smoothing geometric models is a wide topic, similar to smoothing image data. Simple methods tend to remove not only noise but also relevant features. Advanced methods, such as those based on diffusion theory better retain relevant features. The improved quality is attained at the expense of long computation times. However, no single smoothing method is appropriate for all anatomic and pathologic structures. Pathologic structures, for example, should not shrink in the smoothing process, whereas this requirement is less crucial for large organs. Again, the suitable selection, combination and parameterization of smoothing techniques requires 'a priori knowledge with respect to the shapes to which they are applied. Smoothing techniques also alter the geometry and therefore, must be evaluated by measuring distances to "correct" visualizations. The appropriateness of these techniques depends on a

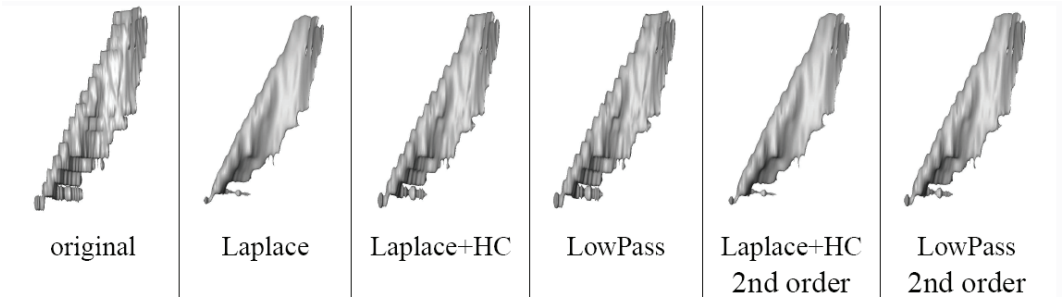


Figure 2 Smoothing results of an elongated surface model (a muscle in the neck region). Images are courtesy of Jens Haase, University of Magdeburg.

large number of parameters:

- *image acquisition parameters*, such as the slice distance.
- *category of anatomic object*. Anatomic objects have strongly different shapes and sizes: elongated, branching, planar, and compact objects occur. Smoothing techniques which are appropriate for one category may lead to inacceptable results for another.
- *task-specific requirements*. Objects which serve as anatomic context should be smoothed strongly even at the expense of accuracy. Objects relevant for the surgical strategy must be more carefully processed since accuracy is more important. For some structures, such as a malignant tumor, accuracy has the utmost priority.

As a first step, towards a model-based solution, we explored the effects of wide-spread and fast smoothing approaches [2]. We restricted the comparison to smoothing algorithms which have a linear complexity (in each iteration, each vertex is replaced by a weighted sum of its previous position and the positions of adjacent vertices). The Laplace filter, the Laplace-filter with the so-called HC correction [42], and the LowPass filter [38] were considered(Figure 2). Median and average filter, applied to surface normals [44], were also explored but initial results were not encouraging in particular for small objects where new artifacts were created in some cases. The major strength of the latter filters is the preservance of sharp edges as they occur in CAD-models.

Visual quality as well as different metrics (volume preservation, Hausdorff distance to the original model, and the average curvature) were employed to compare different methods. An essential aspect of each filter is the neighborhood which is considered at each point. All filters were implemented with the topological-distance of 1 (only vertices which share an edge with the current vertex) and topological distance of 2 (vertices which are connected to the current vertex by a path of at most two edges are considered). Figure 3 compares smoothing with both neighborhoods.

Each filter was applied to a set of six anatomic objects each representing a different class or category of objects. Each of the filters has two parameters influencing the accuracy and visual quality: the number of iterations and the smoothness factor. The investigation considered 6 weighting factors (from 0.05 to 0.9) and 4 different numbers of iterations (from 5 to 50). The website `http://wwwisg.cs.uni-magdeburg.de/cv/projects/LST/smoothing/` presents all results.

The *LowPass* filter turned out to be the most appropriate fundamental smoothing filter for all reference objects. To smooth compact objects (e.g. organs, lymph nodes), the *LowPass* filter with a 2nd order neighborhood, a weighting factor of about 0.7, and 20 to 50 iterations should be used. A similar smoothing strategy can be applied to planar objects

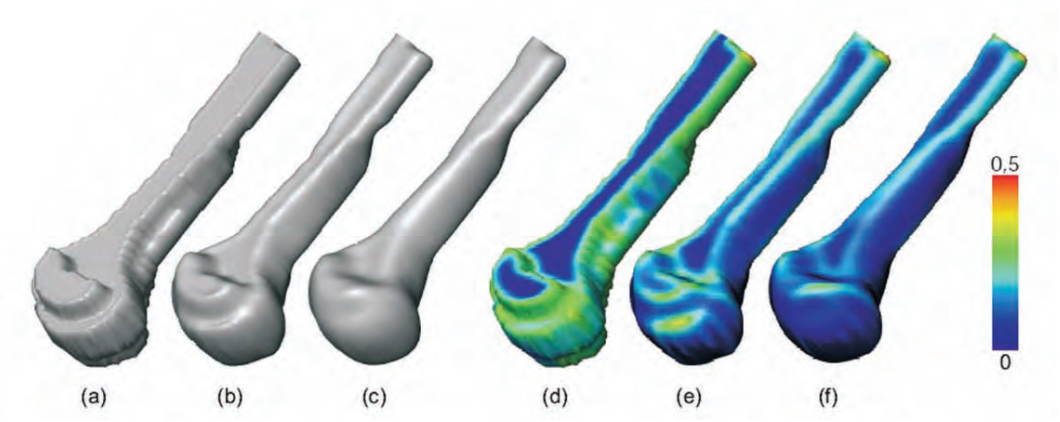


Figure 3 A bone (left) is smoothed with the LowPass-filter with the normal (b) and extended neighborhood (c). The curvature plots show the effect of smoothing on the mean curvature. LowPass-filtering with the extended neighborhood strongly reduces the curvature (f). Images are courtesy of Jens Haase, University of Magdeburg.

(e.g. ligaments), whereas here not more than 20 iterations should be applied. With the recommendations above, the volume of the smoothed models is preserved well: it is exactly preserved for large compact models and for smaller or elongated objects shrinkage was lower than 4% and the Hausdorff distance which is a worst case approximation of the distance error was between 3 to 6 mm. This amount of distance error is reasonable compared to the resolution of the underlying data.

Flat objects (thin objects which might be curved, such as ligaments) with holes and frayed parts should be smoothed with a 1st order neighborhood. Holes cannot be closed by any smoothing algorithm; but at least they should not be enlarged. Elongated objects with many small branches and detached object parts (recall Fig. 1, right) cannot be smoothed appropriately with any of the general smoothing filters. For smoothing simple none branched elongated objects, the *LowPass* filter with a weighting factor of 0.5 and 10 iterations is recommended. The visual results achieved with the Laplace-filter with correction are similar to the *LowPass* filter. However, for larger numbers of iterations and/or larger smoothing factors volume shrinkage (8 to 12%) and Hausdorff distances are larger. The accuracy of the Laplace-filter with HC-correction is considerably better with a 2nd order neighborhood.

2.1 Validation

Similar to new segmentation methods, model-based visualization techniques should be carefully validated with respect to accuracy. This includes qualitative and quantitative comparisons with other methods. Quantitative comparisons are based on metrics which characterize distances between segmentation or visualization results or based on volume overlaps [45]. In particular, the comparison with a "gold standard" is essential. The "gold standard" represents the solution which is regarded as "true" or at least as the most accurate result which could be generated so far. For image segmentation, the manual segmentation of medical experts is usually considered as gold standard.

For model-based visualization, a validation is required to investigate whether the segmentation result is correctly displayed. For our purposes, we considered isosurface rendering with the Marching Cubes method [25] as gold standard (taking 0.5 as isolvalue, when "1" represents foreground voxels and "0" represents background voxels). As has been discussed

by [7], Marching Cubes is not the most accurate visualization. However, it is close to it and a more precise visualization (trilinear interpolation) is more complex to implement and considerably slower.

With respect to the smoothing techniques, we evaluated primarily the volume preservation. It is known and not surprising that Laplacian smoothing leads to a strong loss of volume (with larger smoothness factors). The relation between Laplace with correction and the *LowPass* filter as well as the precise influence of the neighborhood on accuracy were not known before.

2.2 Discussion

We derived initial recommendations for smoothing surfaces based on the analysis of widespread and fast iterative methods. It is necessary to include more algorithms, in particular those which adapt the surface normal's [37] and diffusion based methods [11, 37]. So far, our results are limited by the fact that only one segmentation result for each category of anatomic structure is considered. We derived results in a systematic but purely empirical manner. An alternative and more elegant approach would be to derive hypothesis on the suitability of smoothing methods based on an analysis of the properties of the algorithms. We have not investigated so-called normal errors – the angle between the correct surface orientation and the surface computed with a new method. Pommert et al. [32] discussed the validation of medical visualization techniques with respect to distance and normal errors and employed a set of phantoms to study different aspects, such as the influence of the sampling density (see also [31]).

3 Model-based Visualization of Vascular Structures

For intervention planning, it is crucial that spatial relations can be correctly inferred from the visualization. In particular the topology of vascular trees is often essential to decide on the feasibility of a surgical strategy. Moreover, the curvature, the depth relations, and the diminution of the diameter towards the periphery should be depicted correctly. With conventional visualizations, such as Maximum Intensity Projection (MIP) or surface rendering, artifacts arise due to the limited resolution and inhomogeneities of contrast enhancement. Therefore, vascular structures should be reconstructed based on the radiological data of a patient and some model assumptions as to the shape of vasculature [3, 15]. The pioneering work of Barillot et al. [3] is probably the first dedicated effort to generate medical visualizations based on 'a priori knowledge. Healthy vascular structures exhibit a roughly circular cross-section, they are connected with each other and their diameter shrinks from the root to the periphery. Based on these assumptions, a variety of visualization techniques have been developed which use the skeleton and the local vessel diameter as input. Primarily graphics primitives, such as cylinders and truncated cones, were fitted to the skeleton and scaled according to the local vessel diameter [26, 18]. The most advanced explicit reconstruction technique is based on subdivision surfaces [13, 5]. More specialized model assumptions were employed by Puig in [34]. She considered typical elements (cylindrical, stenosis, ...) and branching structures in cerebral vasculature, tried to classify branchings accordingly and used 'a priori knowledge to emphasize the corresponding branching type.

The explicit construction of a geometry however exhibits problems in particular at branchings were discontinuities arise at the joint of truncated cones or cylinders. A superior image quality can be achieved by means of implicit surfaces, where the shape of a vascular system is described by an implicit equation which has to be evaluated along the skeleton.

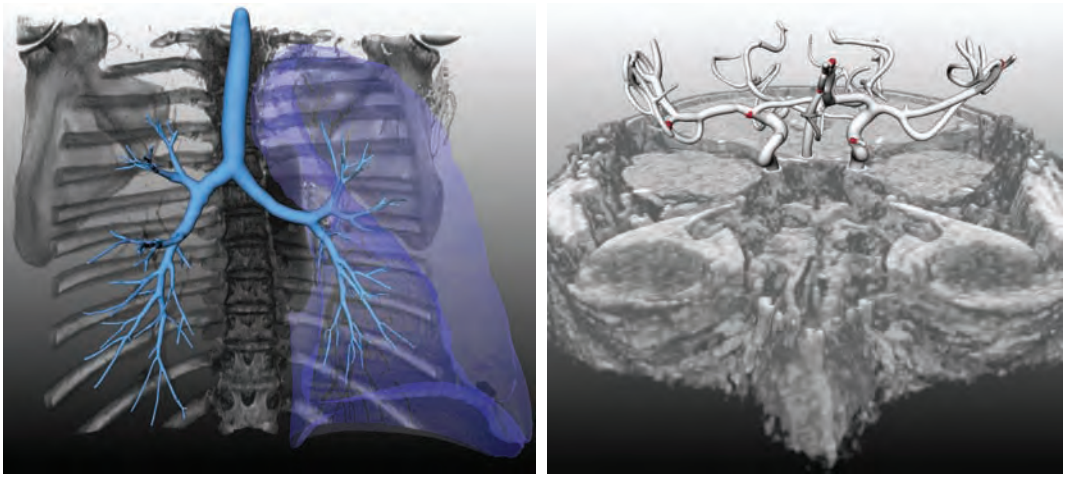


Figure 4 Model-based visualization of vascular structures embedded in direct volume renderings of surrounding structures. Left: The bronchial tree is depicted. Right: a cerebral tree is shown. Images are courtesy of Steffen Oeltze, University of Magdeburg.

The resulting scalar fields are polygonized by means of a threshold. A special variant of implicit surfaces, convolution surfaces, allow to visualize branching skeletal structures by applying a convolution filter to the skeletons. The use of convolution surfaces for medical visualization poses some problems with respect to accuracy; the depicted vascular structures should correctly convey the vessel diameter and the topology of vascular structures. Usually, convolution surfaces exhibit "unwanted effects", such as unwanted blending where two branches are incorrectly merged with each other due to the construction process. Oeltze et al. could show that an appropriate filter selection allows to effectively avoid that the resulting visualizations strongly deviate from the segmentation results on which they are based [29]. Two examples of this work are shown in Figure 4.

3.1 Validation

In order to validate the convolution surface as viable technique for vascular structures, various experiments with small artificial data have been accomplished to study whether unwanted blending or bulging occurs. Another qualitative part of the validation was to analyze visualization results achieved with "real" patient data and compare them with the results achieved with other model-based techniques. After these tests the width of the convolution filter was adapted and a quantitative validation was based on 10 abdominal CT datasets (patients with liver metastases) with different resolution and distances were computed for each vertex of the resulting polygonal mesh.

Distance metrics, such as mean distance and Hausdorff distance, are primarily relevant for assessing the accuracy of vessel visualization techniques. As the major result of a quantitative validation, Oeltze et al. found that the deviation of "their" variant of convolution surfaces to an isosurface rendering of the segmentation result is on average below half the diagonal size of a voxel. Taking into account that half the diagonal size of a voxel is the uncertainty which is due to resolution of the data, this is an excellent result. Only for a very small fraction of the voxels the distance is up to 3 diagonal voxel sizes [30]. Figure 5 illustrates how the results were achieved and analyzed.

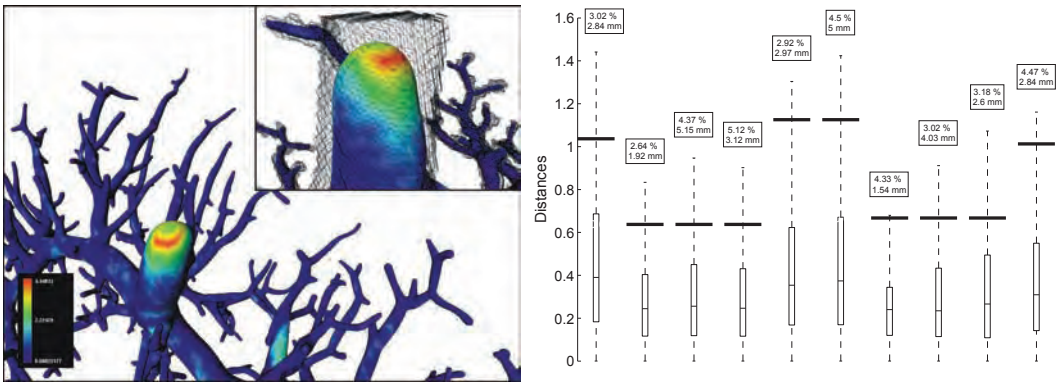


Figure 5 Validation of a model-based visualization techniques. Left: Intensity-coded visualization of the deviation from convolution surface (CS) to isosurface. The strongest deviations occur at the root of the vessel tree (see the inset with the superimposed isosurface in wire-frame mode). Right: Boxplots of the distance measures (in mm) carried out for a comparison of CS and Isosurface based on 10 vascular trees. Each box indicates the lower quartile, median, and upper quartile values. The whiskers extend from each end of the box to show the extent of the rest of the data. The values within each box represent the percentage of data values beyond the ends of the whiskers and the maximum distance. Thick lines indicate the half diagonal voxel size. Images are courtesy of Steffen Oeltze, University of Magdeburg.

3.2 Discussion

Model-based visualization refers to the automatic selection of appropriate visualization techniques. With respect to the visualization of vascular structures this involves an assessment of the local vessel diameter in cross sectional areas.

If it turns out that the assumption of a circular cross section is strongly violated in several adjacent slices, any visualization technique which assumes this property is obviously not suitable. In such cases, a pathology is likely and an isosurface of the segmentation result is a better visualization option. Pathologies such as stenosis or aneurysms occur at small portions of a vascular system. A hybrid combination of isosurface rendering (in pathologic portions) and model-based rendering (in healthy portions) is probably the best choice to depict pathologic vascular systems.

There are some similarities between model-based vessel segmentation and visualization. Model-based vessel segmentation techniques also assume connectedness of vascular structures and try to "bridge" over a few voxels which fail to fulfill a homogeneity criterion due to partial volume effects. An ellipsoidal cross-section is often assumed in vessel segmentation approaches [19]. In general, model assumptions in image segmentation must be less restrictive to cope with the variety of shapes and the imperfect quality of medical image data.

4 Model-based Illustrative Rendering

Conventional 3d visualization includes volume rendering and surface rendering where color and transparency are employed to selectively emphasize anatomic structures. These techniques have obvious limitations if a variety of different objects is relevant for a treatment decision and need to be displayed simultaneously. These limitations recently led to the development of illustrative rendering techniques [4, 17, 40], which can be flexibly combined with conventional medical visualization techniques. These new techniques involve an increased flexibility on the one hand and an increased necessity to adjust parameters on the other hand. In clinical

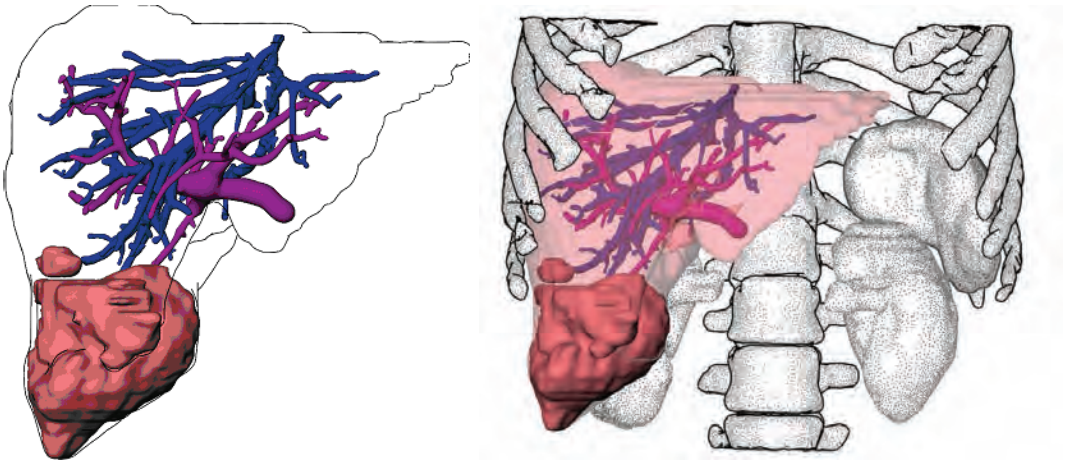


Figure 6 Liver, intrahepatic vasculature as well as a tumor are depicted for intervention planning. Convolution surfaces are used for the vasculature. While the left image is restricted to the liver and their internal structures, the right image also contains surrounding structures as context objects. Stippling is a useful technique for these structures. Left image is courtesy of Christian Tietjen, right image is courtesy of Alexandra Baer (both from University of Magdeburg).

applications, presets are necessary to reduce the interaction effort. These presets must consider which techniques and which parameters of these techniques are appropriate for certain categories of anatomic structures. We regard this as another example of model-based visualization.

Illustrative rendering refers to the use of lines and points as rendering primitives. Silhouette rendering, hatching and stippling are used to render anatomic shapes more comprehensibly. After the pioneering work of Saito and Takahashi [35] a few dedicated therapy planning solutions have been developed, for example radiation treatment planning [20]. Illustrative rendering techniques are based on proven assumptions with respect to shape perception. Object boundaries are recognized faster and more precisely by depicting their silhouettes. Surface orientation is perceived more accurately (compared to shaded surfaces) if hatching lines along the main curvature directions are included [20, 43].

The potential of such visualization techniques for intervention planning can be easily shown. However, for practical use, the selection and parameterization of illustrative rendering techniques must be supported. Our experiments and an informal user study reveal that silhouette rendering is useful for large structures, such as organs (see Figure 6), but not for small structures such as small nodules [40]. Depending on the visualization goal, silhouette rendering may be used as the only rendering mode or combined with surface rendering. The exclusive use of silhouettes clearly indicates that the visualization of an object serves as anatomic context (only). More experiments (more datasets, different visualization goals, ...) and user studies are needed to derive more reliable conclusions.

It is also necessary to investigate the prerequisites for illustrative rendering. A major problem with the automatic use of silhouette and hatching line generation is the smoothness of surfaces. Silhouettes emphasize not only the relevant features of a boundary but also noisy portions which might occur due to smaller segmentation errors or large slice distances. Hatching lines are usually generated by considering curvature directions. Noisy surfaces exhibit frequent strong changes of surface normals and curvatures. Therefore, the hatching directions suddenly change and lead to distracting visualizations. In summary, object shapes

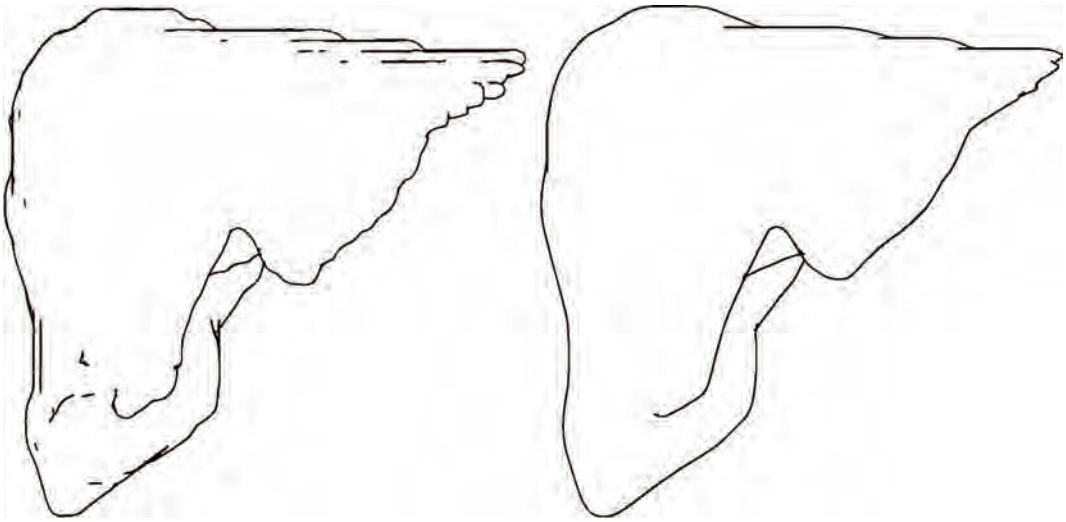


Figure 7 Left: Silhouette generation based on a typical segmentation result of the liver in abdominal CT data. Right: The polygonal model was strongly smoothed (relaxation filter with 7 iterations and relaxation factor 1.0) prior to silhouette generation. Images are courtesy of Christian Tietjen, University of Magdeburg.

resulting from a segmentation process usually require a subsequent smoothing step to be adequate for illustrative rendering (recall Section 2). Figure 7, illustrates how silhouette generation is affected by smoothing.

Illustrative techniques enrich the expressiveness of medical 3d visualizations by emphasizing silhouettes or characteristic features, such as ridges and valleys. This development is not finished yet; new hatching and stippling techniques (see Fig. 6, right) are devised which convey geometric properties such as curvature.

These new techniques exhibit considerably more parameters than silhouette rendering. The analysis of anatomic structures and the evaluation of sample image should lead to recommendations how to apply such techniques for certain anatomic structures.

5 Exploration of Nodules and Lymph Nodes with Cutaways and Ghosting

Emphasis techniques are useful to support the perception of relevant anatomic or pathologic structures. What is *relevant* is determined by the user, by either selecting an object name in a list or by picking its visual representation. A wide variety of emphasis techniques exists [33]. The selection should again consider geometric properties and 'a priori knowledge of the objects. As an example, we discuss emphasis techniques which were developed to support the exploration of lymph nodes, lung nodules and other pathologic structures.

The occurrence and localization of enlarged and potentially malignant lymph nodes is an essential information for planning surgical interventions, for example, in the neck region [23]. By contrast to vascular structures, lymph nodes, tumors and lung nodules do not exhibit a complex topology. Instead they are rather small and compact. They are explored together with adjacent structures in order to evaluate whether these structures are infiltrated. Without other structures displayed, small nodules cannot be localized. Often, it is a severe problem to display these structures simultaneously with sufficient opacity.

Cutaways – originating from technical illustrations – might be applied. Cutaway views

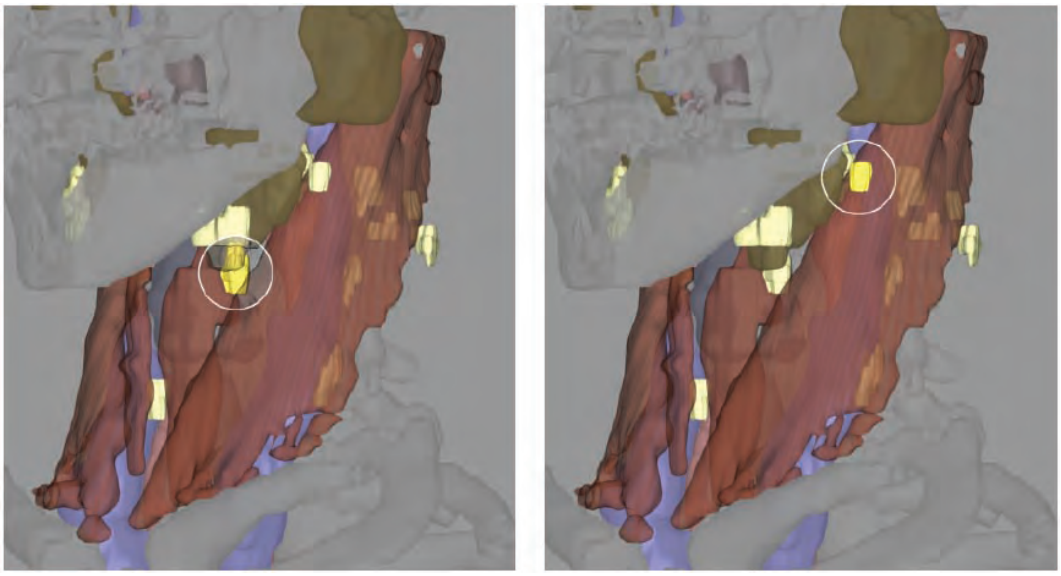


Figure 8 Lymphnodes in the neck region are emphasized by means of cylindrical ghostviews. A sequential exploration of all lymph nodes is supported taking into account the lymph node's size and position. Image is courtesy of Arno Krueger, University of Magdeburg.

are generated by removing a geometric shape to expose hidden objects. Instead, cutaways are applicable to show compact small objects. Compactness and relative size can be geometrically analyzed.

As a variation of cutaway views the cut region can be displayed transparently instead of a complete removal. This technique is referred to as ghostview (Feiner and Seligman [12]). An essential decision in the use of cut-away views and ghostings is the selection of a cut geometry. It should be regular to be recognizable as an illustration technique (anatomical shapes are not regular). The shape should "fit" to the objects which should become visible (see also [41] for a discussion of cutaways and related smart visibility techniques). Since lymph nodes, nodules, and metastasis have roughly circular shapes (model assumption), cylinders are appropriate cut-regions. Figure 8 shows cylindrical ghostviews used for neck dissection planning. Intervention planning, however, requires a systematic exploration of all enlarged lymphnodes. Based on this task knowledge, an exploration technique is needed which supports a sequential emphasis of all relevant lymphnodes. In [23], we suggested to use the Tab-key in order to emphasize all lymphnodes based on a sequence which considers size and local coherency.

6 Discussion

The previous sections presented a variety of examples where visualization techniques have been fine-tuned to particular target structures such as nodules and vascular structures. "Model-based" techniques are also needed for a variety of other applications, such as the visualization of diffusion tensor data, where 'a priori knowledge on white matter tracts and their branchings is incorporated in the visualization and clustering of fiber tracts [14, 46].

Similar to segmentation problems, the suitability of visualization techniques depends on the object shape, size and on the occurrence of other objects in the neighborhood. In many intervention planning applications, image analysis is regarded as a challenge and visualization

as a simple matter of using some wide-spread commercial rendering system. This is an over-simplified view of the difficult problem of conveying the essential information to the user. Visualization, on the other hand, can benefit from the substantial work on representing 'a priori knowledge for image segmentation. Smoothness constraints as they are used for Active Contours are relevant for silhouette rendering: If segmentation results fail to meet smoothness constraints, they cannot be directly employed for silhouette generation.

Model-based image segmentation recently started to represent not only one "target" structure, but also spatial relations of adjacent structures, see e.g., the Active Structural Shape Models developed by [1]. Similarly, the effectiveness of visualization techniques applied to one anatomic structure depends on the visualization techniques applied to other anatomic structures which are displayed simultaneously. Therefore, it is essential that intervention planning tasks are carefully studied in order to determine which collections of objects are explored simultaneously. These collections should be provided as predefined selections and the default visualization techniques should be chosen in such a way that the whole collection is comprehensibly displayed. As a simple example, colors and transparencies of such objects should be selected such that contrasts are easily perceived and all relevant objects are sufficiently visible.

Comparison of model-based segmentation and visualization. In Table 1, we compare information used for model-based segmentation and visualization. While the distribution of grey values of the target structures in CT and MRI data is valuable information for model-based segmentation, this information is not relevant for model-based visualization. Derived information such as gradient magnitude or curvature metrics is essential for edge-based segmentation, such as Live Wire. For visualization, these metrics can be regarded as indicators for the certainty of the visualization. Primary tumors for example, often have weak borders and their precise extent is uncertain. This information can be employed to select a visualization technique which conveys this uncertainty (for example a semitransparent volume rendering instead of a "perfect" shiny isosurface). We regard as geometric shape any shape descriptor; such as compactness or anisotropy. Assumptions related to shape descriptors are useful to identify the target structure and to visualize it appropriately. Similar, topology information, such as connectedness and the number of holes is essential for segmentation and visualization. The use of structural information for image analysis was clearly demonstrated. For visualization, it can be used for the design of color mapping schemes which employ information on adjacency of structures. Finally, visualization strongly benefits from research in visual perception. Whether something can be perceived at all, whether color differences can be discriminated, whether objects can be discriminated at a glance ("preattentive" vision) is dependent on the selection of visualization parameters. A variety of user studies have been carried out and provide a valuable source for 'a priori knowledge (recall Colin Ware's book [43]). Finally, task knowledge can be exploited to derive which objects are essential for certain tasks and to guide the selection of visualization parameters.

Despite the similarities between model-based segmentation and visualization there are also fundamental differences. Model-based segmentation is employed to automatically segment one target structure (with rather fixed topology). Model-based visualization is more general and refers to classes of anatomic structures, such as vascular systems or lymph nodes. Moreover, model-based segmentation is adapted to particular image data, such as T2-weighted MRI data, whereas model-based visualization does not consider the modality of the imaging device.

While there is one correct segmentation, there are potentially many appropriate visualization settings for a particular set of anatomic structures. The suitability of visualization

Table 1 Model-based segmentation and visualization

Information	Model-based segmentation	Model-based visualization
Grey value distribution	x	-
Gradient magnitude/ curvature metrics	x	x
Geometric shape	x	x
Topology	x	x
Structural relation between objects	x	x
Visual perception	-	x
Task knowledge	-	x

parameters depends on user preferences, previous experiences and on the visual capabilities of a particular user. The exploration of 3d models with appropriate interaction facilities is desirable and may lead to additional insights. In particular, rotation and zooming provide insight into spatial relations. However, the unrestricted exploration involves too many parameters. Therefore, a model-based approach is desired to start the exploration with a meaningful combination of visualization techniques.

7 Concluding Remarks

We introduced model-based visualization as a concept where the appropriateness and parameterization of visualization techniques is carefully adapted to the shape and size of the object, and to the context of its visualization in intervention planning. To realize model-based visualizations, the shape of the target structure has to be analyzed, for example, with respect to the branching pattern. The wide literature on shape description may be employed to select appropriate shape descriptors (see e.g. [10] for a recent book on shape classification).

We argue that model-based visualization is an essential goal in order to effectively exploit the huge space of visualization options. The fully interactive specification of all visualization parameters is not feasible since it is time-consuming and leads to visualization results which are neither optimal nor reproducible. Although we presented concepts and solutions for some specific problems, many aspects of model-based visualization require a more in-depth analysis. Even the visualization of vascular structures – probably the aspect which deserved most attention so far – requires the refinement of existing solutions, for example to better represent vascular structures at locations where the cross section is notably not circular.

There is an urgent need for further research in the adaptation of visualization techniques to intervention planning tasks. In particular, the appropriateness of visualization techniques must be assessed by the target users: medical doctors who prepare for complex interventions. Prospective user studies are required which compare visualization techniques with respect to their consequences for the surgical strategies. We restricted the discussion in this paper to static visualizations. The model-based generation of animation sequences for intervention planning is an interesting challenge for future research.

Acknowledgement. This work is based on research at the Visualization group in Magdeburg. Special thanks go to the members of the team Ragnar Bade, Alexandra Baer, Jeanette Cordes, Jana Dornheim, Arno Krueger, Konrad Muehler, Steffen Oeltze and Christian Tietjen as well as to our clinical partners, in particular to Dr. Ilka Hertel, Prof. Dr. Karl J. Oldhafer and Dr. Gero Strauss. The investigation of smoothing algorithms was accomplished by Jens Haase as part of his Master thesis.

References

1 S. Al-Zubi, A. Broemme, and K. Toennies. Using an Active Shape Structural Model for Biometric Sketch Recognition. In *Proceedings of DAGM*, pages 187–195, Magdeburg, Germany, 10.-12. September 2003.

2 Ragnar Bade, Jens Haase, and Bernhard Preim. Comparison of Fundamental Mesh Smoothing Algorithms for Medical Surface Models. In *Simulation und Visualisierung*, pages 289–304. SCS-Verlag, 2006.

3 C. Barillot, B. Gibaud, J. Scarabin, and J. Coatrieux. 3d reconstruction of cerebral blood vessels. *Computer Graphics and Applications*, 5(12):13–19, 1985.

4 S. Bruckner and E. Groeller. VolumeShop: An Interactive System for Direct Volume Illustration. In *Proc. of IEEE Visualization*, pages 671–7678, 2005.

5 K. Buehler, P. Felkel, and A.L. Cruz. *Geometric Modeling for Scientific Visualization*, chapter Geometric Methods for Vessel Visualization and Quantification - A Survey. Springer, 2004.

6 A. Butz. Betty: planning and generating animations for the visualization of movements and spatial relations. In *Proc. of the workshop on Advanced visual interfaces (AVI)*, pages 53–58, New York, USA, 1994. ACM Press.

7 P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computers & Graphics*, 24(3)(3):399–418, 2000.

8 T. Cootes, G. Edwards, and C. Taylor. Active appearance models. In *Proc. of European Conference on Computer Vision*, volume 2, pages 484–498, 1998.

9 T. Cootes, A. Hill, C. Taylor, and J. Haslam. The use of active shape models for locating structuresin medical images. *Image and Vision Computing*, 12(6):355–366, 1994.

10 L.F. Costa and R.M.C. Jr. *Shape Analysis and Classification: Theory and Practice*. CRC Press, Boca Raton, USA, 2001.

11 M. Desbrun, M. Meyer, P. Schroeder, and A.H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 317–324, New York, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

12 S. Feiner and D. Seligmann. Cutaways and ghosting: satisfying visibility constraints in dynamic 3d illustrations. *The Visual Computing*, 8(5 and 6):292–302, 1992.

13 P. Felkel, R. Wegenkittl, and K. Buehler. Surface Models of Tube Trees. *Computer Graphics International*, pages 70–77, 2004.

14 G. Gerig, S. Gouttard, and I. Corouge. Analysis of brain white matter via fiber tract modelling. In *IEEE Engineering in Medicine and Biology*, 2004.

15 G. Gerig, T. Koller, G. Szkely, C. Brechbuehler, and O. Kuebler. Symbolic description of 3d structures applied to cerebral vessel tree obtained from mr angiography volume data. In *Proc. of Information Processing in Medical Image*, Volume 687 of LNCS, pages 94–111, 1993.

16 T. Goetzelmann, K. Hartmann, and T. Strothotte. Contextual Grouping of Labels. In *Simulation and Visualization*, pages 245–258, 2006.

17 M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proc. of IEEE Visualization*, pages 301–308, 2003.

18 Horst Hahn, Bernhard Preim, Dirk Selle, and Heinz-Otto Peitgen. Visualization and Interaction Techniques for the Exploration of Vascular Structures. In *IEEE Visualization*, pages 395–402, San Diego, Oktober 2001.

19 A. Hennemuth, T. Boskamp, D. Fritz, C. Kuehnel, S. Bock, D. Rinck, H.O. Peitgen, and M. Scheuering. One-click coronary tree segmentation in ct angiographic images. In *Proc. of Computer-Assisted Radiology and Surgery (CARS)*, 2005.

20 V. Interrante, H. Fuchs, and S. Pizer. Illustrating transparant surfaces with curvature-directed strokes. In *Proc. of IEEE Visualization*, pages 211–218, 1996.

21 P. Karp and S. Feiner. Issues in the automated generation of animated presentations. In *Proc. of Graphics Interface*, pages 39–48, Toronto, Ont., Canada, 1990. Canadian Information Processing Society.

22 A. Krueger. Automatic graphical abstraction in intent-based 3d-illustrations. In *Proc. of the workshop on Advanced visual interfaces (AVI)*, pages 47–56, New York, USA, 1998. ACM Press.

23 Arno Krueger, Christian Tietjen, Jana Hintze, Bernhard Preim, Ilka Hertel, and Gero Strauss. Interactive Visualization for Neck Dissection Planning. In *IEEE/Eurographics Symposium on Visualization (EuroVis)*, pages 295–302, 2005.

24 T. Lehmann, J. Bredno, and K. Spitzer. *Methods Inf Med 1*, chapter On the design of active contours for medical image segmentation: A scheme for classification and contruction, pages 89–98. 2003.

25 W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. of the 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 163–169, 1987.

26 Y. Masutani, K. Masamune, and T. Dohi. Region-growing-based freature extraction algorithm for tree-like objects. In *Proc. of Visualization in Biomedical Computing*, Volume 1131 of LNCS, pages 161–171, 1996.

27 T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1(2):91–108, 1996.

28 A. Neubauer, M.T. Forster, R. Wegenkittel, Lukas Mroz, and Katja Buehler. Efficient Display of Background Objects for Virtual Endoscopy using Flexible First-Hit Ray Casting. In *Proc. VisSym*, pages 301–310, 2004.

29 S. Oeltze and B. Preim. Visualization of vascular structures with convolution surfaces. *Joint IEEE/EG Symposium on Visualization*, pages 311–320, 2004.

30 Steffen Oeltze and Bernhard Preim. Visualization of Vascular Structures: Method, Validation and Evaluation. *IEEE Transactions on Medical Imaging*, 25(4)(4):540–549, April 2005.

31 A. Pommert and K.H. Hoehne. Evaluation of Image Quality in Medical Volume Visualization: The State of the Art. In *Proc. of Medical Image Computing and Computer-Assisted Intervention (MICCAI), Part II*, Volume 2489 of Lecture Notes in Computer Science, pages 598–605. Springer, 2002.

32 A. Pommert and K.H. Hoehne. Validation of medical volume visualization: A literature review. In *Proc. of Computer Assisted Radiology and Surgery (CARS)*, Volume 1256 of Excerpta Medica International Congress Series, pages 571–576, Amsterdam, 2003. Elsevier.

33 Bernhard Preim, Christian Tietjen, and Christina Doerge. NPR, Focussing and Emphasis in Medical Visualizations. In *Simulation und Visualisierung*, pages 139–152. SCS, 2005.

34 A. Puig, D. Tost, and I. Navazo. An interactive cerebral blood vessel exploration system. In *Proc. of IEEE Visualization*, pages 443–446, 1997.

35 T. Saito and T. Takahaski. Comprehensible rendering of 3-d shapes. In *Proc. of the 17th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 197–206, 1990.

36 D.D. Seligmann and S. Feiner. Automated generation of intent-based 3d illustrations. In *Proc. of the 18th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 123–132, New York, USA, 1991. ACM.

37 T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface smoothing via anisotropic diffusion of normals. In *Proc. of IEEE Visualization*, pages 125–132, Washington, USA, 2002. IEEE Computer Society.

38 G. Taubin. A Signal Processing Approach to Fair Surface Design. In *Proc. of the Conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 351–358, 1995.

39 D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. *Artifical Intelligence*, 36(1):91–123, 1988.

40 Christian Tietjen, Tobias Isenberg, and Bernhard Preim. Combining Silhouettes, Surface, and Volume Rendering for Surgery Education and Planning. In *IEEE/Eurographics Symposium on Visualization (EuroVis)*, pages 303–310, 2005.

41 I. Viola, A. Kanitsar, and E. Groeller. Importance-Driven Volume Rendering. In *Proc. of IEEE Visualization*, pages 139–145, 2004.

42 J. Vollmer, R. Mencel, and H. Mueller. Improved Laplacian Smoothing of Noisy Surface Meshes. In *Proc. EuroGraphics*, pages 131–138, 1999.

43 C. Ware. *Information Visualization.* Morgan Kaufmann, 2000.

44 H. Yagou, Y. Ohtake, and A. Belyaev. Mesh Smoothing via Mean and Median Filtering Applied to Face Normals. In *Geometric Modeling and Processing Theory and Applications (GMP)*, pages 124–131, 2002.

45 Y. Zhang. A survey of evaluation methods for image segmentation. *Pattern Recognition*, 29:1335–1346, 1996.

46 L. Zhukov and A. Barr. Oriented tensor reconstruction from diffusion tensor mri. In *Proc. of IEEE Visualization*, pages 387–393, 2002.

Pre-operative Planning and Intra-operative Guidance for Shoulder Replacement Surgery

Charl P. Botha[1,3], Peter R. Krekel[1,2], Edward R. Valstar[2], Paul W. de Bruin[3], P. M. Rozing[2], and Frits H. Post[1]

1 **Department of Mediamatics, Delft University of Technology, The Netherlands**
{c.p.botha,f.h.post}@tudelft.nl
2 **Department of Orthopaedics, Leiden University Medical Centre, The Netherlands**
{p.r.krekel,e.r.valstar}@lumc.nl
3 **Department of Radiology, Leiden University Medical Center, The Netherlands**
{p.w.de_bruin,p.m.rozing}@lumc.nl

Abstract

Shoulder joint replacement, or arthroplasty, is indicated in cases where arthritis or trauma has resulted in severe joint damage that in turn causes increased pain and decreased function. However, shoulder arthroplasty is less successful than hip and knee replacement, mostly due to the complexity of the shoulder joint and the resultant complexity of the replacement operation.

In this paper we present a complete visualization-oriented pre-operative planning and intra-operative guidance approach for shoulder joint replacement. Our system assists the surgeon by allowing a virtual arthroplasty procedure whilst giving feedback, primarily via patient- and procedure-specific joint range of motion (ROM) simulation and visualization. After a successful planning, our system automatically generates a 3D model of a patient-specific mechanical guidance device that is then produced by a rapid prototyping machine and can be used during the operation. In this way, a computer-based guidance system is not required in the operating room.

1998 ACM Subject Classification I.3.8 Applications, J.3 Life and Medical Sciences

Keywords and phrases Surgery Assistance

Digital Object Identifier 10.4230/DFU.SciViz.2010.179

1 Introduction

Rheumatoid arthritis and osteoarthritis, the two most common forms of arthritis, are diseases that adversely affect joint cartilage and bone quality. They can lead to severe joint damage that in turn causes increased pain and reduced joint mobility and patient function. In these cases, joint replacement is indicated. Joint replacement, or arthroplasty, is a surgical procedure during which diseased parts of a joint are removed and replaced with artificial components.

Hip and knee replacements are successful procedures with regard to pain relief, post-operative joint functionality and durability. At ten years follow-up, the revision rate for cemented total hip prostheses is 7% and about 13% for uncemented total hip prostheses [15]. In contrast to this, shoulder replacement yields good results with regard to pain relief, but fares significantly worse with regard to post-operative joint functionality and especially replacement durability. Literature shows that after nine years, between 24% and 44% of shoulder glenoid components show radiological loosening [22], i.e. loosening that is visible on a radiograph.

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 179–195

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

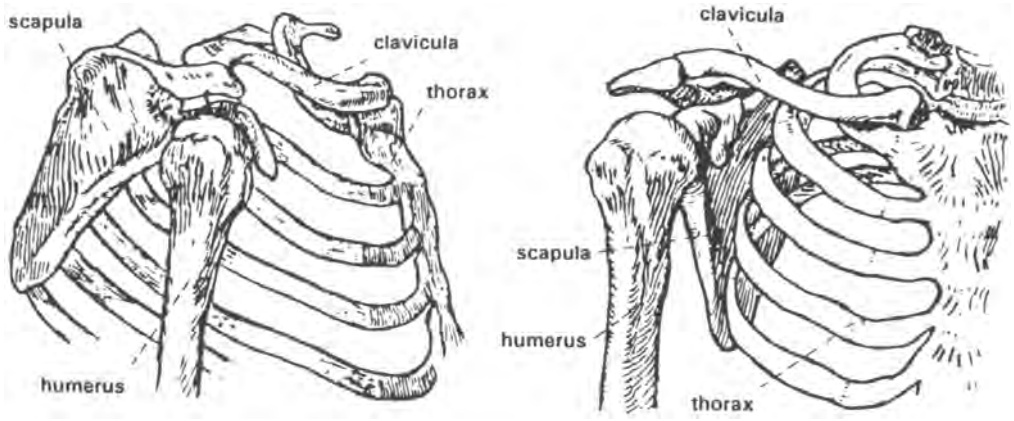


Figure 1 The skeletal structures of the shoulder. Illustration courtesy of the Delft Shoulder Group.

One of the reasons for this situation is the fact that the placement of shoulder prostheses is a difficult procedure. The procedure is complicated by two factors:

- The shoulder joint is a more complex mechanism than either the hip or the knee joints.
- A relatively small incision is made during the shoulder replacement operation. This small incision results in a limited field of view for the surgeon.

Figure 1 shows the skeletal structures in the human shoulder. The upper arm bone, or humerus, rotates against the glenoid, which is the shallow cup-like part of the shoulder-blade, or scapula. The humerus is held in the shoulder joint by a collection of muscles and tendons called the rotator cuff. The scapula itself is non-rigidly attached to the thorax via the collar-bone, or clavicula, and is able to slide over the thorax.

This flexible construction yields an impressive range of motion (ROM) for the shoulder joint, but as is almost always the case with increased complexity, is less robust than for instance the hip joint. The complexity of the shoulder joint contributes to the difficulty of performing a good replacement. Besides the fact that the precise functioning of this joint is not entirely clear, the surgeon has to cope with extremely limited visibility during the replacement operation. These factors contribute to the lower long-term success-rate of shoulder replacement.

In shoulder replacement, or shoulder arthroplasty, the humeral head and the glenoid are replaced with prostheses. Figure 2 shows an illustration of the skeletal structures in the human shoulder with implanted glenoid and humeral head prostheses.

Currently, the de facto standard for pre-operative planning for shoulder replacement is template-over-x-ray planning. The orthopaedic surgeon manually overlays transparencies with various prosthesis sizes and types on an x-ray of the patient's shoulder. Consequently, the surgeon decides on a particular prosthesis type and size, based only on a two-dimensional projection of the bony anatomy of the shoulder. In addition, this method of planning offers no patient-specific intra-operative guidance for prosthesis placement.

In order to improve the existing planning approach and hence also the outcome of shoulder replacement surgery, we have developed a flexible prototype surgical planning software system as well as linked mechanical guidance devices, also called patient-specific templates. The replacement operation is planned by making use of our pre-operative planning software. Integrating patient-specific joint motion simulation, the software is able to predict the post-

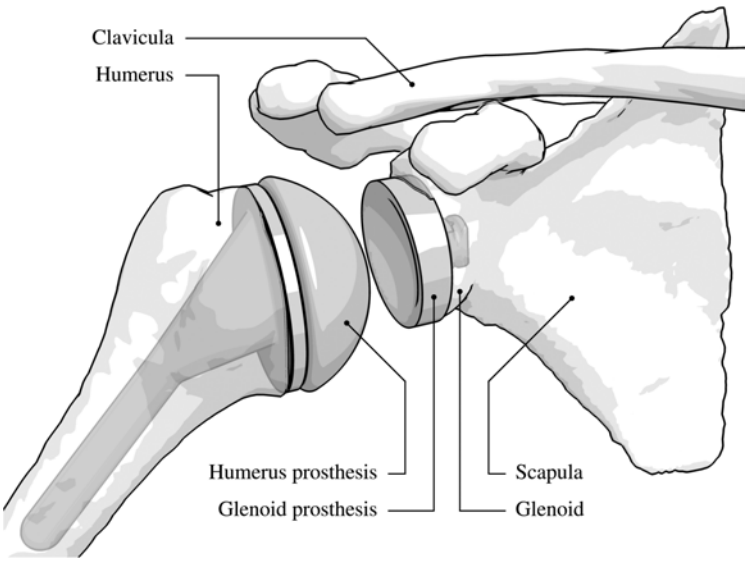


Figure 2 Illustration of a shoulder after an arthroplasty. Both the humeral head and glenoid prostheses have been virtually implanted, i.e. a total shoulder replacement has been performed.

operative patient joint range of motion (ROM). Based on this, the surgeon can optimize their planning. The parameters of the virtually performed operation, such as the insertion position and angle of the glenoid component, are used to design a patient-specific mechanical guidance device that is manufactured with rapid-prototyping techniques. The guidance device uniquely docks with easily identifiable bony structures in the patient's shoulder and is used for robust mechanical intra-operative guidance.

In this paper, we collect a number of our recent advances and show how they are integrated to form a complete pipeline for the pre-operative planning and intra-operative guidance of shoulder arthroplasty surgery. We present a new approach for the segmentation and accurate surface extraction of the bony structures from CT data of shoulder replacement patients. Due to its pathology, segmentation of the data offers unique challenges. However, the segmentation is required for the subsequent simulation steps. We then document the pre-operative approach we have developed, including the real-time post-operative joint range of motion (ROM) prediction and visualization. Finally, we show how the planning information can be used to design a patient-specific mechanical guidance device for glenoid component replacement. The range of motion simulation has been published elsewhere [13], and the idea of the shoulder guidance device has been presented previously [23]. The contribution of the present article consists of the shoulder segmentation technique, the *automatically* designed guidance devices and the integrated shoulder replacement planning and guidance pipeline.

2 Previous Work

We discuss previous and related work in two parts. In the first part we give an overview of techniques related to joint segmentation and in the second part we focus on planning and guidance for joint replacement.

2.1 Joint Segmentation

In previous research we developed a technique to determine the radius and centre of the humeral head using a modified Hough transform [24]. This is an effective approach and might be combined with our current work, but for pre-operative planning and ROM simulation we require complete and accurate surface representations of the skeletal structures in the shoulder.

Branzan-Albu et al. presented an approach to segment the skeletal structures from T1-weighted MRI images of healthy shoulders [3]. A separate segmentation was performed on each 2D slice and the results are combined to form a smooth 3D segmentation. The results are good but this technique has been developed for healthy shoulders with large joint space width and healthy bone density. In addition, it focuses on a small region of interest around the gleno-humeral joint.

A general approach for segmenting skeletal structures from CT data was presented by Kang et al. [11]. This approach is based on 3D region growing with locally adaptive thresholds following by a mixture of 3D and 2D morphological operations to close holes in the segmented surfaces. The resulting segmentation is then smoothed by adjusting its containing iso-surface. The approach is applied on the hip, the knee and the skull. The authors state that a site-specific approach is required for separating different bony structures at joints before their techniques can be applied. In the case of the hip, they make use of a manually-placed sphere. In our opinion, the determination of this site-specific separation method is one of the most complex tasks in the case of the shoulder.

Zoorofi et al. segmented hip joints by making use of histogram-based thresholding and an interesting hip-specific approach based on the convex hull of the initial thresholding and the application of the Hough transform [26]. This is a promising technique, but is very much hip-specific.

There are a number of examples in literature on segmenting skeletal structures from CT data, but very few of these address the problem of separating neighbouring skeletal structures, such as those found in joints. If we further narrow the problem to that of separating neighbouring skeletal structures from pathological CT data, even fewer examples remain. When we add the extra specification that it should deal with shoulder joints, there's almost nothing to be found. Our work attempts to fill this important and unexplored research area.

2.2 Planning and guidance for joint replacement

A wide range of pre-operative planning systems exist, for example Hip-Op [14], HipNav [20] and BrainLAB's VectorVision[1]. However, to our knowledge no such specific planning system for the shoulder joint is available at this time. Probable factors here are the complexity and the relatively lower number of replacements of the shoulder joint.

Most surgical guidance systems with a computer-based pre-operative planning stage make use of optical tracking systems during surgery. In our case, we couple computer-based pre-operative planning with automatically designed patient-specific mechanical guidance devices. The idea of manually designing patient-specific guidance devices or templates for surgical guidance has been explored for spinal surgery [7]. Our first manual designs for the shoulder guidance device have been presented [23]. In this article we present the first example

[1] http://www.brainlab.com/

of an automatically designed patient-specific guidance device that can be produced using rapid-prototyping techniques.

2.2.1 Range of Motion simulation

Joint range of motion (ROM) simulation aims to predict the post-operative motion patterns of a patient joint, based on pre-operative medical image data and the parameters of the planned surgical procedure.

Some research has been done on pre-operative ROM estimation for the hip joint. Jaramaz et al. use analytical modeling of the properties of implants to estimate both the ROM and the chance of dislocation, with bony impingement hardly playing a part [10]. The approach of Richolt et al. resembles our approach more closely, by applying collision detection to the 3D problem of bony impingement [17]. Their system is designed for osteotomy rather than joint replacement and only determines ROM for joint rotation along a single user-defined axis.

The simulator we describe in this paper predicts bone-determined ROM, i.e. soft tissue is not taken into account. For a full arthroplasty simulator we would require additional information on the presence of muscle tissue, ligaments and cartilage. Alternatively, a model of these aspects could be used, such as the Delft Shoulder and Elbow Model (DSEM) [25]. This is a musculoskeletal model of the shoulder and elbow joint that mainly focuses on muscle function and the involved forces and energy. However, the DSEM is not patient-specific.

In section 4.2 of this article we present a summary of our previous ROM prediction and visualization work [13] in order to serve as context for the full planning and guidance system.

3 Segmentation

The pre-operative planning, and especially the ROM simulation component, require high-quality surface meshes describing the bony surfaces in the human shoulder. These surfaces have to be extracted from CT data of patients with bone- and cartilage-affecting diseases. This, together with the complex geometry of the shoulder joint, results in a complex segmentation problem. Figure 3 shows a single axial slice from a typical dataset. In this case, joint space narrowing caused by arthritis results in the humerus and the scapula appearing fused. Decalcification, also related to the disease, in addition to the partial volume effect, leads to edges that are fuzzy and difficult to detect. Traditional approaches fail completely on this data.

Our approach combines a number of powerful techniques to separate complex touching structures by masking and inverse masking, then extracting initial surfaces that are larger than the objects that we wish to segment, and subsequently deflating deformable surfaces to fit precisely on the outside surface of those objects, all the while preserving the topology of the deforming surfaces [2]. Figure 4 is a flow chart illustrating the complete approach. In the following subsections, we will briefly explain each of these stages.

3.1 Deriving structure masks

The first step is to derive conservative structure masks for the structures that are to be segmented and in some cases also for structures that border on those that are to be segmented. We define a conservative structure mask as a list of *all* contiguous voxels that can be reasonably assumed to be contained within the boundaries of the structure that is being masked. Our use of *conservative* indicates that we prefer false negatives over false positives. In other

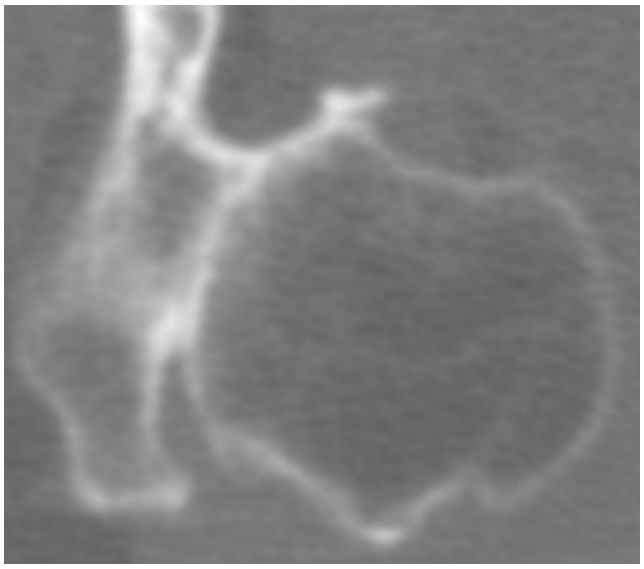


Figure 3 An axial slice from an arthritic shoulder CT dataset showing a humeral head and a scapula that appear to be fused together. This is due to the joint space being extremely narrow and the limited resolution of the CT scanner.

words, we prefer a mask that does not fully cover the structure of interest over one that covers the structure as well as some of its surroundings.

We use two main methods to generate these masks, indicated by the grey blocks marked *A* and *B* in Figure 4. Method *A* is used if structures do not touch and method *B* is used if they do. In most cases, both methods are used to generate a library of masks for each dataset. These masks can be combined to activate or to deactivate arbitrary parts of the dataset.

3.1.1 Method *A*

This method starts with an interactive histogram segmentation. This method is similar to that described in [9] and based on concepts presented in [12] and [18]. It makes use of a linked, or coupled, view that shows a 2-D histogram of image gradient magnitude over image intensity for the CT data that is being examined. By delineating regions on this histogram with closed polygons or splines, the user can select data points on the basis of both their intensity and gradient magnitude.

After this step, other masks from the library (for example generated by method *B*) can be used to remove unwanted components of the selection. Followed by a connected components analysis and our modified hole filling algorithm, filled and contiguous structure masks can be generated. Our modified hole filling algorithm is a standard morphological binary reconstruction hole-filling algorithm where we modify the convential marker image to deactivate one or more of the volume boundaries. By doing this, we allow for expected open cavities, such as the interior of the humerus, to be filled as well.

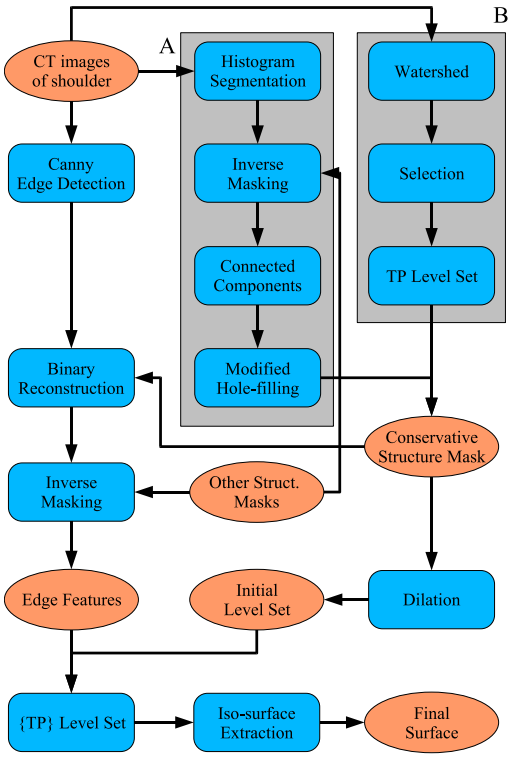


Figure 4 A flow chart illustrating our approach to the segmentation of the skeletal structures of the shoulder. Ovals signify data and blocks signify operations. The *A* and *B* blocks show two distinct ways to generate the initial conservative bone mask. *TP* signifies *Topology Preserving*. All these operations take place in 3D.

3.1.2 Method *B*

Starting with a 3D morphological watershed and basin merging step, this method is able to do an initial separation of touching complex objects, such as the humerus and scapula shown in Figure 3. During an interactive selection step, the user indicates which of the merged watersheds are relevant. The selected watersheds are combined and are used as an initial implicit surface during a level-set [19] based surface inflation step to refine the structure mask. The surface is inflated until it reaches the inside edge of bony structure that surrounds it.

3.2 Deformable surfaces

A 3D Canny edge detection [4] is applied to the input data. With the correct thresholds, this yields a volume containing a number of edge voxels. For each bony structure that we segment, we perform a morphological binary reconstruction of the edge voxels from the conservative mask for that structure. This operation removes all edge voxels that are disconnected from those voxels that touch the mask directly or indirectly (via connections with other edge voxels). Again masks from the library can be used after this step to remove any remaining edge voxels that are not relevant to the structure that we are segmenting. The result is a volume with voxels that are zero everywhere, except on the edges of the structure of interest where the voxels have unit value.

The conservative structure mask of the bony structure that we are segmenting is dilated so that it completely envelops all edge features. The dilated mask is converted to an implicit surface and then deflated using a topologically constrained [8] geodesic active contour [5] using the inverse of the edge features described above as the speed function. In other words, the speed function has unit value everywhere, except on the edges, where it has value zero. This causes the level set to continue deflating until it reaches an edge. The outside surface of both the scapula and the humerus have a topological genus of 0 in the majority of cases. We have integrated this knowledge into the segmentation pipeline by ensuring that we start with a genus 0 initial surface and constraining its topology as it deforms. This technique solves a number of problems related to the thinness and complexity of the scapula and the relatively low resolution of the CT data by preventing holes or isolated points from forming.

The segmentation process is repeated for each of the bones that need to be extracted. In general, we require models of the scapula and the humerus for pre-operative planning and ROM simulation.

4 Pre-operative Planning

During the planning phase, the glenoid and humeral components have to be selected and virtually implanted. During the virtual implantation, the system presents feedback to the surgeon about the structural quality of the planned procedure, e.g. whether there is sufficient contact between the prosthesis and the bone, as well as the functional quality of the planned procedure. Our system also simulates and visualises the ROM of the patient shoulder.

4.1 Virtual implantation

Docking two 3D objects using only a perspective or an orthogonal projection on a conventional screen is by nature quite difficult. In our case, we need precise control over the position and orientation of a prosthesis in order to implant it virtually into the surface model of the scapula.

In order to solve this problem, we made use of some basic CAD interaction techniques combined with the practical techniques used by orthopaedic surgeons during shoulder replacement surgery. Refer to Figure 5 during the following explanation. During a conventionally planned shoulder operation, the orthopaedic surgeon initially tries to align the glenoid prosthesis with the intersection between two imaginary planes. The first plane passes through the centre of the glenoid and the most lateral edge of the scapula. The second plane is roughly orthogonal to the first, parallel to the spina and passes through the centre of the glenoid. Experience has shown that this will help to achieve a post-operative centre of gleno-humeral rotation that is close to, or coincides with, a healthy centre of rotation. This result plays a very important role in the success of the replacement [6].

The planning software integrates a range of geometrical constraint-based object manipulation techniques based on points, axes and planes. With this functionality, the double plane insertion sketched above can easily be constructed and used as a first estimate for glenoid component placement. In order to do this, the surgeon will start by selecting three points: one in the centre of the glenoid and two on the lateral edge of the scapula. These points are shown in Figure 5a and are marked with respectively “Glenoid”, “Lateral 1” and “Lateral 2”. The point selection logic makes it easy to select points on surfaces and will also keep these latched to the surface when they are moved. A scapula lateral edge slice can now be created by making use of these three points as a plane definition. The system will warn if the three selected points do not uniquely define a plane. See Figures 5 and 6 for an example

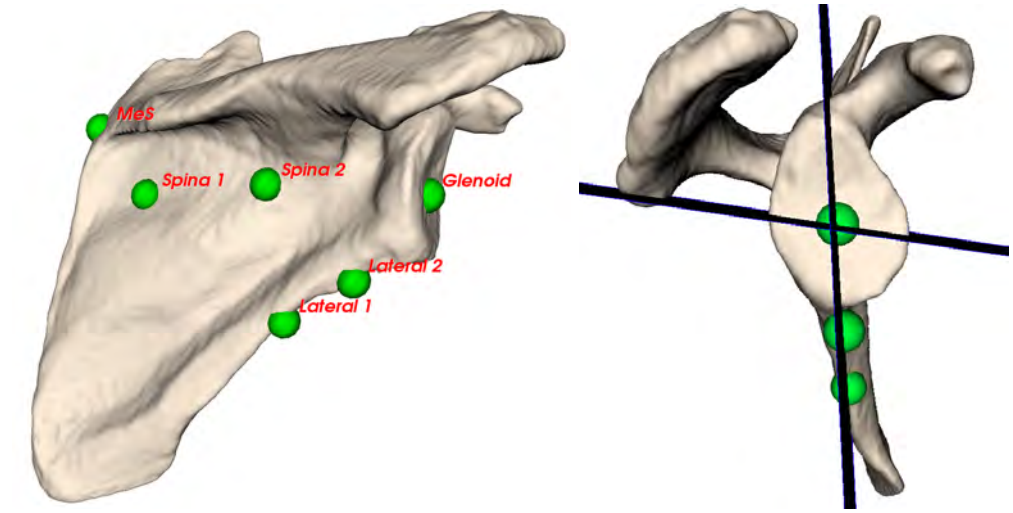


Figure 5 In (a), a scapula is shown with the points that are required to place the two anatomical planes. In (b), the same scapula is shown with the resultant anatomical planes intersecting to form an insertion axis for the glenoid prosthesis.

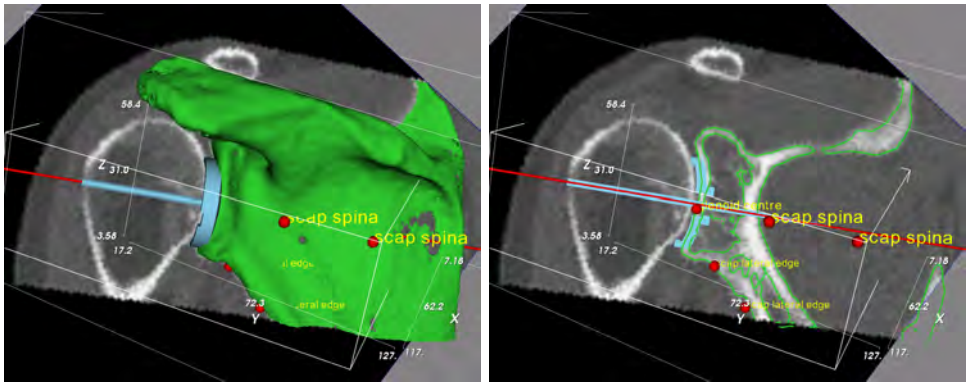
of such a plane intersecting the scapula. It can also happen that the plane defined by these three points does not have the desired orientation. In our experience, substituting one of the lateral edge points with the point where the spina meets the medial edge of the scapula, labeled "MeS" in Figure 5a, solves this problem.

By repeating this process, but instead making use of the point on the centre of the glenoid and two points on the posterior side of the scapula approximately equidistant from the spina, a second plane can be defined. These two extra points are labeled "Spina 1" and "Spina 2" in Figure 5a. The intersection of these two planes constitute a very good first estimation of the glenoid insertion axis. This plane intersection is shown in Figure 5b.

The constraint system can now automatically align the prosthesis axis with the intersection of the two planes and optionally constrain all prosthesis motion to the plane intersection. The object with the horizontal pin in Figure 6a represents the glenoid prosthesis. Logically the constraint system can also work with a single plane: in this case the prosthesis can be aligned with the plane, i.e. be embedded in it, and optionally all prosthesis motion can be constrained to the plane. This is the approach that has been chosen in Figure 6.

During the final prosthesis manipulation, objects can be hidden and object contouring activated, so that the object intersection curves with the anatomical planes can be seen overlaid on the CT-data slices. This is shown in Figure 6b. In this mode, the volume slice can still be moved to and fro. This enables the surgeon to judge the fit of the prosthesis with regard to the bone volume that is visible in the CT-scan. This view is more familiar to surgeons, as it is strongly reminiscent of the template-over-x-ray planning method.

The humeral prosthesis is placed in accordance with the actual surgery. During the procedure, a metal ring is fitted on the humeral head. This defines the location of the centre of rotation as well as the size of the humeral prosthesis. A pin is drilled into the bone through a guidewire at the center of the ring. On this pin a metal block is mounted. When sawing off part of the humeral head, the surgeon follows the plane as imposed by the metal block.



(a) (b)

Figure 6 Part of the glenoid component planning functionality. In (a) the complete models are shown and in (b) only their contours. The contour view enables the user to judge whether the prosthesis has a good fit and is reminiscent of the conventional template-on-x-ray planning. The slice is anatomically oriented but can be moved to check all parts of the glenoid component.

In the pre-operative planning system, we fit a sphere through the humeral head. Virtual models of the pin and metal block are positioned using the centre of rotation defined by the sphere. The cutting plane defined by the metal block is applied and the prosthesis is placed at the centre of the intersection plane. This completes the placement of the prosthesis. All of the parameters can be adjusted at any time during the simulation.

4.2 ROM simulation and visualization

Throughout the surgeon's interaction, the predicted patient-specific range of motion (ROM) can be calculated and visualized [13]. The ROM is bone-determined in that only the bone geometry playes a role in the simulation. Soft tissue does not play any role. The gleno-humeral ROM is visualized with motion envelopes, that indicate the maximum ROM of the humerus in every direction. In addition, a novel visualization technique depicts the differences between the current and previous ROM during the surgeon's interaction.

In order to calculate the ROM using a segmented CT dataset, we implemented a simplified bio-mechanical model of the gleno-humeral joint. A generally accepted hypothesis is that the gleno-humeral joint can be approximated by a ball-joint [16, 24]. We combined this model with collision-detection on surface models of the skeletal structures in the patient's shoulder.

Our simulator is capable of handling hemi-protheses, which is a prosthesis without a glenoid component, and also reversed prostheses, where the spherical component is placed at the glenoid.

For all prostheses, the ROM envelopes are constructed in the following way. The humerus is aligned with an initial orientation, which will be the starting alignment for all iterations. The simulation consists of two nested iterations. During the outer iteration, the humerus is rotated around the axis marked with A_i in Figure 7. Axis A_j rotates along with A_i.

At each rotation, the maximum possible orientation of the humerus around axis A_j is found by making use of a binary search. By repeatedly dividing the search interval in half, our ROM determination executes in $O(\log n)$, where n relates to the effective resolution of the end result of the binary search. Whenever colliding polygons are detected, we reverse the

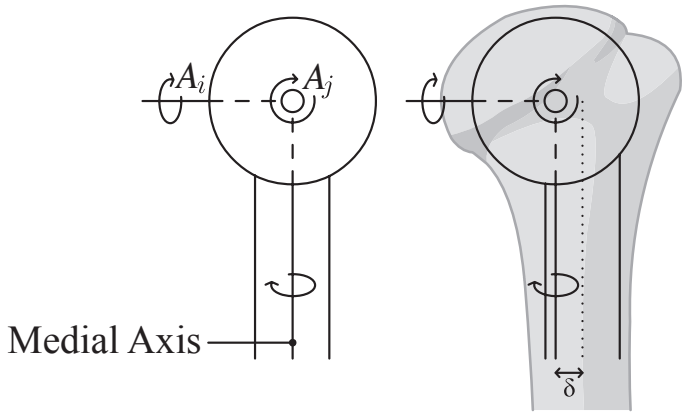


Figure 7 A schematic representation of the humerus and its rotational axes. As can be seen in this image, the additional ROM as the result of endo/exorotation is related to δ, the distance between the medial axis of the humerus and the vertical axis through the center of rotation.

search direction. We use OPCODE [21] (Optimized Collision Detection), an efficient library based on memory-optimized bounding-volume hierarchies to detect colliding polygons.

A complete new ROM has to be determined for each change made by the surgeon that affects the bone or prosthesis geometry, for example a change in the position of the humeral prosthesis.

In order to enable the real-time use of the ROM simulation, we have developed two optimization approaches. In the interpolation approach, a large number of ROM envelopes are precalculated so that ROM envelopes can be interpolated during interaction. In the collision clipping approach we make use of a clipping plane to clip collisions, thus rendering the usual re-initialization of the collision detection data structures during each ROM calculation unnecessary. This same clipping plane is uploaded to the graphics hardware to clip the geometry.

4.2.1 Visualization

Once all directions of our ROM envelope have been probed for their maximum angles, we can begin constructing the ROM visualization. We draw lines between the center of rotation and an arbitrary point within the end of the shaft of the humerus, and then transform these lines according to their respective maximum angles. The resulting envelope is shown in Figure 8.

For each change made to the virtual prosthesis placement by the surgeon, a new ROM envelope is calculated and visualized. This enables the surgeon to visualise directly the complete shoulder ROM for a particular set of operational parameters. Being able to see the envelope update in real-time as changes are being made, helps the surgeon to investigate the effect of even small changes to the planned operation.

Several parameters that define the placement of the humerus prosthesis can be adjusted during the interaction. First of all, the cutting plane at the humeral head can be translated along its normal, as well as rotated around two axes perpendicular to the normal. Also, the position of the humerus prosthesis relative to the humerus can have a small offset in any direction within the cutting plane.

In general, the placement of the glenoid component is constrained by the quality and the

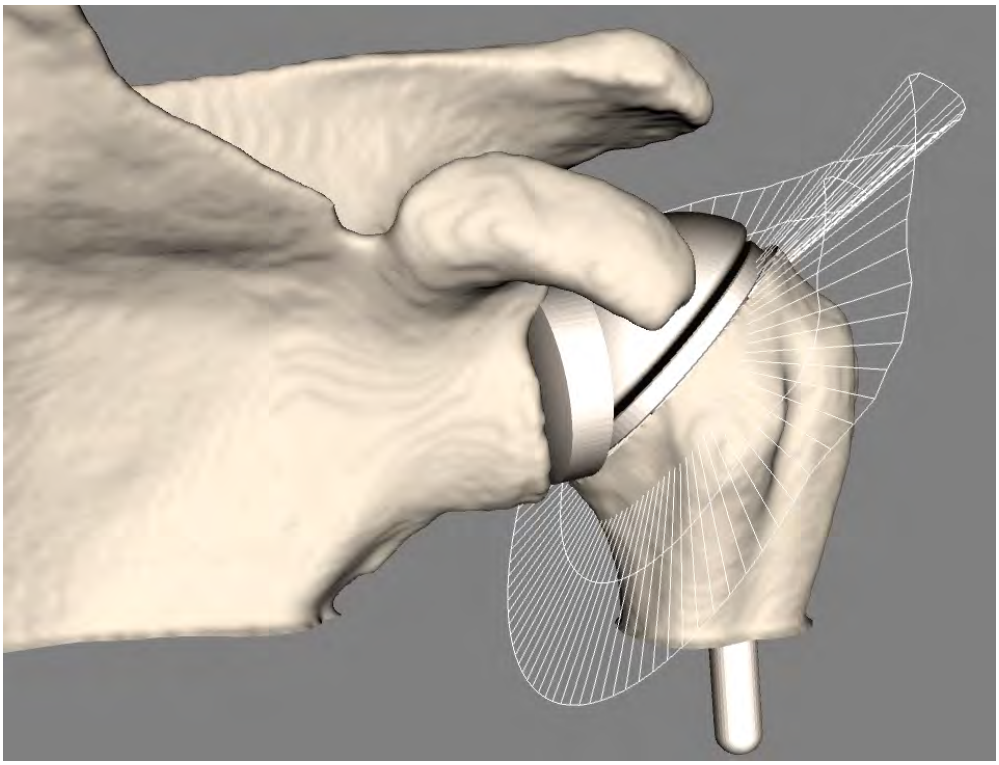


Figure 8 The visualization of ROM by means of envelopes.

geometry of the scapula and takes place according to the method described in Section 4.1. Therefore our ROM simulation and visualization focuses on changes in the humeral component placement, but applying it to glenoid placement changes would be straight-forward.

In order to facilitate this important investigation of the increase or decrease in ROM that results from a particular change in the planning, we have implemented comparative visualization functionality whereby the difference between two ROM envelopes can be explicitly visualized. The comparative visualization is also updated in real-time as the surgeon interacts with the planning.

For two consecutive envelopes we depict improvements and deteriorations by connecting the lines with colored polygons. A red polygon denotes that the most recent envelope has a more limited ROM in that particular direction than the reference envelope, while a green polygon states the opposite. Additionally, the end points of the lines are connected with arrows, pointing towards the most recent envelope. The resulting visualization is shown in Figure 9. The reference envelope can be set to the current or any previously determined ROM envelope at all times.

5 Guidance Device

The guidance device is a crucial link between the pre-operative planning phase and the execution of the procedure in the operating room. During pre-operative planning, the size and type of prosthesis can be determined and this knowledge can easily be applied during the actual replacement. Prosthesis position and orientation, however, require some extra ingenuity.

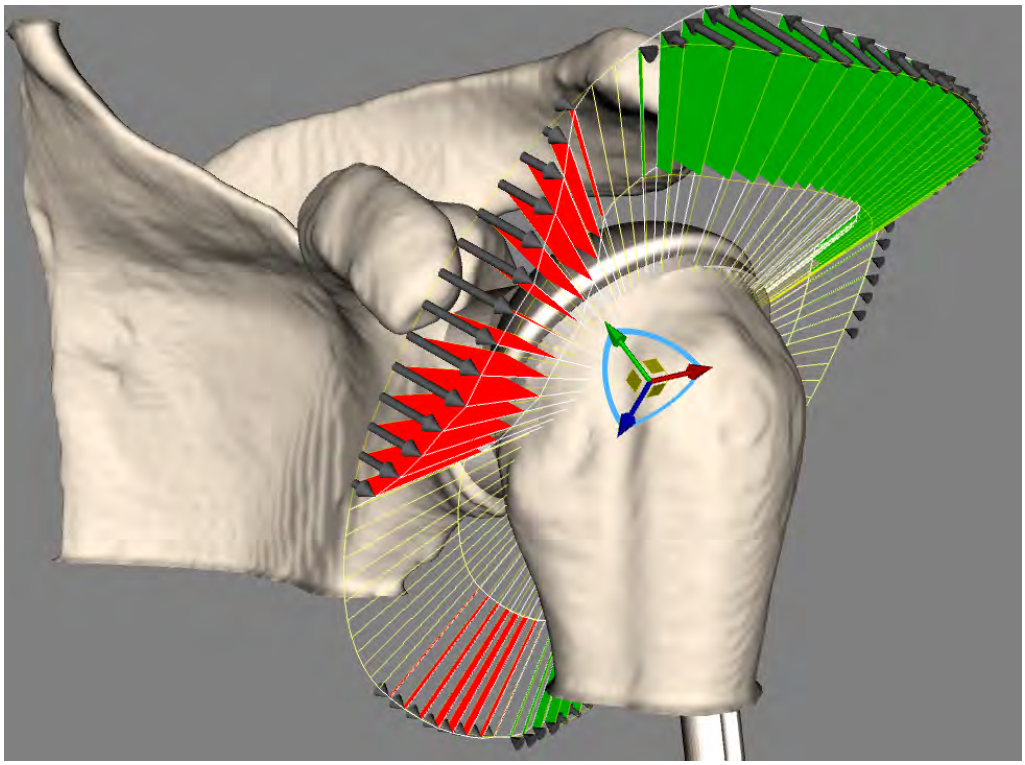


Figure 9 Comparative visualization of two ROM envelopes. The first envelope is a previously determined ROM which was set as a reference envelope by the user. The second envelope is continuously updated for every adjustment applied to the prosthesis placement parameters.

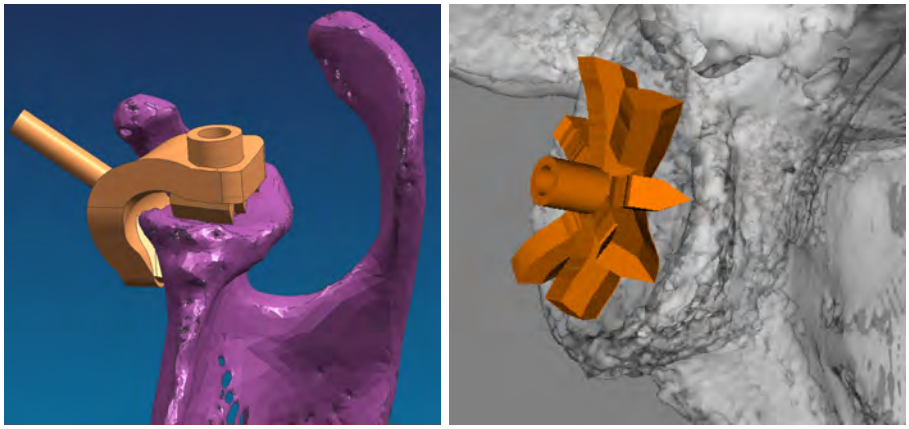
In order to realise this planning in the operating room in a robust and efficient fashion, we are working on applying principles for the design of pedicle screw drill guides [7] to the design of a patient-specific guidance device for shoulder replacement.

The patient-specific mechanical guidance device is a jig, or template, that can be used by the surgeon during the operation to guide the use of another tool such as a drill or a pneumatic saw. In the case of a glenoid replacement, the template uniquely fits the patient's scapular glenoid and has a conduit for a drill, thus acting as a drill-guide. By making use of this template, the surgeon can ensure that the pre-operatively planned glenoid insertion position and orientation are realized. In this way, a cheap and robust coupling between the computer-based intra-operative planning and the operation itself is realized.

Figure 10a shows the latest design of our glenoid drill guide. Figure 10b shows a guidance device that our planning software has designed for the patient and the planned implantation. The guidance device is manufactured with a rapid-prototyping machine.

6 Results

In order to evaluate the results of our segmenation approach, we started by comparing its results on an experimental CT dataset of an *ex vivo* scapula to a manual expert segmentation. The manual segmentation was performed by drawing a contour describing the outside scapular surface on each axial slice of the $512 \times 512 \times 420$ dataset, sampled at a resolution of $0.333 \times 0.333 \times 0.5$mm, and then linking these contours together with triangles. The resultant mesh consisted of 6900 triangles, was closed and it was of genus 0 topology.



(a) (b)

Figure 10 In (a), the latest design of the glenoid drill guide is visualized on a surface model of a scapula, courtesy of Liesbet Goossens and Bart de Schouwer, Katholieke Universiteit Leuven, Belgium. In (b), another design that has been automatically made patient-specific by the planning software is shown.

We used our method to segment the scapular mesh from the same data. The resultant mesh consisted of 729212 triangles, was closed and was also of genus 0 topology. The large number of triangles is the usual result of applying the Marching Cubes algorithm to a volume of this resolution to extract a surface of this complexity. We did not apply any decimation techniques to the mesh.

We compared the meshes using the Mesh software [1]. The maximum, mean and root-mean-squared (RMS) symmetric Hausdorff distances are 3.7, 0.4 and 0.5mm respectively. The symmetric distance is the maximum of the distances from the first mesh to the second mesh and from the second mesh to the first. The distances as a percentage of the largest diagonal of the bounding box of the scapula are 1.6, 0.2 and 0.2% respectively. The RMS error is in the order of a single voxel. We find this result quite acceptable for a manual vs. automatic reconstruction.

Most of the outliers, and specifically the parts of the mesh that cause the maximum Hausdorff distance, are caused by cartilage being misclassified as cortical bone. This is especially a problem with cadaveric material, as the cartilage seems to harden over time and thus appears in the CT data with greater intensity.

We have also tested the complete prototype planning system, including the segmentation, on a number of in-vivo shoulder CT datasets of patients requiring total shoulder replacements. Feedback from the operating orthopaedic surgeon, who used the system primarily for pre-operative exploration, was positive. The fact that our bone-determined ROM simulation can be used to predict impingement, i.e. when due to prosthesis placement joint motion is restricted by bone colliding with bone, is seen as particularly important.

We benchmarked the interactive performance of the ROM simulator on a Pentium 4 running at 2.66 GHz with 512 MB of RAM. The humerus model consisted of 50.000 polygons, while the scapula consisted of 155.000 polygons.

Performance figures for the interactive ROM simulation are listed in Table 1. As can be seen, the speed increase due to collision clipping varies from a factor of 1.33 to 8.22 for normal usage of the simulator. The speed increase due to interpolation varies from a factor of

Table 1 Speed of simulation and rendering in updates per second. All performance figures are specified as updates per second, figures in parentheses refer to speedups relative to "no optimization" performance.

	With rendering	Without rendering
no optimization	0.27	0.30
collision clipping	2.22 (× 8.22)	2.69 (× 8.97)
interpolation	10.81 (× 40.04)	374.4 (× 1248)

10.74 to 40.04. If we discard the graphical representation, the speed increase by interpolation gets as high as a factor of 1248.

The precalculation interpolation optimization leads to a great speed-up, but can only be used if suffcient time for pre-processing is available (approximately 20 minutes for the cases presented here) and if the surgeon's changes are limited to a known set of possibilities. The collision clipping optimization leads to a less significant speed-up, but can be applied in all cases and does enable the interactive use of the ROM simulation.

We have performed initial validation experiments on cadaver scapulae to test the design and functioning of the drill-guide [23]. The results were promising, but do require improvement, as the cartilage covering the glenoid has a significant impact on the orientation of the resultant implant. The drill guide is designed based on the bony surface of the glenoid and not the cartilage covering it. Cartilage will be taken into account in future experiments.

7 Conclusions and Future Work

In this paper we have described a complete visualization-oriented process for the pre-operative planning and intra-operative guidance of shoulder replacement surgery.

We presented an approach for segmenting the bony structures from CT data of arthritic patients. Existing methods, mostly designed for other joints, all fail on this type of data. We then showed how these segmentations are used in an interactive pre-operative planning solution that assists the surgeon in virtually implanting shoulder prostheses. The pre-operative planning software gives real-time feedback on structural and functional aspects of the planned procedure. The bone-determined shoulder ROM is calculated and visualized interactively, so that the surgeon gets feedback on every small change made to the planning. Changes in ROM are also explicitly visualized, helping the surgeon to fine-tune the planning.

The CT data and pre-operative planning are used to automatically generate a patient-specific mechanical guidance device that can be used during the replacement operation to execute the planning. To increase accuracy and reliability of the guidance devices during surgery, we will focus on improved imaging of cartilage, and refine the modelling to improve mechanical support from the cartilage.

We are currently working on a new segmentation approach that exploits image-based curvature in combination with a priori knowledge of the shoulder anatomy, leading to more robust and accurate segmentation results.

The ROM simulator will be further refined to include effects of soft tissue and active motion on joint mobility. Reference standards for mobility of healthy joints will be developed to estimate the amount of functionality to be restored after joint replacement. We are currently validating aspects of the ROM simulator as part of a cadaveric study.

We are also preparing for a series of clinical studies in using our pre-operative planning system for actual replacement surgery. A thorough evaluation of all stages of the pipeline

will be carried out, including post-operative functional tests, to compare the overall results of the process with current clinical practice.

References

1 N. Aspert, D. Santa-Cruz, and T. Ebrahimi. Mesh: Measuring error between surfaces using the Hausdorff distance. In *Proceedings of the IEEE International Conference on Multimedia and Expo 2002 (ICME)*, pages 705–708, 2002.

2 Charl P. Botha. *Techniques and Software Architectures for Medical Visualisation and Image Processing*. PhD thesis, Delft University of Technology, 2005.

3 Alexandra Branzan-Albu, Denis Laurendeau, Luc Hébert, Helene Moffet, Marie Dufour, and Christian Moisan. Image-guided analysis of shoulder pathologies: Modeling the 3D deformation of the subacromial space during arm flexion and abduction. In Dimitris Metaxas Stephane Cotin, editor, *International Symposium on Medical Simulation (ISMS 2004)*, volume 1 of *Lecture Notes in Computer Science*, pages 193–202, Cambridge, MA, USA, June 17-18 2004. Springer Verlag.

4 J. Canny. A computational approach to edge detection. *IEEE Trans. PAMI*, 8(6):679–698, 1986.

5 Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

6 O. de Leest, P.M. Rozing, L.A Rozendaal, and F.C.T. van der Helm. The influence of glenohumeral prosthesis geometry and placement on shoulder muscle forces. *Clinical Orthopedics*, 330:222–233, 1996.

7 J. Goffin, K. Van Brussel, K. Martens, J. Vander Sloten, R. Van Audekercke, and M.H. Smet. Three-dimensional computed tomography-based, personalized drill guide for posterior cervical stabilization at c1–c2. *Spine*, 26(12):1343–1347, 2001.

8 Xiao Han, Chenyang Xu, and Jerry L. Prince. A topology preserving level set method for geometric deformable models. *IEEE Trans. PAMI*, 25(6):755–768, 2003.

9 F. Vega Higuera, N. Sauber, B. Tomandl, C. Nimsky, G. Greiner, and P. Hastreiter. Enhanced 3D-visualization of intracranial aneurysms involving the skull base. In R.E. Ellis and T.M. Peters, editors, *MICCAI 2003*, volume LNCS 2879, pages 256–263. Springer-Verlag Heidelberg, 2003.

10 B. Jaramaz, C. Nikou, D. Simon, and A.M. Di Gioia. Range of motion after total hip arthroplasty: Experimental verification of the analytical simulator. Technical Report CMU-RI-TR-97-09, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, February 1997.

11 Yan Kang, Klaus Engelke, and Willi A. Kalender. A new accurate and precise 3-d segmentation method for skeletal structures in volumetric ct data. *IEEE Trans. on Med. Imaging*, 22(5):586–598, 2003.

12 Gordon Kindlmann and James W. Durkin. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. In *IEEE Symposium on Volume Visualization*, pages 79–86, 1998.

13 Peter R. Krekel, Charl P. Botha, Edward R. Valstar, Paul W. de Bruin, Piet M. Rozing, and Frits H. Post. Interactive simulation and comparative visualisation of the bone-determined range of motion of the human shoulder. In *Proceedings of Simulation and Visualization*, March 2006.

14 R. Lattanzi, M. Petrone, P. Quadrani, C. Zannoni, and M. Viceconti. Applications of 3d medical imaging in orthopaedic surgery: Introducing the hip-op system. *Proc. First International Symposium on 3D Data Processing Visualization and Transmission*, pages 808–811, June 2002.

15 H. Malchau, P. Herberts, P. Söderman, and A. Odén. Prognosis of total hip replacement. Update and validation of results from swedish national hip arthroplasty registry 1979–1998. In *Scientific exhibition presented at the 67th AAOS meeting*, 2000.

16 C.G.M. Meskers, F.C.T. van der Helm, L.A. Rozendaal, and P.M. Rozing. *In vivo* estimation of the glenohumeral joint rotation center from scapular bony landmarks by linear regression. *Journal of Biomechanics*, 31:93–96, 1998.

17 J.A. Richolt, M. Teschner, P. Everett, B. Girod, M.B. Millis, and R. Kikinis. Planning and evaluation of reorienting osteotomies of the proximal femur in cases of scfe using virtual three-dimensional models. In *MICCAI '98*, pages 1–8, London, UK, 1998. Springer-Verlag.

18 Iwo Serlie, Roel Truyen, Jasper Florie, Frits Post, Lucas van Vliet, and Frans Vos. Computed cleansing for virtual colonoscopy using a three-material transition model. In R.E. Ellis and T.M. Peters, editors, *MICCAI 2003*, volume LNCS 2879, pages 175–183. Springer-Verlag Heidelberg, 2003.

19 J.A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 2nd edition, 1999.

20 D.A. Simon, B. Jaramaz, M. Blackwell, F. Morgan, A.M. DiGioia, E. Kischell, B. Colgan, and T. Kanade. Development and validation of a navigational guidance system for acetabular implant placement. *Proc. of the First Joint CVRMed / MRCAS Conference*, pages 583–592, 1997.

21 Pierre Terdiman. Memory-optimized bounding-volume hierarchies. 2001. `http://www.codercorner.com/Opcode.htm`.

22 M.E Torchia, R.H. Cofield, and C.R. Settergren. Total shoulder arthroplasty with the neer prosthesis: long-term results. *Journal of Shoulder and Elbow Surgery*, 6(6):495–505, 1997.

23 E.R. Valstar, K van Brussel, B.L. Kaptein, B.C. Stoel, and P.M. Rozing. CT-based personalized templates for accurate glenoid prosthesis placement in total shoulder arthroplasty. In *Proc. CAOS*, 2003.

24 Marjolein van der Glas, Frans M. Vos, Charl P. Botha, and Albert M. Vossepoel. Determination of Position and Radius of Ball Joints. In Milan Sonka, editor, *Proceedings of the SPIE International Symposium on Medical Imaging*, volume 4684 - Image Processing, 2002.

25 Frans C.T. van der Helm. A finite element musculo-skeletal model of the shoulder mechanism. *Journal of Biomechanics*, 27(5):551–569, 1994.

26 Reza A. Zoroofi, Yoshinobu Sato, Toshihiko Sasama, Takashi Nishii, Nobuhiko Sugano, Kazuo Yonenobu, Hideki Yoshikawa, Takahiro Ochi, and Shinichi Tamura. Automated segmentation of acetabulum and femoral head from 3-d ct images. *IEEE Transactions on Information Technology in Biomedicine*, 7(4):329–343, 2003.

Patient-Specific Mappings between Myocardial and Coronary Anatomy

Maurice Termeer[1], Javier Oliván Bescós[2], Marcel Breeuwer[2], Anna Vilanova[3], Frans Gerritsen[2], M. Eduard Gröller[4], and Eike Nagel[5]

1 **Vienna University of Technology**
`maurice@cg.tuwien.ac.at`
2 **Philips Healthcare**
{`javier.olivan.bescos,marcel.breeuwer,frans.gerritsen`}`@philips.com`
3 **Eindhoven University of Technology**
`a.vilanova@tue.nl`
4 **Vienna University of Technology**
`groeller@cg.tuwien.ac.at`
5 **King's College London**
`eike.nagel@kcl.ac.uk`

Abstract

The segmentation of the myocardium based on the 17-segment model as recommended by the American Heart Association is widely used in medical practice. The patient-specific coronary anatomy does not play a role in this model. Due to large variations in coronary anatomy among patients, this can result in an inaccurate mapping between myocardial segments and coronary arteries. We present two approaches to include the patient-specific coronary anatomy in this mapping. The first approach adapts the 17-segment model to fit the patient. The second approach generates a less constrained mapping that does not necessarily conform to this model. Both approaches are based on a Voronoi diagram computation of the primary coronary arteries using geodesic distances along the epicardium in three-dimensional space. We demonstrate both our approaches with several patients and show how our first approach can also be used to fit volume data to the 17-segment model. Our technique gives detailed insight into the coronary anatomy in a single diagram. Based on the feedback provided by clinical experts we conclude that it has the potential to provide a more accurate relation between deficiencies in the myocardium and the supplying coronary arteries.

1998 ACM Subject Classification I.3.8 Applications, J.3 Life and Medical Sciences

Keywords and phrases Voronoi Diagram, Segmentation, Myocardium

Digital Object Identifier 10.4230/DFU.SciViz.2010.196

1 Introduction

In the field of cardiac tomography, the American Heart Association (AHA) has published a set of recommendations to standardize part of the clinical routine [4]. These recommendations concern the segmentation of the left ventricular myocardium, the heart muscle of the left ventricle. They also include a standardized nomenclature for the coronary arteries, small arteries surrounding the left ventricle that supply the myocardium with oxygenated blood. One of these recommendations is the use of a 17-segment model for the segmentation of the myocardium. This model defines a mapping of each of the segments to the coronary arteries that supply that region. The relation between the myocardium and its supplying coronary arteries is important during diagnosis.

The individual paths of the coronary arteries and the way they supply the myocardium varies greatly among different patients. This causes the standardized mapping between myocardial segments

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 196–209

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

and coronary arteries to be of varying accuracy across different patients. This issue is acknowledged by the American Heart Association, with the note that the suggested model is based upon the methods available at that time [4].

With current MRI (Magnetic Resonance Imaging) technology it is possible to segment the three major coronary arteries from a whole heart scan [11]. The three primary coronary arteries are the left anterior descending (LAD), the left circumflex (LCX) and the right coronary artery (RCA). While there are still cases where a complete segmentation is not feasible, these issues will most likely be resolved in the near future.

We use the patient-specific coronary anatomy to improve the relation between the myocardium and the supplying coronary arteries. We explore two possible approaches. In our first approach we adapt the 17-segment model as defined by the AHA to fit to the coronary anatomy of the patient in question. In other words, we set a number of constraints to force the mapping between the myocardium and the areas the coronary arteries are supplying to correspond to the 17-segment model. In our second approach we do not set any constraints and thus allow for arbitrary mappings between the coronary arteries and the myocardium. Both approaches are based on Voronoi diagrams of the coronary arteries using geodesic distances along the epicardium, the outer part of the myocardium. The resulting relation divides the myocardium into several coronary territories, regions of the myocardium supplied by the coronary artery that corresponds to that territory. These territories, along with the coronary arteries themselves, are finally projected onto a two-dimensional bull's eye plot.

Both of our approaches result in customized diagrams that better reflect the actual relation between myocardial tissue and supplying coronary arteries. The approach of adapting the 17-segment model is less sensitive to incomplete segmentations of the coronary anatomy. On the other hand, unconstrained coronary territories may be more accurate for patients that do not correspond well to the 17-segment model.

Our article is structured as follows. We first give an overview of related work on the 17-segment model from the American Heart Association and patient-specific coronary territories in Section 2. In Section 3 we discuss the computation of the coronary territories and the projection on a bull's eye plot. In Section 4 we discuss both approaches to computing patient-specific coronary territories. In Section 5 we describe the results of an evaluation experiment and provide a use case of our techniques with viability data. In Section 6 we discuss the medical applications and issues of our approach. Finally, in Section 7 we conclude our work.

2 Related Work

The American Heart Association (AHA) published a set of recommendations to standardize the segmentation of the left ventricular myocardium [4]. This includes a division of the myocardium into 17 parts accompanied by a mapping of each of the segments to coronary territories. As a reference this 17-segment model is depicted in Figure 1. It shows the myocardium in a bull's eye plot, a two-dimensional representation of the left ventricle. Since its introduction, the 17-segment model has become widely accepted in clinical practice and it has replaced previous models. In the area of SPECT, for example, a 20-segment model [1] was more common prior to the introduction of the 17-segment model.

A medical study performed by Pereztol-Valdés *et al.* [10] using myocardial perfusion nuclear imaging gives a more precise relation between each of the 17 segments and the supplying coronary arteries. It shows that contrary to the model of the AHA, only nine segments are commonly supplied by one coronary artery; the remaining segments are supplied by multiple coronary arteries. It also verifies that there is great variability among patients, especially in the apical area of the left ventricle. Based on the outcome of this study, the authors presented a revised mapping of segments to coronary

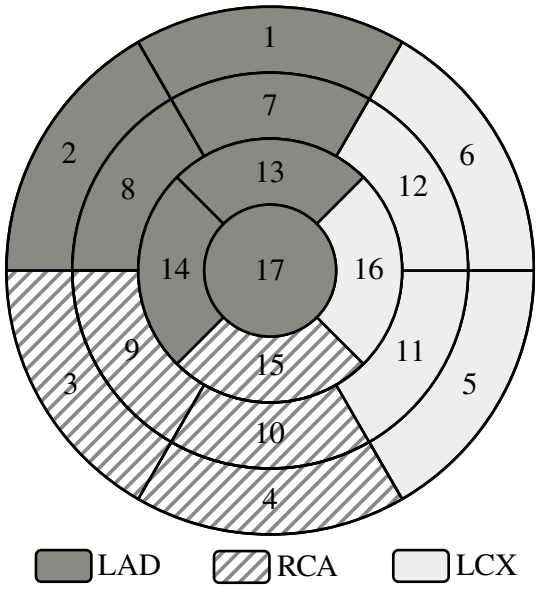


Figure 1 The 17-segment model for the segmentation of the left ventricle as presented by the American Heart Association [4]. The patterns indicate the mapping of the segments to coronary territories.

territories, where some segments are assigned to multiple territories. Recently a similar study was published by Ortiz-Pérez *et al.* [9] using contrast-enhanced MRI. Their work also confirmed the variability in coronary anatomy and the need for a better model to more accurately attribute perfusion defects in segments to coronary arteries.

Beliveau *et al.* [2, 3] previously presented the idea of computing patient-specific coronary territories. In their work coronary arteries are segmented from a whole heart CT scan and projected on the segmented epicardial surface. The patient-specific coronary territories are computed by using a Voronoi diagram of the projected arteries. While their work is in several ways similar to ours, there are several key differences. Besides a different implementation approach and a different method to project the territories on a bull's eye plot, no explicit relation between the 17-segment model and the coronary territories is made, although the 17-segment model is mentioned in their work. Their approach is thus not capable of forcing correspondence to this model. We demonstrate how we can use this to regularize a bull's eye plot of viability data (see Section 5.2), while their approach is limited to showing an overlay. In other work, Voronoi diagrams are also being used for the analysis of modeled and segmented artery trees [6].

Oeltze *et al.* presented several novel visualization techniques for the analysis of perfusion data [8]. Their redefined bull's eye plot shows myocardial perfusion data of both rest and stress states in a single bull's eye plot based on the 17-segment model. They also presented an interactive bull's eye plot that is linked to a three-dimensional visualization showing coronary artery branches. Picking a segment on the bull's eye plot automatically adjusts the view of the three-dimensional view to show the corresponding artery.

3 Computation of Coronary Territories

A coronary territory defines the region of the myocardium supplied by a specific coronary artery. Determining which part of the myocardium a coronary artery is supplying is a complex and yet unsolved problem. We approach this by assuming that each part of the myocardium is supplied

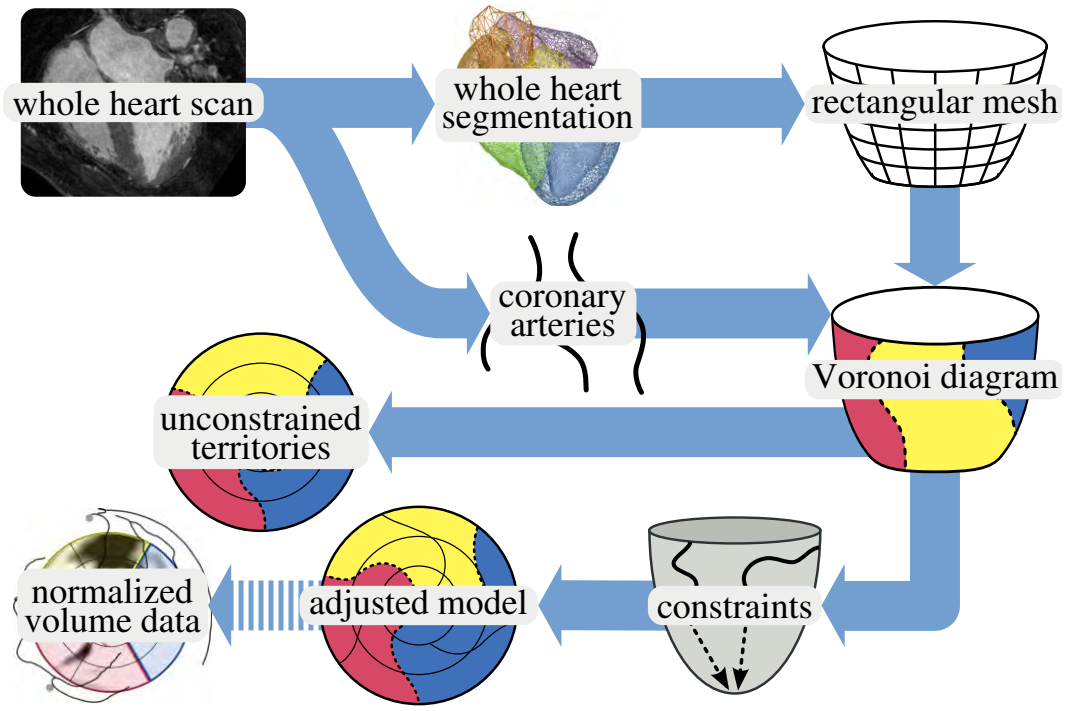


Figure 2 Overview of our approach towards patient-specific coronary territories.

by its closest coronary artery. This approximation suffices for our purposes. We also perform all computations on the epicardial surface, instead of taking the three-dimensional nature of the myocardium into account.

An overview of our approach is shown in Figure 2. We obtain a whole heart segmentation from a whole heart MRI scan using an approach by Ecabert *et al.* [5]. Using only the left ventricular epicardium part of this segmentation, we create a rectangular mesh of the epicardium to gain more control over the resolution and uniformity of the mesh. We track the centerlines of the three primary coronary arteries in the same whole heart scan using a semi-automatic vessel tracking method [7] and project them onto the mesh of the epicardium. We subsequently compute a Voronoi diagram of these projected arteries. This diagram is finally projected on a bull's eye plot.

Prior to computing the Voronoi diagram, we can set a number of constraints to force the resulting division of the epicardium into coronary territories to correspond to the 17-segment model. In this case we can use the borders and middle lines of the coronary territories to adapt the 17-segment model to the patient. We can then also use the deviation between the original and adapted models to deform data from a different scan of the same patient to fit to the original 17-segment model. Without setting any constraints, the resulting coronary territories can have arbitrary shapes and need not correspond to the 17-segment model. Both these approaches are discussed in more detail in Section 4.

3.1 Mesh of the Epicardium

Our automatic segmentation algorithm gives a segmentation of the left ventricular epicardium in the form of an unstructured polygonal mesh. In order to guarantee a sufficient accuracy throughout the mesh during the Voronoi diagram computation we generate a rectangular mesh instead. We intersect the unstructured mesh with two sets of planes. One set consists of evenly spaced planes orthogonal to the long axis, the other set consists of planes through the long axis with evenly spaced angular

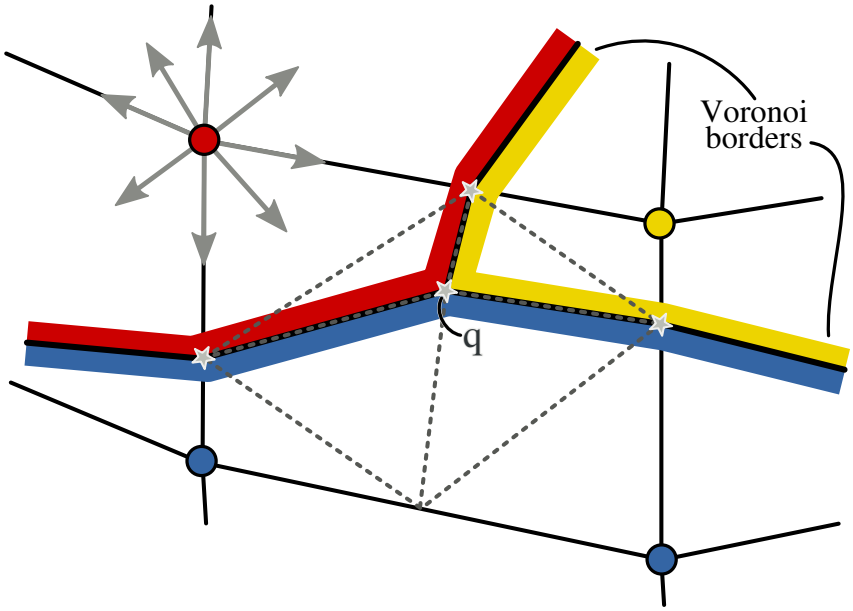


Figure 3 The distances to the closest coronary artery are propagated to all neighbors of a vertex along the edges and diagonals of each quadrilateral it is part of. Each quadrilateral can be divided in up to four territories. The gray lines indicate the triangularization of this particular case. Note that our approach, opposed to the traditional marching squares algorithm, generates a vertex in the middle of the quadrilateral.

offsets to the short axis. The points at the intersections of a plane from each set and the unstructured mesh form control points for a set of interpolating Catmull-Rom spline patches. We generate the final rectangular mesh by tessellating each patch at the desired accuracy.

3.2 Projection of the Coronary Arteries

Once we have a proper mesh of the epicardium, we project the coronary arteries onto this mesh. We compute a discrete representation of the coronary artery tree by generating a set of points spaced approximately one millimeter apart along each coronary artery. For each of the points along each coronary artery, we compute the closest vertex of the mesh. The index of the artery and the Euclidean distance to that artery is stored at that vertex.

3.3 Computation of the Voronoi Diagram

As a first step towards computing the Voronoi diagram of the arteries on the epicardial mesh, we compute the closest coronary artery and the distance to that artery for all vertices of the mesh by propagating the information in the vertices encountered during the projection of the arteries in the previous step. Starting with an active set consisting of only those vertices, we propagate the distances to all neighboring vertices using an 8-neighborhood approach. In other words, each vertex has eight neighbors to which distances are propagated, except at the edges of the mesh (see Figure 3). Distances and indices to arteries are only updated if the distance is shorter than the one already present in the vertex, if any. Once the algorithm terminates, each vertex contains an index of and distance to its closest coronary artery.

Here “closest” refers to the distance along the epicardium to a projection of a coronary artery. We believe this method, although still very simple, approaches reality better than using Euclidean distances in space. Since we use a discrete mesh of the epicardium, the distance is the approximate

geodesic distance along this mesh. In order to obtain a good approximation of the true geodesic distance, we construct a sufficiently fine-grained mesh. In our experiments we construct a mesh with 80 contours between the apex and the base of the left ventricle and contours divided into 128 vertices. Experiments with using finer grained meshes indicate that the maximum error due to interpolation between vertices is less than one millimeter.

The Voronoi diagram is given by the lines passing through edges whose vertices have different closest coronary arteries. We extract these lines using an approach based on marching squares, as all faces of our mesh are quadrilaterals. The difference with traditional marching squares is that each quadrilateral can be divided in up to four parts. Figure 3 shows a case where the four vertices of a quadrilateral belong to three different coronary territories, dividing the quadrilateral into three parts. We have generalized the division of the quadrilateral to cover the cases where it needs to be divided into three or four parts. The intersection points on the edges of the quadrilateral are first computed. To improve the quality of the dividing lines, we use linear interpolation to compute more exact intersection positions. We then compute the average of these points, giving a point q inside the quadrilateral. For edges that have no intersection point, we compute the point on the middle of those edges. We can then divide the quadrilateral into four regions where each region is given by one corner point, the two intersection or middle points of the edges connected to the corner and point q. Note that each of these four regions is always convex, as point q is inside or on the rhombus spanned by the four intersection or middle points. The four regions can then be triangulated independently using at most two triangles, as is shown in Figure 3 by the gray dashed lines. The number of triangles used for parts consisting of multiple regions can be reduced. The blue part in Figure 3, consisting of two regions of the quadrilateral, can for example also be triangulated using three triangles. The edges of the Voronoi diagram are given by the edges of the triangles that belong to different parts.

3.4 Projection onto a Bull's Eye Plot

Once we have computed a complete Voronoi diagram on the epicardial surface, we project it onto a two-dimensional bull's eye plot. The projection method is based on a parameterization of the left ventricle, which is illustrated in Figure 4. Each point in the left ventricular myocardium can be specified using a 3-tuple (ϕ, h, r). In this tuple ϕ represents the angle with the short axis in a plane orthogonal to the long axis, h the distance to the apex along the long axis and r the distance to the long axis. The projection of the Voronoi edges is constructed by computing the parameters of all vertices and directly interpreting (ϕ, h) as polar coordinates. The coronary arteries are projected on the bull's eye plot using the same approach.

4 Patient-Specific Coronary Territories

In the 17-segment model there is no variation in shape among patients of the three coronary territories. Although this eases interpretation and comparison, it does not give any details about the patient-specific coronary anatomy.

Using the method of computing patient-specific coronary territories described above, we can improve on the 17-segment model in two different ways. The first approach is to alter the 17-segment model by fitting the model's edges to the patient-specific coronary territories. The second approach is to compute the coronary territories without taking any model into account.

4.1 A Patient-Specific 17-Segment Model

For the first approach to work, the morphology of the patient-specific coronary territories should match that of the 17-segment model. The first 16 segments of the 17-segment model divide the

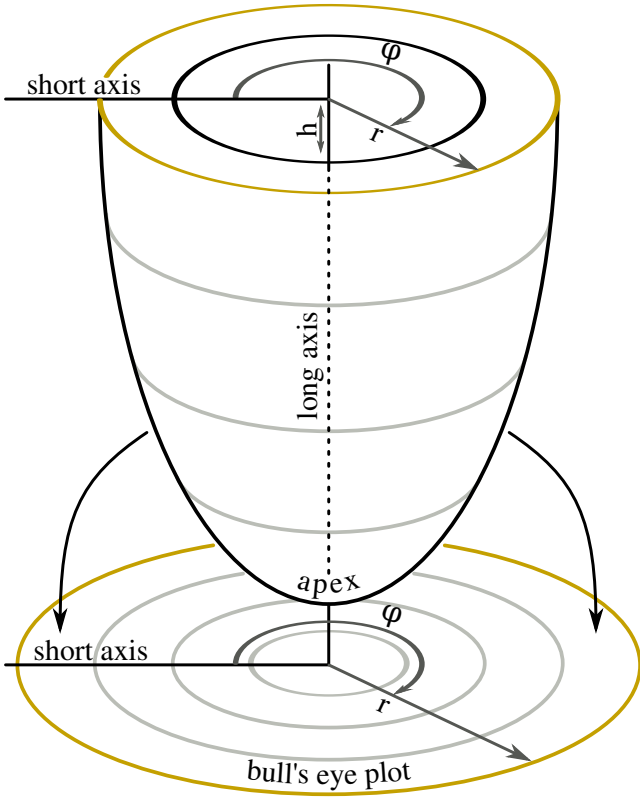


Figure 4 Parameterization of the left ventricle and its mapping to a bull's eye plot.

left ventricle into three parts: a basal part, a mid-cavity part and an apical part. Each of these parts correspond to one of the rings in the model shown in Figure 1. Each of the primary coronary arteries supplies part of each of those three parts.

The segmentation of the coronary arteries in a whole heart MRI scan is a difficult task. As the coronary arteries become thinner towards the end, segmentation of the complete branch is rarely possible due to the resolution of current whole heart MRI scans. This often results in incomplete segmentations of the coronary arteries, which in turn causes the morphology of the coronary territories based on these segmentations to rarely match that of the 17-segment model. We can force a correspondence by artificially completing the segmentation of the coronary arteries by connecting the endpoint of each artery to the apex. This approach is a compromise between a solely segmentation-driven approach, like the approach discussed in Section 4.2, and a solely model-driven approach, like the 17-segment model.

When adapting the 17-segment model, the borders of the coronary territories in the model are fit to the corresponding borders in the Voronoi diagram. The edges of segments indicating the center of each territory are mapped to lines equidistant to the borders of that coronary territory. Both these mappings require that each coronary territory has two borders, i.e. it has a "left" and a "right" side. This means that each coronary territory should run from the apex to the base and should be connected, just as is the case in the 17-segment model.

The basal, mid-cavity and apical segments of the 17-segment model are equal thirds of the heart along the long axis. This means that any differences between the original and an adapted 17-segment model are strictly *angular* differences. The relation of the distance to the apex along the long axis is also preserved in the 17-segment model; adapting the model thus does not affect h. We conclude

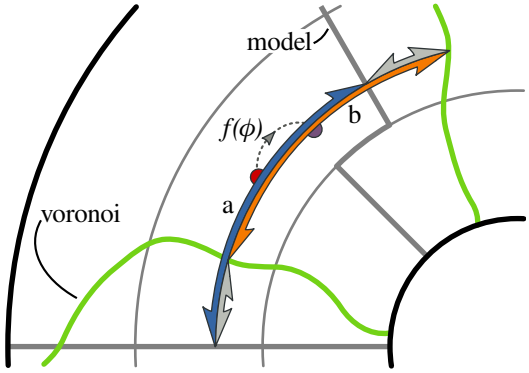


Figure 5 Function f transforms the query point (red dot) from the 17-segment model to the adapted model. The angular differences (gray arcs) are used to transform the blue arc A into the orange arc B.

that a function f exists such that for any point (ϕ, h) in a bull's eye plot of the 17-segment model, $(f(\phi), h)$ gives the corresponding point in a bull's eye plot of an adapted 17-segment model.

Figure 5 gives an overview of the implementation of function f. Given a tuple (ϕ, r) in a coronary territory (the red dot), we compute the two angular differences between the borders of the coronary territory according to the 17-segment model and the Voronoi diagram respectively (the gray arcs), at distance r from the apex along the long axis. We use linear interpolation between these two angular differences to find the angular difference for the specified value of ϕ (the arc marked f). This approach essentially transforms the blue arc A into the orange arc B using an angular scaling and translation. Note that function f is in fact defined for every point inside the myocardium, even through it is only applied to the edges of the segments for computing the adapted 17-segment model. In Section 5.2 we use this property to adapt volume data to the 17-segment model instead, using the inverse of f.

Figure 6 shows a comparison between a standard 17-segment model (Figure 6a) and the same model adapted to the patient-specific coronary anatomy (Figure 6b). The coronary arteries are also projected on top of the bull's eye plots. Note that the Voronoi diagram is computed using geodesic distances in three-dimensional space, so the borders of the coronary territories generally are not equidistant to the coronary arteries between them. Also, part of the RCA wraps around the right ventricle. Since the distance to the long axis is lost in the bull's eye plot, this is no longer visible in the projection of the RCA. For this particular patient, the divergence from the model is not particularly big. This is not too uncommon, as the 17-segment model is based on an average of the population.

4.2 Unconstrained Coronary Territories

The previously discussed approach artificially completes the segmentation of the coronary arteries by connecting the end points to the apex. We can also omit this process and use the coronary anatomy directly. However, the resulting division of the myocardium into coronary territories does not necessarily correspond to the 17-segment model. Our experiments show that there is no complete correspondence for most patients. In most of the cases this is due to varying way the apex is supplied, as is confirmed by earlier studies [9,10]. Also the fact that the LAD becomes too thin to be successfully segmented well before it has reached the apex often plays a role.

Figure 6c shows the coronary territories computed without artificially completing the coronary artery segmentation. In this particular case, the divergence from the 17-segment model is rather minor and correspondence is only lost in the apical segment of the model, corresponding to the inner ring. Since no constraints are set, the divergence may be arbitrarily large. In fact, the territories may even

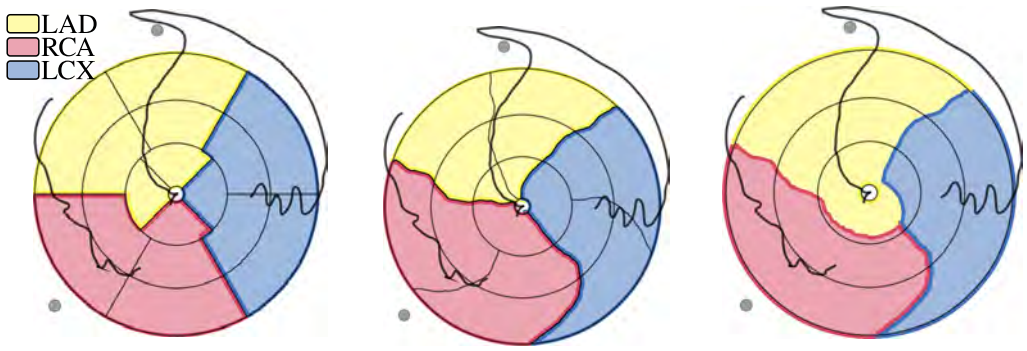


Figure 6 A comparison of (a) the 17-segment model, (b) the 17-segment model adapted using the patient-specific coronary anatomy and (c) unconstrained coronary territories. The three coronary arteries are projected as black lines. All three bull's eye plots correspond to the same patient.

be disjoint, although this rarely occurs in practice. While this approach imposes less constraints on the coronary territories, it is also more sensitive to incomplete segmentations of the coronary arteries.

5 Medical Expert Evaluation

We have performed an informal evaluation of our approach by selecting five patients who underwent a whole-heart cardiac MRI scan and generating four bull's eye plots for each patient. The first bull's eye plot contained only a projection of the three primary coronary arteries. The second, third and fourth bull's eye plots also contained the original 17-segment model, an adapted 17-segment model and unconstrained coronary territories, respectively. Figure 6 gives the latter three bull's eye plots for patient 2 used in this experiment. For patient 4 an adapted 17-segment model could not be generated. This issue is further discussed in Section 6.

An experienced cardiologist first manually drew the coronary territories on the first bull's eye plot for each patient. The next task was to rate the correspondence between the projected coronary arteries and the coronary territories as produced by each of the three approaches on a scale from one (very bad) to five (very good). The results of this evaluation are listed in Table 1.

Case	Method 1	Method 2	Method 3
Patient 1	5	5	5
Patient 2	4	4	4
Patient 3	2	2	4/5
Patient 4	1	N/A	4/5
Patient 5	1	2	4/5

Table 1 Results of our evaluation experiment. Method 1 corresponds to the original 17-segment model, method 2 corresponds to the adapted 17-segment model and method 3 corresponds to using unconstrained coronary territories. The scale ranges from one (very bad) to five (very good).

Finally our expert provided an overall rating for each approach of determining coronary artery territories. The manual approach was given a score of 4, both the original and our adapted 17-model were rated 3 and using unconstrained coronary territories was rated 4.

Our expert expressed that it was difficult to get a feeling for the concept of forcing the coronary territories to correspond to the 17-segment model. The continuous bull's eye plot projection we used also created some confusion. It was not clear whether strong variations within one ring, such as in the lower right part of Figure 6c, corresponded to epicardial and endocardial territories or whether the diagram should be interpreted as having infinitely many rings. Finally, a suggestion was made to restrict the segmentation of the LAD in the inferior wall, as current segmentations often led to an overestimation of the LAD territory. This is also the reason why we were unable to compute constrained coronary territories for patient 4.

In the manually drawn territories the inner ring was entirely allocated to the LAD in all patients except the first. Since this does not correspond to the original 17-segment model, this was only reflected by the unconstrained coronary territories. This is clearly visible in the scores given in Table 1 and is also the primary reason this method is preferred. In summary, our expert was positive about our work, especially the unconstrained coronary territories. Our evaluation indicates that this approach can compete with a manual approach.

5.1 Application to CT Data

We have also applied our technique to a whole heart CT scan. We applied the same segmentation techniques as for the MRI scans. Due to the better spatial resolution of CT, we were able to extract a more detailed coronary artery tree. Figure 7 shows the unconstrained coronary territories of this dataset. We use hue to encode to which primary coronary artery a territory belongs to and use lightness to distinguish territories from subbranches. This visualizes the hierarchy in the coronary artery tree. It also demonstrates that our approach is capable of generating a large number of territories. Due to the more fine-grained segmentation of the coronary artery tree, there is little difference when the end points of the coronary arteries are connected to the apex. For the same reason as before, forcing correspondence to the 17-segment model was not feasible for this patient.

5.2 Application to Viability Data

The primary advantage of patient-specific information on the coronary territories is that it allows to establish a more accurate relation between a functional deficit and the coronary arteries. This involves including other types of data, such as cine, late enhancement and perfusion data. Here we give an example of how our approach can be combined with viability information from a late enhancement scan. A late enhancement scan shows areas where a contrast agent has accumulated, typically indicating dead tissue called scar. We demonstrate that the patient-specific coronary territories can lead to a better understanding of which coronary arteries are related to areas of scar.

Figure 8 shows two bull's eye plots that each depict both the coronary territories and viability data. The darker areas in the bull's eye plots correspond to areas of scar. The top bull's eye plot shows the coronary territories as an adapted 17-segment model. In the bottom bull's eye plot, we adapted the volume data to fit the original 17-segment model using the inverse of function f discussed in Section 4.1. We implemented this using on-the-fly resampling of the volume data. In other words, function f^{-1} is evaluated for every pixel during the rendering of the bull's eye plot. In this case the scar on the top side of the bull's eye plot is stretched to fill the entire segment.

On the right of Figure 8 a three-dimensional view is shown to relate the scar to the three-dimensional coronary and cardiac anatomy. The coronary territories are projected on the epicardial surface, using the adapted 17-segment model approach. To establish a strong correspondence between the bull's eye plot and the three-dimensional view, we show a cursor on both views, indicated by the red arrows in Figure 8. To facilitate easy navigation, the viewpoint can be controlled by the position

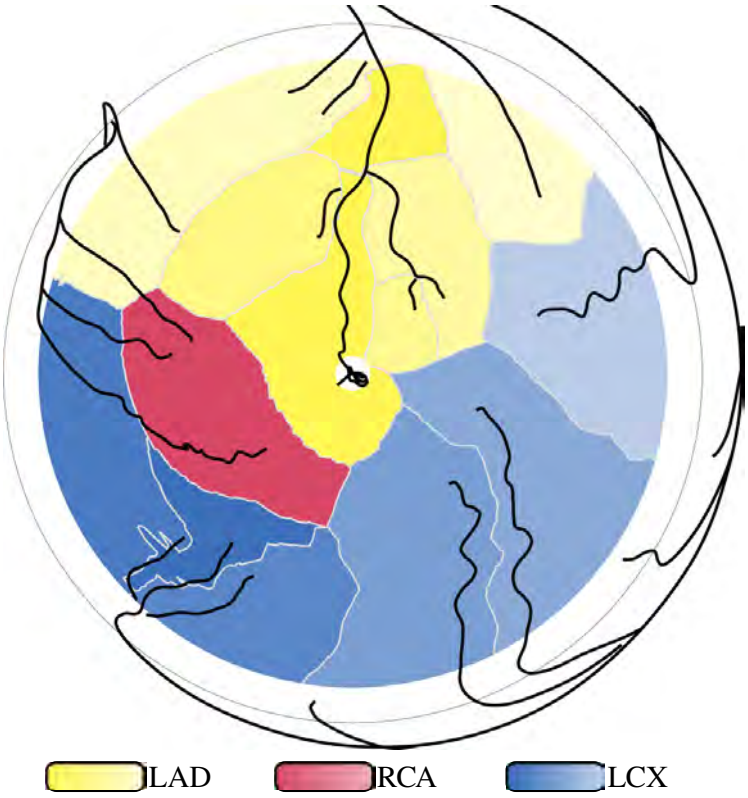


Figure 7 Unconstrained coronary territories computed from a detailed coronary artery tree extracted from a CT scan. The hue of each territory encodes its primary coronary artery, while the lightness is used to distinguish territories from subbranches.

of the cursor. The user can then explore the three-dimensional view by moving the cursor on the bull's eye plot.

6 Discussion

For some patients, the coronary anatomy is too different from what the 17-segment model assumes. As was verified by Pereztol-Valdés *et al.* [10] and Ortiz-Pérez *et al.* [9], especially in the apical region there is a lot of variance among patients. Figure 9a shows the unconstrained coronary territories of one of our patients. The territory of the LAD is relatively large because the LAD covers both the anterior and inferior left ventricular walls in the apical area. Moreover, the part of the RCA that was visible in the whole heart MRI scan did not extend toward the apical region.

For this patient, our approach of adapting the 17-segment model fails since we cannot force a correspondence due to the dominant LAD. If an adapted 17-segment model is desired, the segmentation of the LAD could be restricted in the apical area. Analyzing the unconstrained coronary territories as shown in Figure 9a is in fact especially useful for patients with a coronary anatomy that considerably diverges from the average.

There are also patients that correspond very well to the 17-segment model. Figure 9b shows a comparison of an adapted 17-segment model and the standard 17-segment model in red and black lines, respectively. For this particular patient, the edges of the coronary territories map fairly well to what the 17-segment model predicts. It should be noted however that the model of each part of the myocardium being exclusively perfused by a single artery is especially inaccurate near these borders

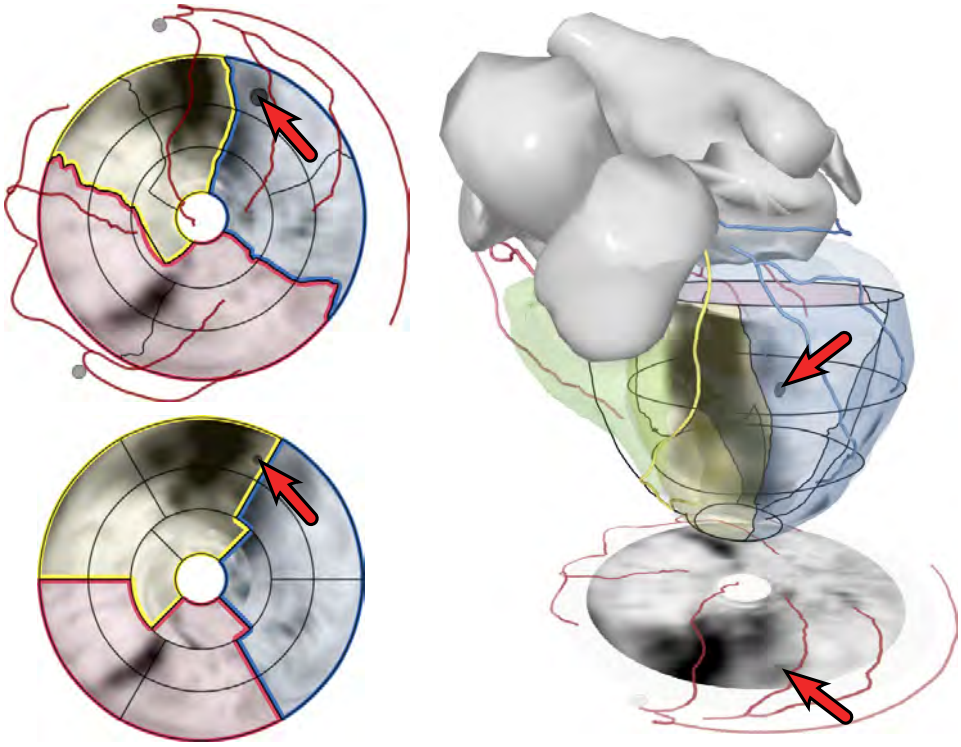
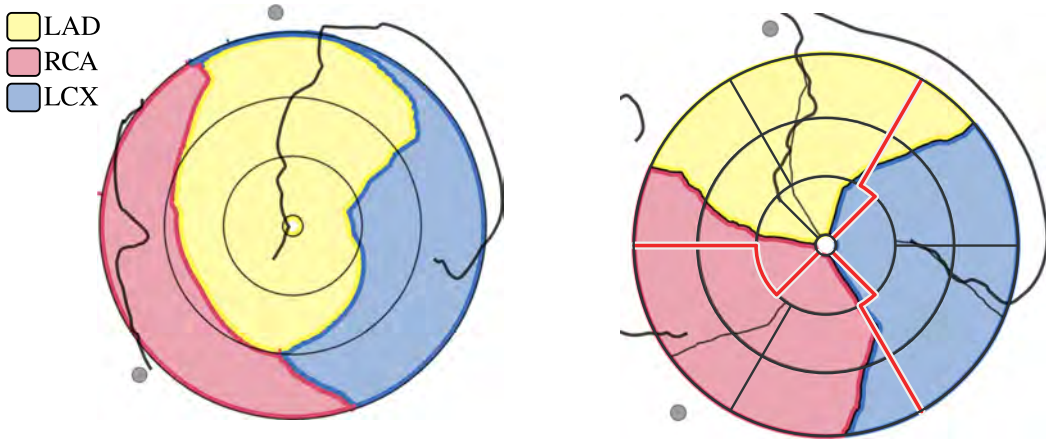


Figure 8 Combined visualization of coronary territories and viability information. The darker areas in the bull's eye plots and the three-dimensional view correspond to areas of scar. A cursor (red arrows) shows corresponding points on both bull's eye plots and the three-dimensional view. The 17-segment model is shown in the three-dimensional view as a set of black lines around the epicardium.



(a) (b)

Figure 9 Varying correspondence to the 17-segment model: (a) a case where adapting the model fails, (b) a case with relatively little difference to the model.

of the coronary territories. Another aspect that is visible in Figure 9b is that due to the circular nature of the bull's eye plot, differences near the apex result in smaller visual differences in the diagram.

We have not included segment number 17, corresponding to the apex, in our experiments. This segment forms a special case compared to the other 16 segments. It it is connected to all other territories and only has one border. A special approach is required to assign it to a coronary territory, since there is no clear mapping between this segment and the Voronoi diagram on the epicardial mesh. Thus when adapting the 17-segment model, it would only have to be decided to which coronary territory this segment should be assigned, but no edges need to be adjusted.

In our experiments we used meshes consisting of approximately 10^4 quadrilaterals. The entire preprocessing phase, including the mesh computation, coronary artery projection, Voronoi diagram computation and computing the adapted 17-segment models typically takes less than one second on a modern workstation (Intel Xeon 3 Ghz, 2 GiB RAM, NVIDIA GeForce 8800 GTX).

7 Conclusion

We have introduced two approaches for establishing a relation between the patient-specific coronary anatomy and the myocardium. In the first approach we adapt the borders of the segments of the 17-segment model as defined by the American Heart Association. In the second approach we compute the coronary territories without taking any model into account. Both approaches are based on a Voronoi diagram computation of the coronary arteries projected onto a quadrilateral mesh of the epicardium.

Both approaches provide detailed insight in the patient-specific coronary anatomy. Adapting the 17-segment model forces a correspondence between the coronary territories and the 17-segment model by connecting the end points of the primary coronary artery branches to the apex. This approach forms a compromise between using a model-based approach and a segmentation-based approach. The second approach does not pose any constraints and thus allows for arbitrarily shaped coronary territories. It is therefore also more sensitive to incomplete segmentations of the coronary arteries. While the uniformity appearance of the 17-segment model is lost, the additional patient-specific information on the coronary territories allows for a better understanding on the relation between the coronary arteries and the myocardium.

We have presented an application of using patient-specific coronary territories with visualizing viability information from a late enhancement scan. Instead of altering the 17-segment model to correspond to the coronary anatomy, we show that it is also possible to fit the underlying volume data to the original 17-segment model. This approach creates uniformly looking bull's eye plots, regardless of the coronary anatomy of the patient.

Feedback obtained through an evaluation experiment with a cardiologist gave a clear signal that our work has clinical relevance. Results indicated that there is a preference towards using unconstrained coronary territories. The latter method produced territories that best matched those indicated manually by our cardiologist and are considered to have a greater correspondence than those that the original 17-segment model suggest.

Acknowledgements

This work was performed in the scope of the COMRADE project funded by Philips Healthcare, Best, The Netherlands. The datasets were provided by Hyogo BHC at Himeji and the Tokyo Metro Police Hospital.

References

1 American Society of Nuclear Cardiology. Imaging guidelines for nuclear cardiology procedures. *J Nuclear Cardiology*, 6:G47–G48, 1999.

2 P. Beliveau, R.M. Setser, F. Cheriet, R.D. White, and T. O'Donnell. Computation of coronary perfusion territories from CT angiography. *Computers in Cardiology*, 34:753–756, 2007.

3 Pascale Beliveau, Randolph Setser, Farida Cheriet, and Thomas O'Donnell. Patient-specific coronary territory maps. *Proc. SPIE*, 6511:65111J, 2007.

4 Manuel D. Cerqueira, Neil J. Weissman, Vasken Dilsizian, Alice K. Jacobs, Sanjiv Kaul, Warren K. Laskey, Dudley J. Pennell, John A. Rumberger, Thomas Ryan, and Mario S. Verani. Standardized myocardial segmentation and nomenclature for tomographic imaging of the heart. *Circulation*, 105:539–542, 2002.

5 Olivier Ecabert, Jochen Peters, and Jörgen Weese. Modeling shape variability for full heart segmentation in cardiac computed-tomography images. In *Proc. SPIE*, volume 6144, pages 1199–1210, 2006.

6 Rufold Karch, Friederike Neumann, Martin Neumann, Paul Szawlowski, and Wolfgang Schreiner. Voronoi polyhedra analysis of optimized arterial tree models. *Annals of Biomedical Engineering*, 31:548–563, 2003.

7 Cristian Lorenz, Steffen Renisch, Thorsten Schlathoelter, and Thomas Buelow. Simultaneous segmentation and tree reconstruction of the coronary arteries in MSCT images. In *Proc. SPIE*, volume 5031, pages 167–177, 2003.

8 S. Oeltze, A. Kuß, F. Grothues, A. Hennemuth, and B. Preim. Integrated visualization of morphologic and perfusion data for the analysis of coronary artery disease. In *Proc. EuroVis*, pages 131–138, 2006.

9 José T. Ortiz-Pérez, José Rodríguez, Sheridan N. Meyers, Daniel C. Lee, Charles Davidson, and Edwin Wu. Correspondence between the 17-segment model and coronary arterial anatomy using contrast-enhanced cardiac magnetic resonance imaging. *JACC: Cardiovascular Imaging*, 1(3):282–293, 2008.

10 Osvaldo Pereztol-Valdes, Jaume Candell-Riera, Cesar Santana-Boado, Juan Angel, Santiago Aguade-Bruix, Joan Castell-Conesa, Ernest V. Garcia, and Jordi Soler-Soler. Correspondence between left ventricular 17 myocardial segments and coronary arteries. *European Heart Journal*, 26:2637–2643, 2005.

11 O. M. Weber, A. J. Martin, and C. B. Higgins. Whole-Heart Steady-State Free Precession Coronary Artery Magnetic Resonance Angiography. *Magnetic Resonance in Medicine*, 50(6):1223–8, 2003.

Modeling and Visualization of Cardiovascular Systems

Thomas Wischgoll[1]

1 **Computer Science and Engineering Department, Wright State University**
thomas.wischgoll@wright.edu

Abstract

Modeling complex organs, such as the human heart, requires a detailed understanding of the geometric and mechanical properties of that organ. Similarly, the model is only as accurate as the precision of the underlying properties allow. Hence, it is of great importance that accurate measurements of the geometric configuration are available. This paper describes the different steps that are necessary for creating and visualizing such a vascular model, ranging from determining a basic geometric model, gathering statistical data necessary to extend an existing model up to the visualization of the resulting large-scale vascular models.

1998 ACM Subject Classification I.3.8 Applications, J.3 Life and Medical Sciences

Keywords and phrases Volumetric Data, Curve-skeleton, Cardiovascular Structure

Digital Object Identifier 10.4230/DFU.SciViz.2010.210

1 Introduction

In order to precisely model a complex organ, such as the human heart, the organ's mechanical and physiological properties need to be fully understood. This includes the analysis of, for example, the geometric configuration of the myocardium and its supporting vasculature as well as the structural configuration of the vessels themselves. Therefore, this paper describes methodologies that help analyze such properties and provides some answers to the visual analytics challenges this ongoing research is facing.

In order to extract morphometric data from a volumetric data set that resembles a scan of a coronary system with the arteries highlighted using some form of contrast agent, the center lines of the individual vessel segments need to be identified resulting in a curve-skeleton describing the medial axis of the vessels. This then allows the software to compute the vessel radii as the distance between the center lines and the vessel boundary. Similarly, the length of a vessel segment can be computed as the length of the center lines, as well as bifurcation angles as the different angles formed by the center lines at a bifurcation. Obviously, it is quite important that both the boundary and the center lines are determined as accurately as possible. The method used in this paper computes both at a sub-voxel level to achieve a very high precision.

Curve-skeletons represent the very basic features of an object. They describe a thinned version of the object represented as some type of stick model resulting in the center-lines of the object. Therefore, the use of curve-skeletons can prove useful for applications, such as animation [46] or flight planning for virtual colonoscopy [19]. Similarly, accurate curve-skeleton methods can be used for extracting quantitative measurements from computed tomography (CT) scanned vascular structures. Here, the curve-skeleton describes the center lines of the vessels. These can then be used to measure vessel radius, vessel lengths, and angles between vessels within the volumetric data set retrieved by using the CT scanner. In order to derive these measurements from the volumetric data set, an accurate extraction method for curve-skeletons is desirable. For example, thinning-based techniques that work

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 210–226

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

in the voxel space of the volumetric data set tend to generate jagged lines which are in no way suitable for determining angles between vessels. Similarly, inaccuracies can occur when computing the radii of the vessels. Hence, an approach that only uses the volumetric data set in order to identify the boundary surface of the contained object is more promising.

The algorithm used in this paper is exactly of this type. It is capable of extracting the boundary surface of an object that is defined by a volumetric data set at sub-voxel level. For this, it determines the location of the maximal gradient within the volumetric data set similar to Canny's [8] maxima-suppression technique but extended to three dimensions. Since the algorithm only relies on points extracted from the volumetric data set but not on its underlying structured grid, it can also be applied to objects defined by a point set without any restrictions.

Techniques used for computing the topological graph of a vector field are applied to determine the curve-skeleton. First, for all points on the object's boundary vectors are computed that are orthogonal to the boundary surface. There are different options for computing these vectors. They either can be derived by determining the normal vector of a plane that is defined by a least-square fit of the point and its neighbors. Or – in case of the object being defined by a volumetric data set – the image gradients determined in the previous step can be used. In both cases, the normal vectors can be determined in such a way that they are facing inwards with respect to the object. The entire vector field can then be determined by computing a tetrahedrization of the entire point cloud and then linearly interpolating within the tetrahedra. In order to ensure that only the curve-skeleton inside the object is extracted, all tetrahedra that are located outside the object are removed based on the normal vectors.

A topological analysis of the vector field within the faces of every tetrahedron yields points on the curve-skeleton. By following the topology of the tetrahedrization, points on the curve-skeleton within neighboring tetrahedra can be connected resulting in the entire curve-skeleton. Based on this curve-skeleton, vessel radii are computed at a very high accuracy. Similarly, bifurcation angles can be computed and statistically analyzed. This information, i.e. statistical information about the vessel lengths, vessel radii, and bifurcation angles for vessels of different sizes, can then be used to grow additional vessels onto an existing model resulting in a large-scale model of the vasculature that includes vessels down to the capillary level.

Rendering such a large-scale model is quite challenging for currently available computing hardware since commodity graphics cards are presently not able to display this amount of information interactively. For the complete coronary arterial model, traditional rendering algorithms require the total of 6 GB of geometric information to be transferred from main memory to the graphics card, which presents a limit for interactive rendering. Furthermore, most desktop computers are not capable of handling this amount of data due to insufficient main memory. Hence, the size of such a large-scale anatomical model is prohibitive for rendering on desktop computers without employing out-of-core techniques or more sophisticated rendering algorithms. Occlusion culling techniques are usually capable of achieving better performance. However, when applied to tree-shaped data sets only little occlusion occurs which in turn requires the removal of partly visible areas of the data set in order to achieve an increase in rendering performance [49]. In addition, medical personnel and researchers tend to prefer to see the entire data set without anything being removed in fear of missing essential information. Hence, the visualization methods in this paper refrained from applying any type of geometry reduction methods but improved on the rendering method itself instead.

The objective of this paper is to outline methods that allow for the analysis, modeling, and visualization of cardiovascular structures. This includes methodologies that are capable of extracting quantitative morphometric measurements from scans of cardiovascular data sets. These statistical data can then be used to generate a large-scale vascular model with vessels from the most proximal level down to the capillaries. The enormous amount of vessel segments included in such a model requires novel visualization methods in order to achieve interactive rendering of these data sets. The

proposed approach computes the necessary geometry that is required for visualizing the data set on-the-fly and utilizes all computational resources available on today's desktop computers, i.e. all cores of the CPU and the GPU at the same time, thereby achieving maximal performance of the visualization algorithm. The rendering algorithm is especially optimized for rendering large-scale tree-shaped data sets and the computation of all necessary geometry in parallel on the CPU assisted by the GPU. The techniques described in this paper can easily be applied to data extracted from any type of tree-like structure.

The following section discusses work related to the topics described in this paper. A description of the algorithms can be found in section 3. Section 5 discusses the performance of the algorithm applied to the large-scale vascular data set, followed by conclusions and future work.

2 Related Work

Several approaches for extracting curve-skeletons or medial axes can be found in the literature. A very good overview of available techniques can be found in the paper by Cornea et al. [9].

Some methods start with all voxels of a volumetric data set and use a thinning technique to shrink down the object to a single line. Directional thinning approaches use a specific order in which voxels are removed. For example, directions, such as up or down, are used to define this order and conditions are used to identify endpoints [2, 7, 17, 22, 24, 31, 32, 37, 45]. Since these methods are sensitive with respect to the order in which the voxels are removed the resulting curve-skeleton may not be perfectly centered. Non-directional methods [5, 42] or fully parallel approaches [12, 25, 27] do not suffer from this disadvantage. Ideally, the topology of the object should be observed. Such an approach was proposed by Lobregt et al. [23] which is the basic technique used in commercial software systems, such as Analyze^{TM} developed by the Mayo Clinic. The disadvantage of this approach is that it tends to produce jagged lines which do not allow accurate measurements of angles between parts of the object, such as individual vessels of a vascular structure. Other approaches [43] classify the voxels in different groups, such as edge, inner, curve, or junction and re-classify after removal of a voxel. A similar algorithm is proposed by Palagyi et al. [31]. The disadvantage of thinning algorithms is that they can only be applied to volumetric data sets due to the nature of these algorithms.

To avoid this disadvantage, other approaches deploy the distance transform [16] or distance field in order to obtain a curve-skeleton. For each point inside the object, the smallest distance to the boundary surface is determined. For this, the Euclidean metric or the <3,4,5> metric [4] can be used. Also, fast marching methods [39, 44] can be deployed to compute the distance field. Voxels representing the center lines of the object are identified by finding ridges in the distance field. The resulting candidates must then be pruned first. The resulting values are then connected using a path connection or minimum span tree algorithm [41, 47, 52]. Methods used to identify points on the ridges include distance thinning [10, 14, 15, 34], divergence computing [6], gradient searching [3], thresholding the bisector angle [26], geodesic front propagation [33], or shrinking the surface along the gradient of the distance field [38]. The distance field can also be combined with a distance-from-source field to compute a skeleton [53]. Based on an anisotropic diffusion applied to the image gradients, Yu et al [51] extract skeletons from 2D images.

Techniques based on Voronoi diagrams [1, 11] define a medial axis using the Voronoi points. Since this approach usually does not result in a single line but rather a surface shaped object, the points need to be clustered and connected in order to obtain a curve-skeleton. Voronoi-based methods can be applied to volumetric data sets as well as point sets. Due to the fact that clustering of the resulting points is required, these approaches can lack some accuracy.

The center lines in combination with the vessel radii computed at the center points allow for a geometric reconstruction of the vasculature. Various techniques for visualizing vascular structures

can be found in the literature. Hahn et al. [18] employ geometrical primitives, such as truncated cones, to visualize vessels inside the human liver. Masutani et al. [28] used cylinders aligned to the vessel skeleton to visualize the vasculature. Different radii at branchings resulted in discontinuities when using this method. Felkel et al. [13] reconstructed liver vessels from center line and radius information to supply an augmented reality tool for surgery. Puig et al. [35] developed a system for exploring cerebral blood vessels using a symbolic model with a focus on geometric continuity and on realistic shading. Oeltze et al. [29, 30] use convolution surfaces to obtain a smoother representation of blood vessels extracted from CT or MR data. Ritter et al. [36] extended on illustrative rendering techniques to accentuate spatial depth by use of GPU-accelerated shadow-like depth indicators. The method was applied to vascular structures to better distinguish vessels in the front from vessels further in the back. Stoll et al. [40] introduced stylized primitives which utilize the GPU to render cylindrical objects using a single quadrilateral. The approach was used for fast rendering of streamlines in vector field data sets.

3 Methodology

As indicated in the Physiome Project [20], modeling the human body requires its analysis at various levels ranging from the genomic over to the tissue level and organ level up to the human body itself. Similarly, modeling the human heart requires a detailed understanding of the vascular structure and the individual vessels of which the structure composed.

A previously developed software package extracts morphometric data from a volumetric image in several steps. Although a brief summary of the algorithm is given here, a detailed description can be found in the original publication [48]. The algorithm first segments the tubular objects within the volumetric image based on the image gradients. In order to get a more accurate representation of the boundary, the points resulting from the segmentation step are moved along the gradient direction in such a way that they are located at the maximal gradient. This provides a more precise and smoother representation of the boundary compared to using the original voxel locations. Then, a vector field is computed in such a way that all vectors are pointing inwards to the center of the tubular object. In the simplest case, the image gradients can be used at the boundary. Using a tri-linear interpolation, the vector field can be computed after a tetrahedrization of all the boundary points is determined. Finally, the points on the center lines are computed using a topological analysis of the vector field within the cross sectional area of the tubular objects and connected based on the topology of the tetrahedrization. This then results in a precise representation of the center lines of all tubular objects within the volumetric image. At the same time, the algorithm computes the radii of the tubular shapes along with the center lines as the distance between center line at boundary.

The algorithm has a proven accuracy with smaller error than most other methods. For the validation of the algorithm, vessel radii were computed for the main trunk of the arterial branches of a series of five porcine heart data sets as the distance between the center line and the vessel boundary and then compared to manual optical measurements. The agreement between the measurements is very good, with an error of $0.06mm$ (scan resolution was $0.6mm \times 0.6mm \times 1.0mm$), which underlines the accuracy of the center lines. For the three major branches (LAD, LCx, and RCA) of the five porcine hearts, the root mean square error between the two measurements is $0.16mm$ and the average deviation is $0.13mm$.

Based on these accurate morphpometric data, the vascular geometry can be reconstructed. The morphometric data extracted from the volumetric data set provides vessel segments identified as line strips defined by the center lines of the vessels with radii measurements at both ends of each line segment. Conic cylinders can then be computed that represent each vessel segment. Since two consecutive line segments may not have the exact same direction, the end caps of the conic cylinders

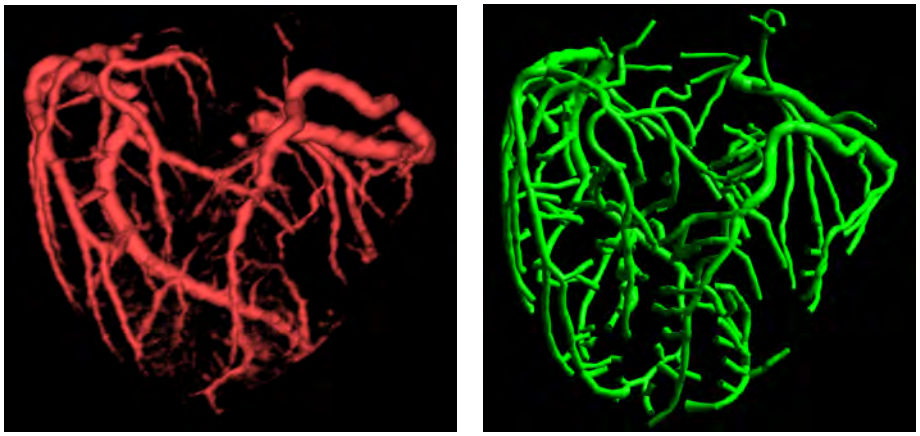


Figure 1 Vasculature of a pig heart: volume rendering (a) and geometric reconstruction (b) of the same heart.

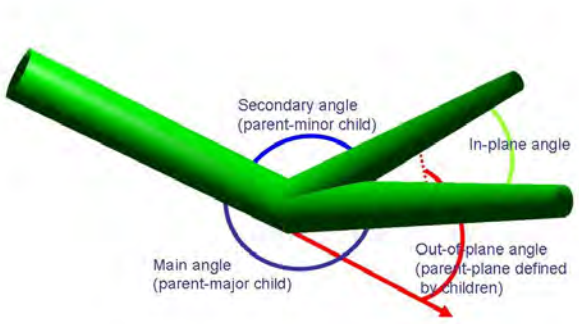


Figure 2 Definition of bifurcation angles.

are rotated in such a way that they describe half the angle between the line segments. This way, a smooth transition between segments can be created that avoids gaps at the transition. By computing such a conic cylinder for every vessel segment that was extracted from the volumetric data set, a geometric reconstruction of the vasculature is achieved. Figure 1 depicts such a reconstruction and compares it to a volume rendering of the exact same volumetric data set.

Using the above algorithm, the arterial center lines were determined for seven porcine hearts and for all vessels detected by the software. Since the accuracy for vessels smaller than the scan resolution is questionable as indicated by our previous study, vessels with a diameter of less than $1mm$ were not included in the analysis. Based on these center lines of the vessel segments, bifurcation angles were measured as defined in Figure 2. The two child segments define a plane. The angle between the two children within that plane describes the in-plane angle. The out-of-plane angle is formed between the parent segment and the plane defined by the two child segments. The main angle refers to the angle between the parent and the larger child segment while the secondary angle is determined by the angle between the parent and the smaller child segment.

Due to the fact that the first segment of a center line merely represents the connection of the vessel

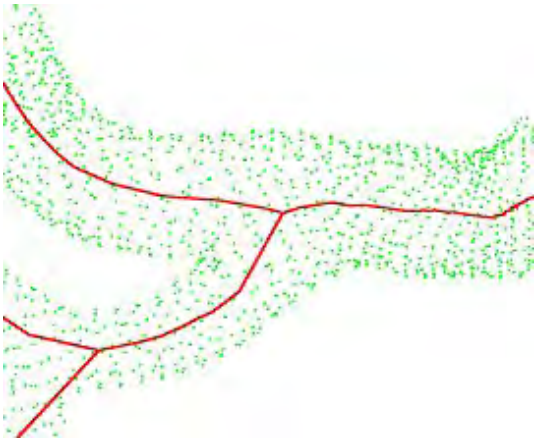


Figure 3 Sample bifurcation extracted from the volumetric data.

branches but not necessarily the direction of the daughter vessel due to possible curvature, more than just the first segment were included in the calculation of the bifurcation angle (Figure 3). The first center line segment usually only leads out of the main vessel formed by the parent and the larger child. Using this segment alone would result in erroneous bifurcation angles. Hence, the center line segments starting with the first segment after the bifurcation until the length of the daughter segment reaches three times the radius of the parent vessel were considered. The vectors representing the center line segments identified in such a way were then averaged and weighed by the length of each vector to determine a representative for the orientation of the daughter vessels. These were used for the computation of the bifurcation angles as described before.

For the statistical analysis, the bifurcations were grouped with respect to the order of the respective vessel segments. These groups were defined by the diameter of the vessels thereby associating a range of diameter values to a specific order. The relation between vessel diameters and orders are shown in Figure 4 for LAD, LCx, and RCA. The bifurcations were then classified based on the order of the parent and daughter vessels. Bifurcations between vessel orders ranging from 9 to 11 for LAD and RCA, and 9 and 10 for LCx were considered since vessels of that specific size can be extracted reliably from CT scanned imagery with a resolution of $1mm$. Average angles and standard deviation within each of these groups were computed for each of the major coronary arterial branches.

Figures 5 through 8 list the statistical summary of the secondary angles and out-of-plane angles in matrix form according to the order number of the parent vessel segment (vertical axis) and the daughter vessel segment (horizontal axis). Each entry lists the average angle within that group and the standard deviation. The tables confirm assumptions typically made for vascular structures. The smaller daughter vessel usually forks off at a relatively larger angle whereas the larger vessel continues the vessel segment in a rather straight fashion. The low out-of-plane angles confirm that bifurcations typically are relatively planar. Despite the important fact of confirming observations and assumptions often found in the literature, it is often difficult to analyze the values for specific bifurcations. A focus+context-oriented visualization can help in this case. By annotating a visualization of the vascular geometry with textual information representing the angular values, it is much easier to correlate the values with the geometry of the vascular structure making an analysis significantly less cumbersome. Figure 9 shows an example for such a visualization. The user can zoom in on specific bifurcations and check the attached angular values interactively.

The size of the vessel segments that can be extracted from the volumetric data depends on the resolution of the scanning device used. Typically, capillary vessels have a diameter of $6-7\mu m$. This is significantly smaller than the resolution of typical scanners. In order to derive a detailed model

		range	
RCA	Order	low	high
	9	552	955
	10	955.1	2,319
	11	2,319.1	3,606

		range	
LAD	Order	low	high
	9	554	986
	10	986.1	2,189
	11	2,189.1	3,830

		range	
LCx	Order	low	high
	9	649	1,782
	10	1782.1	2,940

Figure 4 Relation between order numbers and vessel diameter.

RCA	order 9	order 10	order 11
order 9	29.66±13.93	19.79±11.17	12.10±0.64
order 10		24.72±20.69	15.73±7.79
order 11			17.31±1.06

LAD	order 9	order 10	order 11
order 9	25.36±15.88	23.26±13.33	15.41±9.48
order 10		26.94±20.66	23.28±17.93
order 11			35.00±20.65

LCx	order 8	order 9	order 10
order 9		20.05±15.00	17.35±16.95
rder 10			24.74±30.33

Figure 5 Main bifurcation angles.

RCA	order 9	order 10	order 11
order 9	73.13±28.13	56.99±26.18	60.71±13.14
order 10		63.87±34.63	57.55±22.56
order 11			36.31±21.93

LAD	order 9	order 10	order 11
order 9	66.19±31.75	70.87±28.62	64.89±19.56
order 10		66.85±32.95	65.02±26.20
order 11			51.36±6.31

LCx	order 8	order 9	order 11
order 9		65.75±31.95	69.46±19.11
order 10			86.75±60.92

Figure 6 Secondary bifurcation angles.

RCA	order 9	order 10	order 11
order 9	89.46±19.12	65.18±27.89	70.35±11.46
order 10		68.80±35.15	62.30±20.31
order 11			38.79±43.47
LAD	order 9	order 10	order 11
order 9	66.37±31.68	76.55±32.03	69.25±16.86
order 10		68.57±33.86	75.29±31.87
order 11			25.49±26.03
LCx	order 8	order 9	order 10
order 9		71.99±32.03	64.39±26.57
order 10			74.08±48.50

Figure 7 In-plane bifurcation angles.

RCA	order 9	order 10	order 11
order 9	12.81±11.26	10.97±10.01	5.82±3.95
order 10		11.47±10.62	7.02±4.05
order 11			9.32±8.65
LAD	order 9	order 10	order 11
order 9	16.55±13.52	12.26±8.21	10.78±9.74
order 10		12.92±13.46	12.09±10.11
order 11			21.27±1.24
LCx	order 8	order 9	order 10
order 9		9.58±10.13	6.74±7.01
order 10			6.61±4.04

Figure 8 Out-of-plane bifurcation angles.

of the vasculature that includes vessels on the capillary level, the vasculature extracted from the volumetric data can be extended based on statistical properties of the vasculature. Recently, Kaimovitz et al. [21] developed a three-dimensional (3-D) geometric model of the entire coronary arterial tree (right coronary artery, RCA; left anterior descending artery, LAD; and left circumflex, LCx arterial tree). The model is purely based on morphometric data as extracted using the methodology as described previously, i.e. statistical information of vessel length, vessel diameter, and bifurcation angles. The model spans the entire coronary arterial tree down to the capillary vessels in a prolate spheroid model of the heart and encompasses about 11 million segments. The 3-D tree structure was reconstructed initially in rectangular slab geometry by means of global geometrical optimization using a parallel Simulated Annealing (SA) algorithm. The SA optimization was subject to a global boundary avoidance constraint and local constraints at bifurcations prescribed by previously measured data on branching asymmetry in the coronary arterial tree. Subsequently, the reconstructed tree was mapped onto the prolate spheroidal geometry of the heart. The transformation was made through least squares minimization of the deformation in segment lengths as well as their angular characteristics.

Due to the high number of vessel segments, the geometric data that needs to be generated for creating a high-quality visualization of this data set is quite substantial. When discretizing the end caps of the conic cylinders to represent each vessel segments with 8 vertices, the overall geometric data

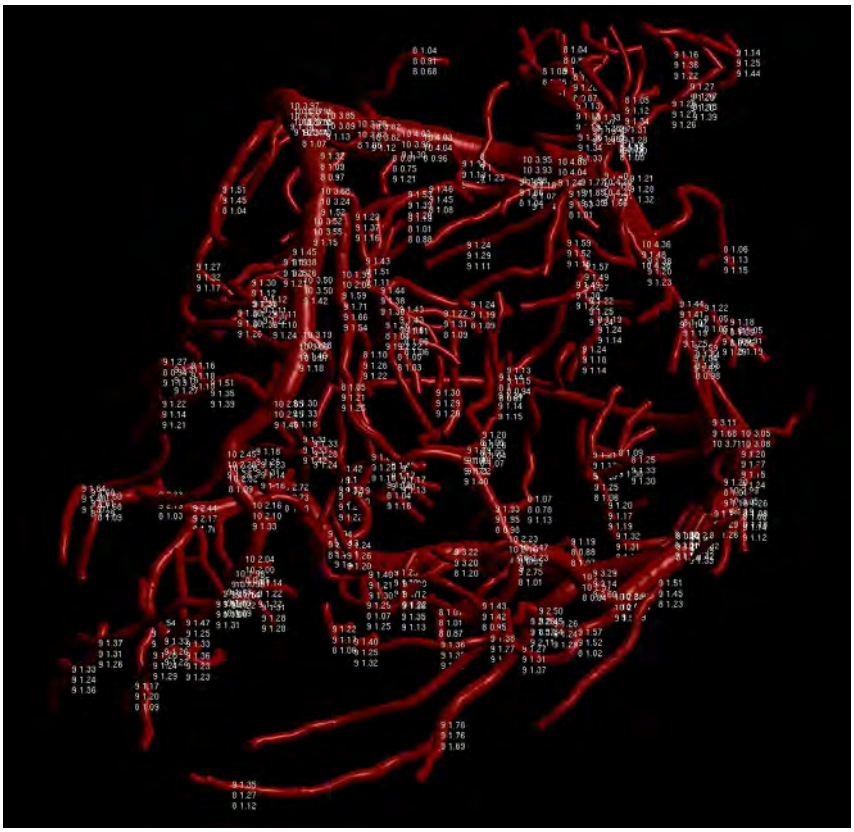


Figure 9 Focus+context visualization of vascular geometry and bifurcation angles.

amounts to 6 GBs of data. Since this amount of data exceeds the amount of memory typically available in desktop computers, out-of-core rendering approaches are required to visualize the data [50]. The entire geometric information is stored in a file and then accessed using memory-mapping so that the operating system automatically swaps chunks of the data set in and out as necessary. In addition to the geometry, a pressure simulation can be performed based on the geometric configuration of the vasculature. The resulting pressure data can then be used to color-code the vessels. Figure 10 shows such an image where the vessel segments are colored from red (low pressure) to green (high pressure).

Even though the rendering is of very high-quality, the fact that out-of-core methods are required results in rather slow rendering performance of about one minute per frame. This obviously is not interactive. However, the center line information combined with the vessel radii as described by the vascular model requires comparably less memory and fits into most desktop computer's main memory. Hence, a visualization technique that is solely based on these center lines and radii only would not require out-of-core techniques and therefore could be able to achieve faster rendering times.

The rendering technique described in this paper follows a similar approach than Stoll et al. [40] where the GPU is used to make lines appear cylindrical for rendering streamlines. In this paper, this idea is extended to allow for more general conic cylinders with different radii at each end since this is a requirement for rendering the vessel segments. In addition, the rendering technique is optimized for the purpose of rendering large-scale tree-shaped data sets and parallelized to run on the multiple cores of a CPU at the same time.

In this method, each vessel segment is represented only by a single quadrilateral. This approach provides two advantages. First, the quadrilateral can be solely computed based on the center line

Figure 10 High-quality rendering of the arterial tree of a pig heart including vessels from the most-proximal level down to the capillaries.

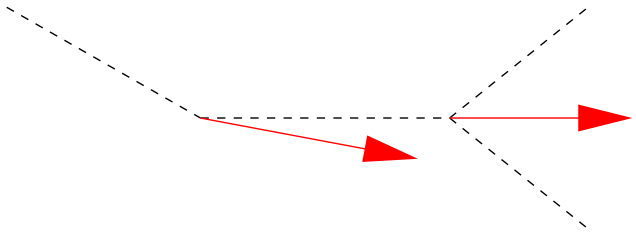


Figure 11 Computation of vessel directions (red arrows attached to its center point) based on the center lines of the vessels (dashed).

and radii information describing the vessel segments. Second, the rendering algorithm draws only four vertices per vessel segment compared to at least 18 vertices for the traditional approach. A fragment program can be used to make the quadrilateral appear like a cylindrical object. Of course, this rendering technique requires that the quadrilateral is always parallel to the projection plane in order to be completely visible. Only then a cylindrical appearance can be achieved. Consequently, the quadrilateral needs to be recomputed for every frame if the view has changed.

In a pre-computational step, the vessel directions at each center point are computed as the average of the direction of vessel segment ending at that center point and the direction of the vessel segment starting at that center point. In case of bifurcations, there may be more than one vessel segment starting at that center point. Consequently there are more than one direction available as well which then are included in the computation of the average direction. Figure 11 illustrates two examples. The resulting vessel direction is necessary to ensure a smooth transition between the vessel segments.

In order to compute the quadrilateral to represent a vessel segment, an offset vector is computed for every center point in such a way that this offset vector is parallel to the projection plane and orthogonal to the vessel direction at that center point. Figure 12 illustrates this procedure. Based on

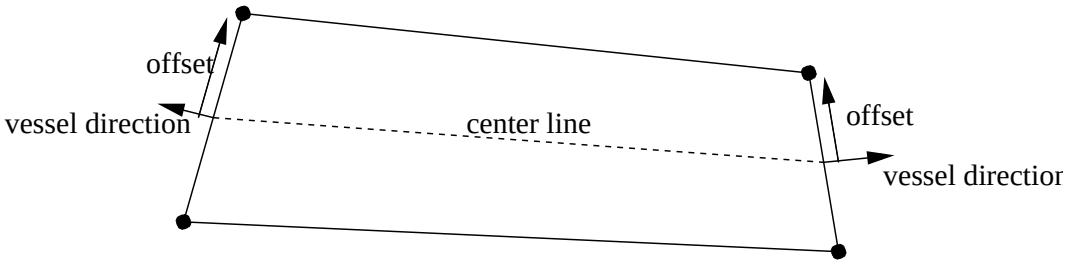


Figure 12 Computation of the quadrilateral representing a single vessel segment.

the offset vectors of two consecutive center points, four vertices can be computed by offsetting the center point in positive and negative direction of the offset vector as described in figure 12. This then results in the quadrilateral representing that vessel segment. Since neighboring quadrilaterals are based on the same vessel direction and therefore on the same offset vector, a smooth transition from one vessel segment to the next is guaranteed. Due to the fact that the offset vectors are parallel to the projection plane, it is guaranteed that the quadrilateral itself is parallel to the projection plane as well. This quadrilateral is then used as an approximation for the vessel segment.

Obviously, representing the vessel segment using a single quadrilateral works best when looking at the segment from the side. When looking at the vessel segment's cross-sectional area, it is approximated as a single vertical line since only a single quadrilateral is used as a representation. However, this is more obvious only for terminal vessels, i.e. vessel segments that represent leaves in the vascular tree. For vessels with adjacent segments most of the resulting artifact will be covered by that adjacent vessel segment. In addition, the percentage of vessels that are orthogonal to the projection plane and where the cross-sectional area would be visible is very small among all 11 million vessel segments. It would be easy to test for those cases and draw a circular area instead of a quadrilateral. However, this would require additional computation for the test and drawing more vertices for approximating a circle compared to drawing a quadrilateral. The test would have to be performed for every vessel segment. Considering the minimal artifact resulting from neglecting these cases and the loss of performance from the additional testing, this was not implemented to achieve the maximal performance possible.

In order to find the most efficient way of computing the quadrilaterals representing the vessel segments, three different approaches were implemented and compared. The first one computes all offset vectors for two consecutive center points individually. Hence, the offset vector for two consecutive quadrilaterals are computed twice. To avoid this double computation, two other approaches were tested. The first one stored all offset vectors in memory. This resulted in a huge array of offset vectors so that eventually the amount of available main memory was exceeded and the system started swapping during the rendering. Consequently, the performance of the rendering algorithm dropped. The second approach recursively traversed the vascular tree structure so that consecutive quadrilaterals were computed right after each other. This allowed the system to reuse the offset vector computed for the previous quadrilateral. However, the additional burden introduced by the recursion slowed down the rendering compared to the direct approach. Even transforming the recursion into a sequential approach using queues resulted in a slower performance. As a result it was found that computing both offset vectors for each quadrilateral results in the best performance possible despite the fact that some offset vectors are computed twice.

Rendering the vessel segments as quadrilaterals results in a rather flat appearance. In order to achieve a more cylindrical effect, a fragment program can be used that uses darker colors at the edge of the vessel, i.e. the sides of the quadrilateral that are not connected to other quadrilaterals. To achieve this effect, texture coordinates can be assigned to the vertices of the quadrilateral. Based on

```
void main (in float4 color : COLOR0,
           in float2 texcoord : TEXCOORD0,
           out float4 result : COLOR) {
  result = color * (-(texcoord.y - 0.5) *
           (texcoord.y - 0.5) + 1.0);
}
```

Figure 13 Fragment program for making a quadrilateral appear as a cylindrical object.

these texture coordinates, the fragment program can determine the local position of every fragment with respect to the vessel segment and change the color for the fragment if the edge of the vessel is approached. Figure 13 shows the Cg code of the fragment program used in this implementation. The fragment program simply fades the color when approaching the edge of the vessel. A similar effect could be achieved by using a simple texture. However, a fragment program has two advantages over a texture. First, the fragment program is resolution independent and computes the color for each fragment whereas a texture may produce aliasing artifacts. Second, the fragment program is color independent. Hence, color coding could be used for the vessel segments which would be preserved by the fragment program but overwritten by a texture.

Since CPU manufacturers nowadays often times provide more than just a single core, this additional computational resource can be used by the current implementation as well by simply adding a thread that computes quadrilaterals representing vessel segments on the CPU for very core. To synchronize the threads, a queue was introduced which contains all vessel segments that are not rendered yet. Then a separate thread is started which acquires the next vessel segments from the queue and computes the quadrilaterals to represent these vessel segments. The resulting geometry data is stored in another queue. This other queue is then processed by the main thread which renders the quadrilaterals contained in that queue. On a dual-core processor, this then results in two parallel processes which compute the necessary geometric data required for rendering the vascular data set. As can be seen in the next section, the current implementation of the algorithm scales very well and utilizes all computational resources available resulting in an increased rendering performance.

4 Performance

Figure 14 shows the resulting image of the performance rendering method based on a single quadrilaterals for every vessel segment. As can be seen in the image, the rendering quality is still very good. Every of the 11 million vessel segments is represented in this image.

The performance of the algorithms is very good as well. All measurements were performed on a computer equipped with an Intel Core2 Duo E6850 running at 3.0 GHz with 2 GB of memory and an NVidia Quadro FX3700 graphics card running Linux. A Western Digital Raptor WD740ADFD spinning at 10,000 RPM was utilized as the permanent storage device. The original implementation based on an out-of-core methodology requires over a minute (65 seconds on average) to render a single frame due to the fact that the geometry needs to be loaded from disk all the time. Figure 15 shows a plot of the rendering frame rate over a period of time.

When using the improved rendering algorithm, the performance increases significantly. Since the methodology no longer has to rely on the hard drive as the main storage medium, but instead can store all the necessary information in main memory, a rendering frame rate of about 0.68 is achieved which amounts to 1.5 seconds per frame on average. A plot of the frame rates over time is included in Figure 16.

The algorithm scales very well as can be seen when adding a second thread to assist in the

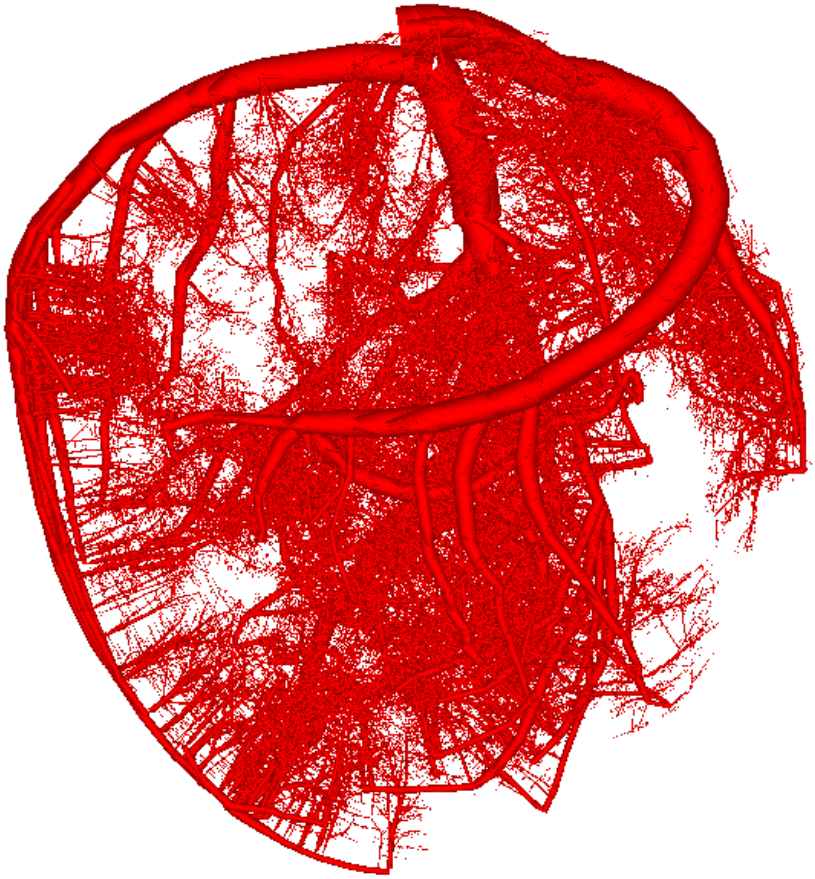


Figure 14 Performance rendering of the arterial tree of a pig heart.

computations necessary to compute all the quadrilaterals to represent the vessel segments. The rendering performance doubles to 1.32 frames per second which is equivalent to 0.75 seconds per frame which allows for an interactive investigation of the data set.

5 Conclusion and Future Work

This paper presented methodologies for extracting quantitative measurements from volumetric data sets at a very high accuracy. These morphometric measurements can be used to reconstruct the geometry of the vasculature resulting in an accurate visualization of the data. The measurements extracted include vessel length, vessel radius, and bifurcation angles which can be used as a statistical basis for growing additional vessels onto an existing vasculature. This way, large-scale vascular models can be generated that include vessels from the most proximal level down to the capillaries. Due to the vast amount of vessels included in such a data set, fast rendering methodologies are required to handle this amount of information. As discussed in this paper, using a more suitable rendering algorithm can result in an improvement of a factor of 43 in rendering performance over traditional out-of-core approaches.

In the future, the geometric reconstruction will be used to perform detailed flow simulation within the vessels to accurately analyze the flow properties within the vascular structure. Due to the high accuracy of the extracted measurements, the information can be used for disease detection, such as identification of areas of reduced flow. In addition, the extraction algorithm will be extended to

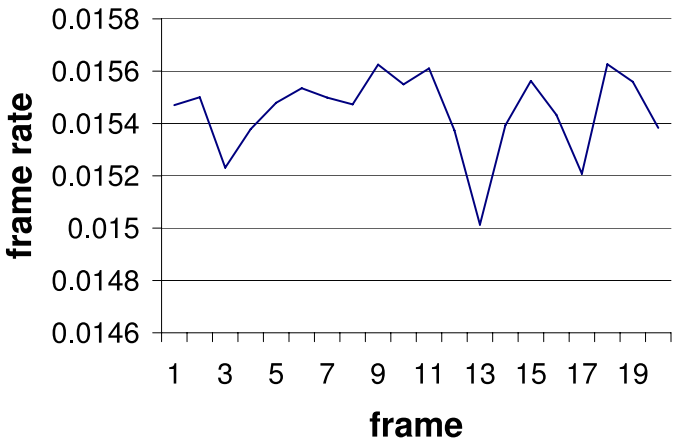


Figure 15 Rendering performance of out-of-core method for the large-scale arterial tree of a pig heart.

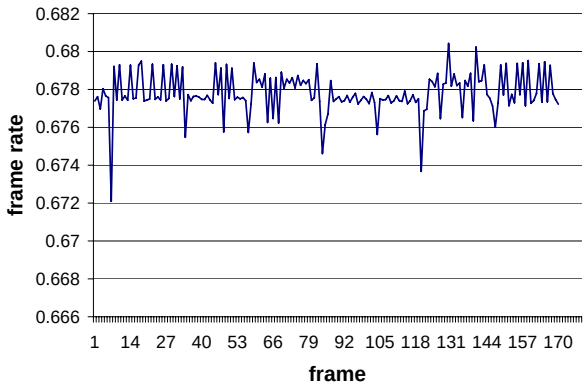


Figure 16 Rendering performance of performance method for the large-scale arterial tree of a pig heart.

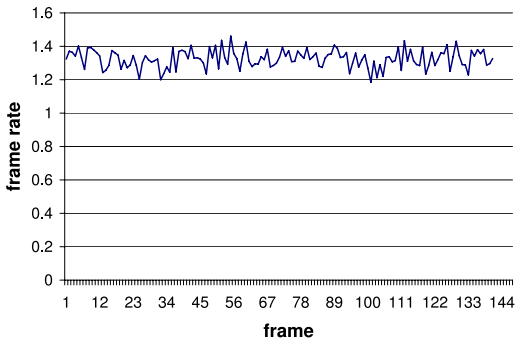


Figure 17 Rendering performance of performance method using two threads for the large-scale arterial tree of a pig heart.

identify individual fibers within volumetric data sets captured using optical coherence tomography of vascular tissue. An analysis of the fiber structure will result in the determination of mechanical properties of individual fibers to model the expansion of a fiber due to increased pressure induced by the blood flow.

6 Acknowledgements

The author wishes to thank Dr. Ghassan Kassab for a fruitful collaboration and providing the data sets. Also, the support of the Ohio Board of Regents and Research and Sponsored Programs, Wright State University, and its department of Computer Science and Engineering as well as the College of Engineering and Computer Science is greatly appreciated.

References

1 N. Amenta and R.-K. Kolluri S. Choi. The power crust. In *Proc. of 6th ACM Symp. on Solid Modeling*, pages 249–260, 2001.

2 G. Bertrand and Z. Aktouf. A three-dimensional thinning algorithm using subfields. *Vision Geometry III*, 2356:113–124, 1994.

3 I. Bitter, A. E. Kaufman, and M. Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Trans. Visualization and Comp. Graphics*, 7(3):195–206, 2001.

4 G. Borgefors. On digital distance transforms in three dimensions. *Computer Vision and Image Understanding*, 64(3):368–376, 1996.

5 G. Borgefors, I. Nyström, and G. S. Di Baja. Computing skeletons in three dimensions. *Pattern Recognition*, 32(7):1225–1236, 1999.

6 S. Bouix and K. Siddiqi. Divergence-based medial surfaces. *ECCV*, 1842:603–618, 2000.

7 D. Brunner and G. Brunnett. Mesh segmentation using the object skeleton graph. In *Proc. IASTED International Conf. on Computer Graphics and Imaging*, pages 48–55. ACTA Press, 2004.

8 J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

9 N. D. Cornea, D. Silver, and P. Min. Curve-skeleton applications. In *IEEE Visualization 2005*, pages 95–102, 2005.

10 M. Couprie and R. Zrour. Discrete bisector function and euclidean skeleton. *Lecture Notes in Computer Science*, 3429:216–227, 2005.

11 T. K. Dey and S. Goswami. Tight cocone: A water-tight surface reconstructor. In *Proc. 8th ACM Sympos. Solid Modeling Applications*, pages 127–134, 2003.

12 U. Eckhardt and G. Maderlechner. Invariant thinning. *Pattern Recognition and Artificial Intelligence*, 7:1115–1144, 1993.

13 P. Felkel, A. L. Fuhrmann, A. Kanitasar, and R. Wegenkittel. Surface reconstruction of the branching vessels for augmented reality aided surgery. *BIOSIGNAL*, 1:252–254, 2002.

14 N. Gagvani and D. Silver. Parameter controlled volume thinning. *Graphical Models and Image Processing*, 61(3):149–164, 1999.

15 N. Gagvani and D. Silver. Animating volumetric models. *Academic Press Professional*, 63(6):443–458, 2001.

16 P. Golland and W.E.L. Grimson. Fixed topology skeletons. In *IEEE CVPR*, volume 1, pages 10–17, 2000.

17 W. Gong and G. Bertrand. A simple parallel 3d thinning algorithm. In *IEEE Pattern Recognition*, pages 188–190, 1990.

18 H. K. Hahn, B. Preim, D. Selle, and H. O. Peitgen. Visualization and interaction techniques for the exploration of vascular structures. In *IEEE Visualization 2001*, pages 395–402, 2001.

19 T. He, L. Hong, D. Chen, and Z. Liang. Reliable path for virtual endoscopy: Ensuring complete examination of human organs. *IEEE Trans. Visualization and Comp. Graphics*, 7(4):333–342, 2001.

20 Peter J. Hunter and Thomas K. Borg. Integration from proteins to organs: the iups physiome project. *Nature Reviews | Molecular Cell Biology*, 4:237–243, 2003.

21 B. Kaimovitz, Y. Lanir, and G. S. Kassab. Large-scale 3-d geometric reconstruction of the porcine coronary arterial vasculature based on detailed anatomical data. *Ann. Biomed. Eng.*, 33(11):1517–1535, 2005.

22 T. Lee and R.L. Kashyap. Building skeleton models via 3-d medial surface/axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

23 S. Lobregt, P. W. Verbeek, and F. C. A. Groen. Three-dimensional skeletonization: Principle and algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):75–77, 1980.

24 C. Lohou and G. Bertrand. A 3d 12-subiteration thinning algorithm based on p-simple points. *Discrete Applied Mathematics*, 139:171–195, 2004.

25 C. M. Ma and M. Sonka. A fully parallel 3d thinning algorithm and its applications. *Computer Vision and Image Understanding*, 64(3):420–433, 1996.

26 G. Malandain and S. Fernandez-Vidal. Euclidean skeletons. *Image and Vision Computing*, 16:317–327, 1998.

27 A. Manzanera, T. Bernard, F. Preteux, and B. Longuet. A unified mathematical framework for a compact and fully parallel n-d skeletonization procedure. *Vision Geometry VIII*, 3811:57–68, 1999.

28 Y. Masutani, K. Masamune, and T. Dohi. Region-growing-based feature extraction algorithm for tree-like objects. *Visualization in Biomedical Computing*, pages 161–171, 1996.

29 S. Oeltze and B. Preim. Visualization of Anatomic Tree Structures with Convolution Surfaces. In *IEEE/Eurographics Symposium on Visualization*, pages 311–320, 2004.

30 S. Oeltze and B. Preim. Visualization of Vasculature with Convolution Surfaces: Method, Validation and Evaluation. *IEEE Trans. Medical Imaging*, 24(4):540–548, 2005.

31 K. Palagyi and A. Kuba. Directional 3d thinning using 8 subiterations. *Discrete Geometry for Computer Imagery: Lecture Notes in Computer Science*, 1568, 1999.

32 K. Palagyi and A. Kuba. A parallel 3d 12-subiteration thinning algorithm. *Graphical Models and Image Proc.*, 61(4):199–221, 1999.

33 D. Perchet, C. I. Fetita, and F. Preteux. Advanced navigation tools for virtual bronchoscopy. In *SPIE Conf. on Image Processing: Algorithms and Systems III*, volume 5298, pages 147–158, 2004.

34 C. Pudney. Distance-ordered homotopic thinning: A skeletonization algorithm for 3d digital images. *Computer Vision and Image Understanding*, 72(3):404–413, 1998.

35 A. Puig, D. Tost, and I. Navazo. An interactive celebral blood vessel exploration system. In *IEEE Visualization 97*, pages 443–446, 1997.

36 Felix Ritter, Christian Hansen, Volker Dicken, Olaf Konrad, Bernhard Preim, and Heinz-Otto Peitgen. Real-Time Illustration of Vascular Structures. In *IEEE Visualization 2006*, pages 877–884, 2006.

37 P.K. Saha, B.B. Chaudhuri, and D. Dutta Majumder. A new shape preserving parallel thinning algorithm for 3d digital images. *Pattern Recognition*, 30(12):1939–1955, 1997.

38 H. Schirmacher, M. Zöckler, D. Stalling, and H. Hege. Boundary surface shrinking - a continuous approach to 3d center line extraction. In *Proc. of IMDSP*, pages 25–28, 1998.

39 J.A. Sethian. Fast marching methods. *SIAM Review*, 41(2):199–235, 1999.

40 Carsten Stoll, Stefan Gumhold, and Hans-Peter Seidel. Visualization with stylized primitives. In Claudio Silver, Eduard Gröller, and Holly Rushmeier, editors, *IEEE Visualization 2005*, pages 695–702, 2005.

41 H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling Int'l*, pages 130–139, 2003.

42 K. Suresh. Automating the cad/cae dimensional reduction process. In *ACM Symp. On Solid Modeling and Applications*, pages 76–85, 2003.

43 S. Svensson, I. Nystrom, and G. Sanniti di Baja. Curve skeletonization of surface-like objects in 3d images guided by voxel classification. *Pattern Recognition Letters*, 23(12):1419–1426, 2002.

44 A. Telea and A. Vilanova. A robust level-set algorithm for centerline extraction. In *Eurographics/IEEE Symp. On Data Visualization*, pages 185–194, 2003.

45 Y. F. Tsao and K. S. Fu. A parallel thinning algorithm for 3d pictures. *Computer Vision, Graphics and Image Proc.*, 17:315–331, 1981.

46 L. Wade and R. E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer*, 18(2):97–110, 2002.

47 M. Wan, F. Dachille, and A. Kaufman. Distance-field based skeletons for virtual navigation. In *IEEE Visualization 2001*, pages 239–246, 2001.

48 Thomas Wischgoll. *to appear in Visualization and Mathematics*, chapter Computing Center-Lines: An Application of Vector Field Topology. Springer-Verlag, Heidelberg, Germany, 2008.

49 Thomas Wischgoll, Joerg Meyer, Benjamin Kaimovitz, Yoram Lanir, and Ghassan S. Kassab. A Novel Method for Visualization of Entire Coronary Arterial Tree. *Annals of Biomedical Engineering*, 35(5):694–710, 2007.

50 Thomas Wischgoll, Joerg Meyer, Benjamin Kaimovitz, Yoram Lanir, and Ghassan S. Kassab. A novel method for visualization of entire coronary arterial tree. *Annals of Biomedical Engineering*, 35(5):694–710, 2007.

51 Z. Yu and C. Bajaj. A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion. In *CVPR 2004*, pages 415–420, 2004.

52 Y. Zhou, A. Kaufman, and A. W. Toga. Three-dimensional skeleton and centerline generation based on an approximate minimum distance field. *The Visual Computer*, 14:303–314, 1998.

53 Y. Zhou and A. W. Toga. Efficient skeletonization of volumetric objects. *IEEE Trans. Visualization and Comp. Graphics*, 5(3):196–209, 1999.

From Visualization to Visually Enabled Reasoning

Joerg Meyer[1], Jim Thomas[2], Stephan Diehl[3], Brian Fisher[4], Daniel A. Keim[5], David H. Laidlaw[6], Silvia Miksch[7], Klaus Mueller[8], William Ribarsky[9], Bernhard Preim[10], and Anders Ynnerman[11]

1 **University of California, Irvine, USA** – jmeyer@uci.edu
2 **Pacific Northwest National Laboratory, USA** – Jim.Thomas@pnl.gov
3 **University of Trier, Germany** – diehl@uni-trier.de
4 **Simon Fraser University, Vancouver, Canada** – bfisher@sfu.ca
5 **University of Konstanz, Germany** – keim@inf.uni-konstanz.de
6 **Brown University, USA** – dhl@cs.brown.edu
7 **Vienna University of Technology, Austria** – silvia@ifs.tuwien.ac.at
8 **Stony Brook University, New York, USA** – mueller@cs.sunysb.edu
9 **University of North Carolina, Charlotte, USA** – ribarsky@uncc.edu
10 **University of Magdeburg, Germany** – preim@isg.cs.uni-magdeburg.de
11 **Linköpings Universitet, Sweden** – Anders.Ynnerman@itn.liu.se

Abstract

Interactive Visualization has been used to study scientific phenomena, analyze data, visualize information, and to explore large amounts of multi-variate data. It enables the human mind to gain novel insights by empowering the human visual system, encompassing the brain and the eyes, to discover properties that were previously unknown. While it is believed that the process of creating interactive visualizations is reasonably well understood, the process of stimulating and enabling human reasoning with the aid of interactive visualization tools is still a highly unexplored field.

We hypothesize that visualizations make an impact if they successfully influence a thought process or a decision. Interacting with visualizations is part of this process. We present exemplary cases where visualization was successful in enabling human reasoning, and instances where the interaction with data helped in understanding the data and making a better informed decision.

We suggest metrics that help in understanding the evolution of a decision making process. Such a metric would measure the efficiency of the reasoning process, rather than the performance of the visualization system or the user. We claim that the methodology of interactive visualization, which has been studied to a great extent, is now sufficiently mature, and we would like to provide some guidance regarding the evaluation of knowledge gain through visually enabled reasoning. It is our ambition to encourage the reader to take on the next step and move from information visualization to visually enabled reasoning.

1998 ACM Subject Classification I.3.8 Applications, J.0 General

Keywords and phrases Interactive Visualization, Reasoning

Digital Object Identifier 10.4230/DFU.SciViz.2010.227

1 Introduction

Visualization of information alone does not provide new insights. It is the discourse between the human brain and the masses of information that enables reasoning and analytics. Information visualization and the science of interaction must not only focus on rendering

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 227–245
Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

performance and methods of human-computer interaction (HCI), but also on successful reasoning. Traditionally, research on HCI uses results from cognitive sciences to enhance the user's experience and performance when interacting with a visualization system. A successful visual data analysis system is usually characterized by an improvement in these two user-related categories, measured by means of a user study.

The next step in the evolution of visual analytics is the integration of interactive visualization and human reasoning. Interactive reasoning is a process that takes place when a dynamic visualization system responds to the user's input and aids the user in gaining *new insights*. A tight coupling between *cognition, interaction* and *visual analytics* is necessary to enable the user to make informed decisions. We suggest a new metric, which is required to evaluate the efficiency of visually enabled reasoning.

The next parts of this chapter provide an overview of some key definitions and define the new science of visually enabled reasoning. Chapter 2 gives a strong motivation for the shift from *visualization* to *visually enabled reasoning*. Chapter 3 outlines a roadmap to *visually enabled reasoning* and describes visual reasoning tools. A discussion is provided in chapter 4, which is intended to motivate the reader to take an integrated approach and to think about visual analytics as a tool that empowers humans to make better informed decisions. As a special feature, we present cases where *interactive visualization* and *visual analytics* were successful in *enabling human reasoning*.

1.1 Definition of Terms

Visual Analytics — Visual analytics is defined as the science of analytical reasoning facilitated by interactive visual interfaces [51]. This research area emerged from the fields of information visualization, scientific visualization, and data analysis. Thus, visualization as one method of analyzing data is combined with data analysis, e.g. statistics with respect to correlations, clusters and distributions of data. Visual analytics goes beyond visual data mining [28], which also emphasized a combination of data analysis and visualization by incorporating HCI. Visual analytics focuses on data analysis by means of interactive data visualization and human-computer interaction. It is understood that the human visual and cognitive system is the most powerful tool for understanding the complex relationships between data elements. Adapting a dynamic visualization system to match the abilities of the human cognitive system is an instrumental key to optimizing user experience and performance. A system that matches visual perception, with respect to resolution, focus, attention and detail without overloading human senses is most suitable for efficient interpretation of large data sets. Human reasoning is an important and indispensable element in this process. It is important to note that the cognitive process of reasoning does not end with a good understanding of the data. Conclusions must be drawn, leading to actions. These actions include planning, decision making, and going back to the visualization in order to change the simulation or planning scenario.

Science of Interaction — The contemplated science of interaction through visual interfaces is broad. From the human perspective, it has both perceptual and cognitive components. Since we are focused on visually enabled reasoning, we will concentrate mostly on the cognitive component here. As shown in Figure 1, interactive visualization is the pervasive mediating component between the user and data, i.e., between information and insight. The science of interaction must formulate principles for all these interactions out of which both descriptive and predictive models must arise. From these principles and inference models, rules of design for interactive visualization systems and for the interactive reasoning will be formulated. There will be general rules that will apply across all visualizations, and

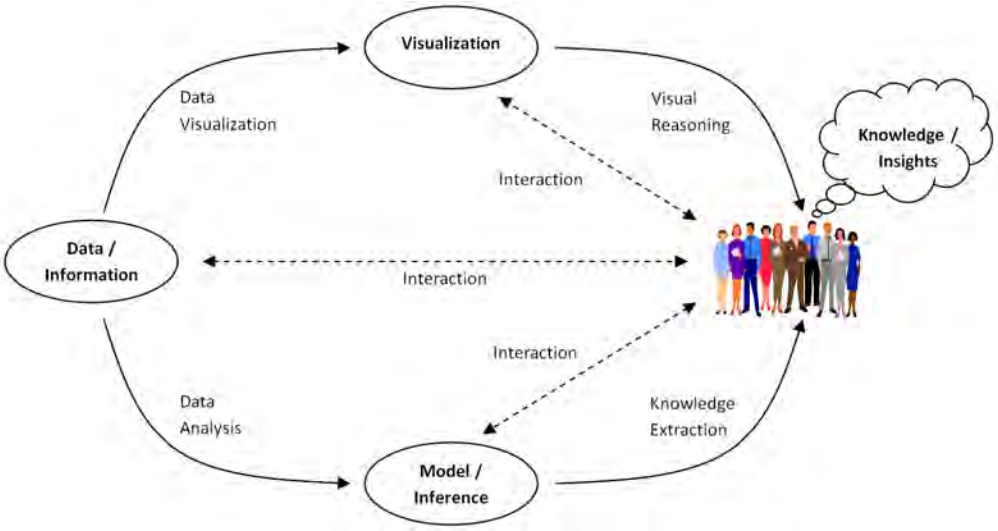


Figure 1 Visually Enabled Reasoning.

there will be specific rules, based on the general rules but developed for domain-specific problems. In this set of principles and models, there must be a model for the human cognitive process engaged in the rhythm of interaction, that is, in simultaneously probing data and analyses, and in assessing visualizations. To emphasize, the human cognitive process must be modeled explicitly as an integrated part of any model of visually enabled reasoning. Once modeled, it can also be evaluated, and the efficiency of the reasoning process can be assessed (Figure 1).

Cognition — The Oxford Dictionary defines *cognition* as "the mental action or process of acquiring knowledge and understanding through thought, experience, and the senses ()". In the context of visually enabled reasoning we extend theories of problem solving and reasoning to include ways in which experience and the senses integrate human knowledge and expertise with information from interactive visual analytic technologies. The ability of these systems to shift the burden of information processing to perceptual processes should enable cognitive operations to take place at a higher level over more complex cognitive "chunks". In this view, cognition cross-cuts levels of the perception/decision/action hierarchy, and can best be examined within the framework of abilities and constraints that make up the human *cognitive architecture*.

Interactive Reasoning is the process of distinguishing between ideas in order to create new relations and insights based on collected evidence. A significant reasoning process is the building and testing of hypotheses (which may involve choosing between competing hypotheses). A hypothesis can either be an argument (e.g., a decision process that determines a course of action or point of view) or a model (e.g., a predictive representation of how something works). Evidence and also reasoning artifacts are derived from relevant information, data, analyses, or previous knowledge. Reasoning is intrinsically interactive. Here we are concerned with reasoning mediated by interactive visualization where the visualization is the means of presenting to humans the relevant information, analyses, and knowledge and also the interface through which the human manipulates the information, analyses, etc. to advance the reasoning process.

Visual Reasoning usually incorporates different people with considerably different qualifications and backgrounds. There are people developing tools for visual reasoning, and other people using these tools to analyze their data. In most scenarios, however, the people who acquire data (the authors of the data) are different from the actual customers. In medicine, for example radiology technicians and radiologists acquire data using a specific imaging protocol based on the request of a referring physician who is the actual end-user. A surgeon, for instance, might use these data for preoperative planning. The development of visual reasoning tools requires a close collaboration of all stakeholders from the onset.

The concept of participatory software development which is well-known in the HCI community since the 80ies [20][24] is essential to enabling end users to influence design decisions and ensure that the constraints of the particular domain are included in system design at an early stage. Visual reasoning and problem solving strategies are very complex issues, and it is very difficult to design them right from the beginning with reliable formal specifications directly for supporting scientists. The process of tool development based on a preliminary requirement analysis and later determining all the flaws of the design in a user study is just not effective. Instead, the authors and the users of the data should be considered as active co-developers and not as passive sources of information, who are supposed to answer questions such as "How often do you need this feature?" This active involvement of end users requires that tool developers communicate in a not too technical way with the domain expert. In particular, formal specifications, such as nested state transition diagrams or even use cases are not a good basis for a discussion with end users. The scenario-based approach by Carroll and Beth Rosson [43] is promising, because it yields specifications which can be discussed with end users, as well as guide the design, development and evaluation process.

Dynamics — Modeling human dynamics is a critical part of understanding and evaluating visually enabled reasoning. The study of human dynamics includes the adaptability and interaction between models and visual interfaces. Today, rapidly growing technologies such as Internet, mobile computing and sensor web have enabled new patterns of human interactions, from social networks to physiological functions [8]. Human dynamics has become more complex and more venerable. Unfortunately, our understanding of human dynamic behavior and machine interaction is very limited. Much is invisible. To make invisible visible is the goal of visual analytics, and to help model the complex, dynamic human machine interaction is the aim of this article.

Social networks represent a good model for studying complex human dynamics when many individuals and large computing networks are involved. The fundamental studies of social networks such as the *six degrees of separation* and the *power law of linked interactions* shed light on the scalability of human networking activities. Those remarkable models enrich our in-depth understanding of the dynamics in a very large network, which is a challenge to a visualization system.

Insight Gain — Gaining insights is a main goal of visual analytics or information visualization methods, however, as Yi et al. point out [57], although a few definitions of insight exist, no commonly accepted definition has emerged in the community. For example, Card et al. [9] declare that *the purpose of visualization is insight, not pictures* (p. 6). Saraiya et al. [44] define insight as an "individual observation about the data by the participant, a unit of discovery" (p. 444). North categorizes insight to be *complex, deep, qualitative, unexpected, and relevant* [34]. Yi et al. [57] identified four types of processes through which people gain insight: Provide Overview, Adjust, Detect Pattern, and Match Mental Model. According to Chang et al. [11] the scope of definitions of insight in the visualization community differs from that of the cognitive community: the definitions of insight in the

visualization community are generally broader but vaguer than those in cognitive science. They suggest defining insight in two parallel meanings: (1) a term equivalent to the cognitive science definition of insight as a moment of enlightenment, and (2) a broader term to mean an advance in knowledge or a piece of information.

1.2 A New Science

We propose a new science of human interaction with information. This new science is based on insight gain through visually enabled reasoning. Interactive visualization, which has matured into a set of well studied tools, plays an important role in this endeavor. Traditionally, user studies have been used to evaluate performance and utility of visualization systems. We propose a new metric which measures human reasoning instead of visualization or interaction performance. We will ask what kind of experiments can be conducted to measure human reasoning and whether user studies are an appropriate method for evaluating this process.

The paradigm of *visually enabled reasoning* is a whole new kind of evolution. How do we create a model to measure human reasoning? How do we measure insight gain? In order to answer these questions, cognitive scientists, visualization researchers and human-computer interaction specialists must collaborate in an interdisciplinary effort to define both system specifications and metrics for human reasoning. This collaboration has already begun, as evidenced by initial steps toward a human cognitive model undertaken by visualization and cognitive scientists working together [23].

1.3 Importance of Interdisciplinary Collaboration

To enable scientists and domain experts to use advanced visual analytics system successfully requires a substantial knowledge of highly specialized scientific domains. This in-depth knowledge is usually not completely formalized. It is applied in an intuitive and implicit manner and thus cannot be easily extracted as a basis for computer support. As an example from engineering, the simulated flow in a designed engine model is analyzed with respect to complex flow patterns using a variety of analysis and visualization methods. In medicine, surgeons have to make treatment decisions based on the assessed risk associated with different interventions, often using interdisciplinary discussions, such as in a tumor board for their planning tasks. Intensive interdisciplinary collaborations are an essential basis for providing visual reasoning tools in such advanced application areas.

2 Motivation

One of the key aspects of research on visually enabled reasoning will be focused investigation of the structure of human perception, cognition and action, i.e. human cognitive architecture. In this effort, we must understand how cognitive functions are distributed across perceptual and perceptually-guided motor processes.

For example, low-level perception of events is itself an inferential process. Irvin Rock's "logic of perception" [41][42] refers to the visual system's ability to compensate for inadequate sensory information (the so-called "poverty of the stimulus") and to reconcile conflicting sensory information through processes of unconscious inference. Perceptual inference is largely data-driven, and does not take into account the perceiver's conscious thoughts, beliefs, intentions, etc.

Pylyshyn's "cognitive impenetrability" test [37] distinguishes these two levels: while learning does train perception's inferential processes (and so individuals will differ one from

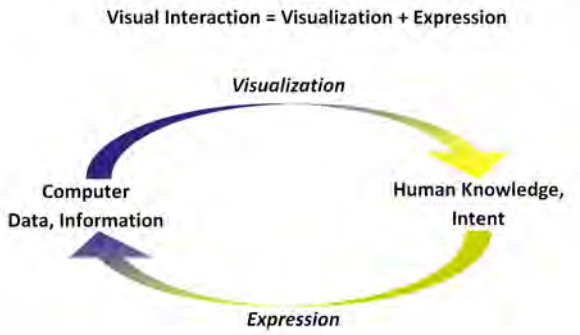


Figure 2 Conveying visual information to the brain.

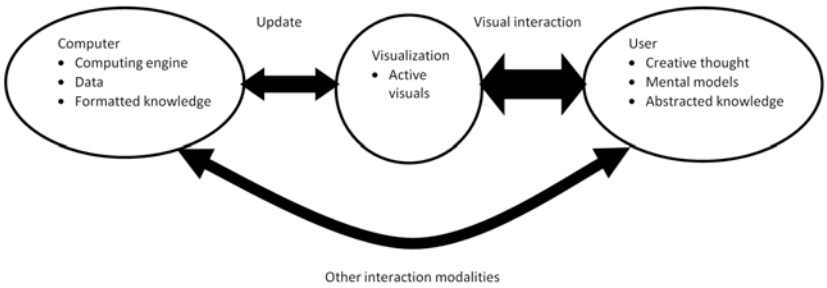


Figure 3 Visualization-centric human-computer interaction.

another in their abilities, and experts will differ from novices), the perceiver's conscious thoughts, beliefs, intentions, etc. do not actively participate in perception. Thus input from end-users, e.g. verbal protocols of a "think aloud" session, can give only limited insight into their perceptual logic. The term "metacognitive gap" [17] describes this counterintuitive break between the ways in which these two logics must be understood, and hence the need for a new cognitive science of human interaction with visualization systems.

From the perspective of the cognitive architecture of sensemaking, when we understand a dataset we do so by attending to the conceptual implications of information that is itself constructed by the pre-conscious logic of perception. Visually enabled reasoning will be most successful when we fully understand how to design images and dialogs that enable the logic of perception to support the logic of conscious reasoning.

In Figure 2, visualization includes both the technical process of producing a visual display of information as well as the process of conveying this information into the human brain to form a mental image. Expression includes both the mental process of deciding what to express as well as the technical process of conveying this to the computer.

In Figure 3, the visualization is in the center and forms the medium that links the human and the computer. The widths of the links themselves illustrate the bandwidth of these channels. Note that the visualization-human bandwidth is perceived the largest, while that of the other modalities is much smaller. The visualization is not just an image, but an active and responsive process, equipped with autonomous methods, such as (semantic) zooming, graphics rendering, context+focus displays, etc. The visualization is updated by both user and computer, and it also stimulates or provokes new updates by both of these processes. In

this diagram, computer and human are equal processes both capitalizing on their individual strengths. The visualization forms a bilateral interaction medium.

The advance in computing power allows for the recomputation of parts of the visualization in real or near real time and thus enables modern interaction techniques like brushing and linking [16][53], panning and zooming [3], focus-context [33], magic lenses [4], as well as animated transitions. But not only the visualization, but also the underlying data and analyses can be partially recomputed.

Highly interactive and dynamic techniques are essential for supporting visual reasoning. With these techniques, changes between complex visualizations can now be directly observed instead of having to be interpreted which poses a much higher mental effort.

Visual or computational steering [54] is the concept that on the basis of a visualization of the current status parameters of a running simulation or analysis are changed and the results are visualized immediately. Thus, visual steering allows for the control of the simulation in real time, instead of running a simulation first and then doing a post-mortem analysis.

As a result, the visualization has turned from an end product of the analysis process to a user interface in an interactive and iterative analysis process.

3 Roadmap to Visually Enabled Reasoning

This chapter outlines the system components and conceptual considerations required for visually enabled reasoning. Most of the system components, such as interaction devices, dynamic and adaptive visualization, animation and sensory feedback, have already been developed and need to be combined with visual cognition techniques. We discuss tools for visually enabled reasoning and define metrics for evaluating these tools with respect to supporting the process of computer-aided human reasoning.

3.1 Science of Interaction

In terms of visually-enabled reasoning, interaction is the means by which the human and the computer work together (see also section 1.1 for a definition of terms). It makes the interface permeable and communication and collaboration possible. The visualization subsystem needs to be dynamic and adaptive, because the analysis and cognitive tasks can be significantly different depending on what direction the user takes in her investigation. Scientific research up to now usually employs user studies, task analyses, usability tests, and system performance evaluations for the evaluation, validation, and improvement of these systems. What has been missing to support visually-enabled reasoning is research that focuses on cognition, the reasoning and argument process itself, and the ability to make decisions. One aspect that must be focused on is the idea of a *mixed initiative system*, where the human and computer work together in intimate collaboration, each doing what it does best and then sharing results at the right time. New models that address the maintaining of cognitive flow for this mixed initiative system, through visual representation and especially through interaction, are starting to be developed [22][23], but much more work remains to be done.

Edward Tufte introduced the concepts of micro/macro reading, in which *detail cumulates into larger coherent structures* [52]. In this sense a figure or diagram can be graphically read at the level of larger contextual structure, at the detail level, or both view connected to (similar to Focus and Context techniques). A good example is the illustration of sleep and wake for newborn infants: each of the individual observations can be seen (micro reading), but collectively all observations reveal the larger 25-hour and then the 24-hour circadian cycles (macro reading). These concepts can be transferred to handle interactions. Interact

methods can be defined on the micro level (e.g., selecting data points) and on the macro level according to particular tasks.

The classical interaction methods proposed by Shneiderman [47] (overview, zoom, filter, details-on-demand, relate, history, and extract) or by Chuah and Roth [13] (basic visualization interaction [BVI] operations: graphical operations [encode data, set graphical value, manipulate objects], set operations [create set, delete set, summarize set, and others], and data operations [add, delete, derived attributes, and others]) can be seen on the micro level.

Yi et al. [56] studied different interaction methods and proposed a novel user intent-based categorization schema, which could be seen on the macro level as

1. Select: mark something as interesting (e.g., brushing),
2. Explore: show me something else (e.g., navigation),
3. Reconfigure: show me a different arrangement (e.g., swap x and y axis of a scatter plot),
4. Encode: show me a different representation (e.g., switching to a different visualization method),
5. Abstract/Elaborate: show me more or less detail (e.g., details on demand),
6. Filter: show me something conditionally (e.g., dynamic queries), and
7. Connect: show me related items (e.g., linking).

Yi et al. [56] also presented the different categorization schemata developed in the past, which makes quite clear that there is no established science of interaction available yet. However, according to the needs and purposes, different categorization schemata are used. As mentioned in Yi et al. [56], their proposed categorization schema does not cover all aspects, which were discovered in their literature and system research.

3.2 User Modeling

User modeling is a mature concept used in the design of intelligent user interfaces [29][30][31]. User modeling is concerned with deriving assumptions on the current user based on their previous activities with the system. These assumptions may be exploited to enhance the interactive use of a visual reasoning system. Thus, the system is adaptive with respect to its user.

The user model might be something as simple as a data structure with parameters and associated values. It might be more complex and represent also ”why” the system has arrived at a certain assumption and describe the certainty of that assumption. In fact, sophisticated knowledge representations are exploited, e.g., in advanced hypertext systems [6][7].

Obviously, user modeling is a concept which is only applicable to systems that are used frequently; otherwise the user modeling effort does not pay off. Once this concept is implemented, experienced long-term users of a visualization reasoning system can be effectively represented and supported. At the lowest level, default values for visualization techniques, such as colors, parameters for diagram representations, isolines and so forth may be adapted to the preferences of the user. At a higher level, the selection of visualization techniques and the screen layout might be adapted. For instance, the system ”classifies” the current task, looks for the visual reasoning strategies applied frequently in the past, and ”suggests” a layout, where, for example, a 3-D visualization, a cross-sectional 2-D visualization and a 2-D histogram of data values and gradient magnitude are presented simultaneously. Similarly, the selection of data analysis techniques, such as cluster analysis, principal component analysis and correlation analysis might be adapted based on previous decisions of the user. Finally, the composition of data analysis results and visualization techniques is based on a huge parameter space and thus would benefit from narrowing this

space based on ”intelligent” decisions. Therefore, a visually enabled reasoning system should include models of the following:

- User,
- Visualization Environment (Output Devices), and
- Data Analysis Methods.

These three components and their combination are essential for modeling visual reasoning.

User modeling, however, describes only one source for providing adaptive behavior. The selection and parameterization of visualization and data analysis techniques should also be adaptive with respect to the available computational output and device resources. Given the large variety of output devices, differing in spatial resolution, number of gray levels, physical size, and, for instance, the ability to render stereoscopic images, the suitability of visualization techniques strongly depends on the particular output device. Adaptive behavior should be guided primarily by existing knowledge of the properties of visual perception, including different abilities of user groups at varying age levels.

Finally, even the same user working in the same environment (computational hardware and output devices) often needs a variety of interactions and parameter adjustments for carrying out a ”similar task”. A large amount of these adjustments is due to the special characteristics of the data sets. Signal-to-noise ratio, frequency, the spatial distribution of unreliable data which should be removed, the amount and characteristics of inhomogeneity, and other imperfections of real-world measured data should be analyzed in order to ”suggest” a meaningful sequence of visual reasoning steps.

The concept of exploiting the great potential of enhancing visual reasoning with adaptive components, however, is not straightforward. Successful examples for user models with the special requirements of visual reasoning in mind have not been developed yet. Existing strategies for collecting, structuring and representing user data have to be refined and evaluated. As a general issue of user modeling, privacy issues must be taken into account. Efficient user modeling inevitably requires collecting and analyzing large amounts of data of specific users. It must be ensured that this data is securely stored and that it cannot be accessed by unauthorized persons.

The experience of many users with adaptive behavior in computing software is often negative. In some cases, users could not predict the feedback of the system, resulting in a feeling of losing control. Moreover, the suggested solutions of the software often do not correspond with the choices that the user would make in a particular situation. Systems tend to over-generalize user interactions from the past. All of these problems have to be taken seriously.

Adaptivity has obvious limitations and therefore should be restricted to situations where a strong effect is likely to occur. Raskin pointed out that users are only effective if they form habits, if the system they use performs in a predictive way and if they are not overwhelmed by necessary, but semantically less relevant decisions [38]. If a system continuously changes its behavior — based on knowledge acquired by analyzing interaction patterns of the user — forming a habit becomes difficult, because the learning process of the user interferes with the adaptive system component [32]. Therefore, changes of the system’s behavior should not occur frequently, they should be motivated and explained appropriately to the user, and they only come into effect after the user accepts them.

As stated above, a visual reasoning environment, in particular a collaborative one will potentially incorporate very different people with considerably different qualifications and backgrounds. These parameters will determine the inherent complexity and style of the visualizations used. A main obstacle in achieving true human-like artificial intelligence is the

fact that human consciousness is highly dependent on personal semantic models, knowledge, past experiences, skills, preferences, and the like, which are hard to capture and to encode in machines. We face the same obstacles when attempting the encoding of data and information into visual representations. These differences are expressed horizontally (same complexity, but different representation) as well as vertically (reduction of complexity). While the former is more a function of personal preferences, possibly motivated by professional or community background, the latter is a function of educational background, classification of the information visualized, and task-mandated (minimal) requirements. Thus, there will not be a single one visual reasoning environment that fits all participants, yet it must allow all participants to communicate with one another and with the computing engine as well. The key here is to develop a parameterized model of users and tasks, methodologies to acquire and test them, procedures to generate the user- and task-suitable visualizations, and finally appropriate means to translate one representation into another (also known as grounding [5][14]). For this, we need to capture personal preference vectors (in terms of visualization paradigms) and correlate them with other user information, such as background, education level, and others. These frameworks can then also be used to parameterize tasks and knowledge. A rich suite of user studies is needed to provide these models, and market research has developed statistical frameworks (such as conjoint analysis [19][26] to efficiently acquire these, with a minimal set of users and user involvement. Such models will then yield adaptive user interfaces that can eventually predict the best visual representations of the information at hand. Finally, once such models are formulated, systems can automatically coach analyst users in the development of strategies or plans of attack for conducting more complex analyses. One will also be able to generate templates that cover the best-fitting strategies for most efficient analysis. Schneider-Hufschmidt et al. [45] provide a good overview on concepts of user modelling and using user models to enhance interaction, and most of the concepts described in this article are still applicable more than a decade later.

3.3 Visual Perception

Research in cognitive science aims at explaining how the human visual system creates perceptual experiences from visual stimuli. The results of this research have many implications for the design of visualizations. For example, due to the different densities and kinds of receptors in the human eye, color should be used for detail information in the user's focus, but not in the periphery (context), and motion can be used as a stimulus in the periphery. Many interesting insights have been gained by looking at visual illusions. Recently, Changizi et al. [12] proposed that many of these illusions are due to the predictive power of the human visual system that tries to compensate for neural delay. It is an open question how we can exploit this predictive power in the design of visualizations to convey information faster.

Another cognitive resource that current visualization techniques do not fully exploit is the human visual memory [46][48] — our ability to store vast numbers of (sufficiently different) pictures.

3.4 Visual Reasoning Tools

Visual reasoning tools are essential for a wide range of tasks. In supervisory control tasks, often characterized by large displays, it is essential that abnormalities are presented (and eventually aurally added) in an attention-grabbing manner. In exploratory tasks, a wide flexibility is useful to enable browsing-like undirected exploration. Finally, many routine tasks involve visual reasoning, not the least important are software assistants for medical

diagnosis. For routine tasks, flexibility should be deliberately restricted or at least hidden behind some "Advance options..." sheet for expert users only. Not only reduced parameter sets but also guided (wizard like) interaction to predefined steps are appropriate to enable users to build habits. The incorporation of analytic capabilities is essential for these three kinds of basic tasks. In exploratory tasks, often involving huge and multi-dimensional data, analytic and aggregating capabilities are essential to cope with the large amounts of data at all. In supervisory tasks, analytic functions may enable not only the localization of a potential failure but also the classification of the problem, the analysis of its severity and potential sources. Finally, in routine tasks, users may be directed to suspicious features. The development of effective tools in any of these task categories requires a model of the task, a model of the user groups as well as a model of visualization options including possible combinations guided by knowledge on human perception.

4 Discussion

The state-of-the-art in visually enabled reasoning has evolved from baby steps (beginning of visualization) to teen years (introduction of interactive visualization) to adulthood (maturing of visualization and interaction research and device technology). We are now approaching the senior years, and we are exploring new ways to use these established technologies for the next evolutionary step, which is visually enabled reasoning.

This chapter discusses challenges and shortcomings of current systems, proposals for improved interactive systems that take human cognition into account, and the potential need for a new science of interaction with information.

4.1 Current Shortcomings in Interactive Visualization Systems

The word *visualization* refers to the process of creating a mental understanding and notion of an object or phenomenon of which information is conveyed to the mind through our sensory channels of perception. Sometimes this is simply referred to as *insight*. Vision is just one of the sensory channels that can be used, but as it is in many aspects the dominating one it has lent its name to the creation of the mental notion of perceived subjects. Visualization is therefore in many cases facilitated by computer rendered images and relies on the power of the human visual sense to analyze the content of images. It should be noted that the mental process of visualization does not only rely on vision, but also makes use of our other senses. The reason for the dominance of the visual channel is found in the high information bandwidth that it creates to the human brain, as we are fundamentally visual beings. This is also reflected in the semantic metaphors we use to describe understanding — "I see", "To Visualize" or indeed "Insight". The inclusion of other senses in the visualization process is sometimes referred to as perceptualization, which emphasizes the collaboration of all human senses. This is a somewhat confusing use of the word and could also be misleading as it points more to the creation of the impression rather than the insight gained.

The apparent success of visualization based on the visual sense is only a part of the reason why other senses have been less explored in visualization. It can be argued that in comparison with vision other senses are more qualitative, less precise and harder to render input for. Also the equipment is expensive and the bandwidth low compared to vision. Despite this there have been many efforts to develop multi-sensory visualization. Most of these efforts have, however, not delivered the added value needed to compensate for the effort involved in generating the associated stimuli, and multi-sensory visualization has not been given high priority in the visualization community. Furthermore, immature implementations

and demonstrators have deterred many users from exploiting the potential of multi-sensory visualization.

There is, however, renewed interest in multi-sensory visualization. Rendering is now reaching a very high level of quality and to be able to bring visualization to the next level, addressing larger and more complex data, more information channels need to be utilized. It has also been shown to that, for instance, haptic interfaces can improve the process of visualization and shorten time to insight. To fully explore the use of multi-sensory visualization a rigorous understanding of what additional sensory channels can contribute, which methods are really effective, and for which applications the additional information channels can provide key information is needed. Used in an effective way it is deemed probable that the multi-sensory visualization can improve the visualization process as much or more than further improvement of rendering of images.

People tend to use both hands if they manipulate 3-D objects [25][40]. In medicine, two-handed interaction has been successfully applied, e.g., to pre-operative planning in neurosurgery. Hinckley et al. [27] argue that for the interaction tasks involved, e.g., exploration of a brain with free orientation of head and cutting plane, the most intuitive handling can be achieved with two-handed 3-D interaction where the dominant hand does fine-positioning relative to the non-dominant hand. In an empirical evaluation they demonstrated that physicians use these interaction techniques efficiently after only a short learning period.

4.2 Improved System Designs

The new science of interaction enabling visual reasoning as experience within visual analytics technologies and systems requires interfaces and interaction something specific of the data type and sometimes specific to the application. For example IN-SPIRE is a suite of technologies designed for unstructured text analytics [1]. The interlinking multiple visual representations, lists, multiple query types, temporal, affect, and other visual representation were deigned to effective text analytics. It applies to a wide range of applications from news, reports, blogs, planning documents, science articles, and many more. Some of the foundational interactions techniques are used with other data types but the implementations of the interactions are quite different such as those for video analysis of news [18].

Some data types and applications will quire unique interactions to ease the cognitive burden between the users, often non computer specialists, and their information. Such an interface can be seen in recent financial analytics system built specifically for fraud detection [10]. The many cyber applications based on billions of transactions are another call of application specific interfaces. Our thinking for a new science of interaction with visual analytics systems must include both the foundational interactions and some application and data type specific applications.

4.3 Cognitive Aspects of Reasoning

Alan Newell posited ”bands” of mental activity ranging from biological-level neural firings over 10 ms and below, cognitive operations that take place on the order of seconds, rational activities that take place on the order of minutes and so on.

What Newell did not explicitly consider were that many temporal constraints on cognitive processing are due to the cost of acquiring information from the environment through motor activity. At lowest level eye and head movements strategically (albeit unconsciously) sample the visual world so as to support processes of perceptual inference discussed above. Newer cognitive science research suggests, the time required to execute an eye movement to acquire

needed information constrains the speed of cognitive processing [2]. In reading, for example, the processing time of a fixated word is slowed so as to enable the eye to have the time to make a saccade to the next word in the sentence. Given the hard constraint of the time required to make an eye movement, this "just-in-time" cognitive processing reduces the load on short-term memory in reading.

The correspondence of eye movement and processing times is characteristic of many perceptuomotor "interactive routines". These routines comprise epistemic actions that reveal information, externalizing actions that modify the perceptual world to reflect conceptual understanding, and coordinating actions that bind concepts to content. In the case of expert performers (e.g. skilled musicians or very experienced computer users) these interactive routines are effectively "complied", taking place automatically under supervisory control of conscious problem solving processes.

This line of investigation has significant implications for visualization applications that support human reasoning. Wayne Gray's "soft constraints" cognitive cost accounting hypothesis [21][55] posits that small changes in the time required for the user to acquire information from a visual display can impact information comprehension and discourse and cause significant shifts in task performance and strategy. Work by Po et al. [36] demonstrated that presence of a cursor and delay in its response had profound impacts on users' ability to target display items via voice and pointing. The effects of these small changes in display response was attributed a shift between dorsal and ventral visual pathways.

If these theories are correct, interactions of temporal patterns in human-information dialog can interact with the intrinsic time course of cognitive processes to support or impede cognitive processing. This suggests that temporal rhythms in solo and collaborative use of technology can both detect and support "flow" [15] of effective cognitive processing and fluency of interaction. Addressing the sequential nature of human-information dialog will require new empirical methods that integrate mathematical modeling of sequences of interaction and human-mediated qualitative research methods.

Many of us are now working within research projects that involve user evaluations of applications and visualization tools at different points throughout the development process. Unfortunately we see a tendency in our community to rely almost exclusively on quantitative methods of evaluation, often associated directly with theories of cognition and sometimes taken without question or modification from the fields of HCI and cognitive science. While these fields are naturally close to our work and collaborations with HCI and cognitive science are and will continue to be very fruitful in our research and in evaluations, we would also like to point out that there are other approaches to user evaluation that can shed light on elements of visualization which will also be very useful for future work. Inspiration to this more varied and qualitative research on usability draws its inspiration from the groundbreaking study of human-machine interaction 'Plans and Situated Action' by Suchman [49][50]. Examples of qualitative methods that we suggest could be useful additions to our projects include:

Interviews with users — both interviews which follow a preset interview guide and open interviews (which allow the user to speak more freely about her/his experience of the tool) can be helpful not only in analyzing how well the user has succeeded in interpreting the data we are presenting, but also in discovering otherwise unknown issues with the tools. More importantly, this method can sometimes uncover questions to and about the data that are important to users but which were not specified in the research project's original remit. Often this is a result of the fact that 'the user' is generally a more heterogeneous category than we imagine it to be. Interviews with several different users can show how this heterogeneity impacts and is impacted by our visualization tools.

Discourse analysis — by this we suggest that it is sometimes useful for us to analyze the discourses which surround that data that we are attempting to visualize. Rather than relying solely on official documentation to describe the data which is provided to or included in our research projects, we suggest that analyzing more unofficial discourses (sourced through searches of academic, trade and news reports, for example) that surround a data set can also shed light both on alternative interpretations of the data and on other possible users, whose needs and impressions should also be considered when we are constructing tools with which to visualize data.

Observation of the tools in use — while this type of method can be used to make quantitative analyses of a tool (by counting eye movements or timing task completion, for example), we suggest that qualitative observations, based on the ethnographic methods like those employed within the field of Science, Technology and Society could also lead us to understand the way people actually use and, perhaps more importantly, 'misuse' our tools, forcing us to see our work in entirely different contexts. Much of this observation could and should be done outside of the laboratory, in 'real world' settings, to give us a better feel for how our research behaves and is experienced once it leaves our hands.

The emphasis on visual display and human visual processing in the literature reflects our understanding that the visual modality is central to human experience and the most likely candidate modality for effective technological enhancement of human reasoning. Behind this assumption lies a concern that adding finely-textured information from other senses — hearing and touch — will distract from the information better presented in the dominant sensory domain. We find this assumption unwarranted. Cognitive neuroscientists have long known that vision itself comprises multiple sensory channels with integration of those channels occurring in higher visual areas that must draw from multiple neural maps of smaller sensory channels such as color, shape, orientation, etc. Many of these areas also draw from sensory channels from hearing and touch.

4.4 Do We Need a New Science of Human Interaction with Information?

It is clear that the study of interaction must be significantly expanded and deepened from what has been done so far. How interaction works with visual display really hasn't been studied in any depth. Yet, it is clear with the new highly interactive exploratory tools that are now being developed that interaction is a very important part of the total system, even if it is not well understood. It is with this and the needs for visually-enabled reasoning in mind that the developers of the visual analytics research agenda called for a new "science of interaction" [51]. This call has been reiterated and progress so far on the science of interaction detailed recently [35][39].

4.5 Unexpected Discovieries

In the discussion of visual reasoning tasks, three categories were defined: exploratory, supervisory, and routine. Of these three, exploration is the most demanding in terms of reasoning, because it involves a process where the investigator does not know what she is looking for (at least not in detail). Hence, the process is one where discovery is emphasized. Once the investigator makes a discovery, she must assess what it means and how it fits into the context of what she already knows. Often a model or argument, which we call here the hypothesis, must be formed that weaves together known facts, the newly discovered evidence, the task at hand, and other relevant knowledge. The hypothesis has predictive

capability and is testable; new evidence must be collected to validate it. There are often multiple competing hypotheses, so one must gather further evidence that lends support to one or the other. Finally, the hypotheses may need to be modified based on new evidence collected or if there is a dynamic situation where circumstances change over time. All these processes are reasoning processes and can be complex and iterative. If, further, one has both complex reasoning *and* large scale, dynamic data with perhaps many related variables, then this is a visual reasoning task requiring visual analyses.

In the visual reasoning process, interaction is key for two reasons. First, exploration implies probing the data in overview, in relations within the data, and in detail. It is only by doing this that discoveries can be made. Second, exploration and discovery must intimately involve the investigator since only she can determine the context, meaning, and relations of the discoveries made. In particular, there cannot be an automated analysis that will extract meaning and relations because the nature of the discovery is not known beforehand and thus cannot be planned for. Therefore, interaction is key, and it must furthermore be through a visual interface.

We will give examples for the class of problems, involving both complex reasoning and large scale data, for which visual reasoning through an interactive visual interface is required. Certainly problems in bioinformatics are of this type. Researchers often have to compare and contrast annotated genomic data for several species, which includes large amounts of information such as relevant publications, various statistical analyses including of the sequences themselves, microarray results, protein expression results, and other information. Often a gene function can only be fully understood as part of a gene network, such as regulatory networks that suppress or enhance functions associated with specific diseases. These are complex reasoning processes involving exploratory analysis for which visualizations are required.

Another example is integrated computer experiments involving weather and environmental effects. Weather drives all these computer experiments (e.g., wind patterns, clouds, rain patterns, etc.), and to get high resolution results, which are necessary for detailed environmental impact studies, one must start with high resolution weather. The weather inputs then drive high resolution air quality models with hundreds of time-dependent output fields involving interacting chemical constituents and items such as various types of particulates. The results are truly stupendous in size and enormously complex in their relations and interactions. Two- and three-dimensional visualization techniques have been used for some time, but to study and understand the interacting 4D fields in the context of factors such as changing pollution sources, population and traffic patterns, and other factors requires visually-enabled reasoning. Without visually-enabled reasoning, there is no hope of understanding these complex processes. General, complex problems that have many applications include exploratory analysis and understanding of large scale collections of text or multimedia. Large text collections, of course, appear in many contexts from the above bioinformatics problems to business intelligence and legal evidence-gathering for large scale civil or criminal investigations. An example of the latter is the ENRON fraud investigation, which involved the sifting of billions of documents of all types, from email exchanges to internal memos and reports, by teams of lawyers. Visual analysis tools such as INSPIRE, developed by PNNL, have proved quite successful for these types of analyses, which cannot be done in such detail by any other method. In it latest versions, INSPIRE is also an example of a tool that is being enhanced to more fully support visually-enabled reasoning. Large scale multimedia collections (containing related images, text, video, closed captions, etc.) are also notoriously difficult to analyze. Unannotated image collections, for example, must

be sorted by hand, and even this categorization will not provide a view of all the relations one might need for an exploratory analysis. Likewise, video collections must be watched, even if annotated at some level, in order to understand, analyze in detail, and relate their contents. For collections containing millions or billions of images (such as collections that can be gleaned from the Web) or tens of thousands of hours of video (such as produced by broadcast news in a period of days), the task is unmanageable. Now exploratory visual analysis tools have been developed that effectively attack both these problems and point the way towards full visually-enabled reasoning capabilities.

The development so far of these tools for exploration and discovery indicates that they will also be quite useful and effective for more modes problems in both size and complexity. It is our position that visually-enabled reasoning tools with high interactivity will be of great use whenever one is faced with an open-ended problem involving the meaning of data or information.

These issues of exploration, discovery, and the role of interaction are, of course, also central issues in visual analytics, which is most succinctly described as "the science of analytical reasoning facilitated by interactive visual interfaces". But whether one approaches visually-enabled reasoning and the science of interaction from the viewpoint of visual analytics or from another direction (e.g., scientific visualization and computational science), the basic needs and the science that must result are the same.

5 Success Story

MSU ERC Space Shuttle Study — In 1990, the National Science Foundation established an Engineering Research Center (ERC) for Computational Field Simulation at Mississippi State University (MSU). The fulfillment of the center's mission is illustrated by the John Glenn space shuttle flight. The center has significantly contributed to the art and practice of "unstructured grid generation", yielding high quality grids in significantly less time. The center focused a team on coupling its structured grid CFD algorithm knowledge within a portable, scalable computational architecture onto unstructured grid solver technology. This required substantial research in both boundary layer gridding and solution algorithms. As it turned out, the parallel solver (research) code had just been assembled for the first time when the Space Shuttle mission STS-95 was launched. NASA Johnson Space Center called seeking simulated analysis of the Space Shuttle Orbiter during the return flight after the Orbiter drag chute door was lost during main engine startup. The NASA engineers wanted to know the dynamic pressure in the region of the missing chute door in order to estimate the aerodynamic loadings during reentry. The ERC group read a previously supplied Space Shuttle Orbiter geometry into the ERC's integrated simulation environment (SOLSTICE) and created the grids within hours. Initial simulation results were computed on a high performance computer within two days. The significance of this endeavor was not that NASA actually needed the results for successful reentry, but rather that the ERC had been able to take a tough real world problem and compute the solutions in two to three days after receiving the geometry description. This demonstrated an achievement that was a direct result of the researchers' ability to simulate very complex real world problems with complex geometries in relative motion. These accomplishments have come from directed cross-disciplinary efforts involving various technologies: grid generation, field solution algorithms, and scientific visualization, coupling human reasoning with computer and computational engineering. The task could not have been accomplished without combining all of the various talents and technologies.
(Source: NSF Engineering Research Center at Mississippi State University)

Acknowledgements

The authors would like to thank Lucille Nowell, who is with the United States Department of Energy, for her valuable suggestions and encouragement. The authors would also like to thank David S. Ebert, Hans Hagen, Kenneth I. Joy, and Daniel A. Keim, the organizers of Dagstuhl (Leibniz Center for Computer Science, Germany) Seminar 07291 on *Scientific Visualization*, Wolfgang Lorenz, Technical Administrative Director, and Reinhard Wilhelm, Scientific Director.

References

1 IN-SPIRE. `http://in-spire.pnl.gov` (accessed March 2, 2010).

2 D. H. Ballard, M. M. Hayhoe, P. K. Pook, and R. P. Rao. Deictic codes for the embodiment of cognition. *The Behavioral and Brain Sciences*, 20(4):723–742, 1997.

3 Benjamin B. Bederson, James D. Hollan, Ken Perlin, Jonathan Meyer, David Bacon, and George Furnas. Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. *Journal of Visual Languages and Computing*, 7(1):3–31, 1996.

4 Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: the see-through interface. In *SIGGRAPH Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 73–80, New York, NY, 1993. ACM. ISBN 0-89791-601-8.

5 S. Brennan, K. Mueller, G. Zelinsky, Ramakrishnan IV, D. Warren, and A. Kaufman. Toward a multi-analyst, collaborative framework for visual analytics. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 129–136, Baltimore, MD, October 2006.

6 P. Brusilovsky, A. Kobsa, and J. Vassileva. *Adaptive Hypertext and Hypermedia*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1998.

7 Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction: Special Issue on Adaptive Hypertext and Hypermedia*, 6(2–3):87–129, 1996.

8 Yang Cai and Judith D. Terrill. Visual analysis of human dynamics: An introduction to the special issue. *Information Visualization Journal*, 5:235–236, 2006.

9 S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, San Diego, CA, 1999.

10 R. Chang, M. Ghoniem, R. Kosara, W. Ribarsky, J. Yang, E. Suman, C. Ziemkiewicz, and A. Sudijianto. WireVis: Visualization of categorical, time-varying data from financial transactions. In *IEEE Symp. on Visual Analytics Science and Technology (VAST)*, Sacramento, CA, Oct. 30–Nov. 1 2007.

11 R. Chang, C. Ziemkiewicz, T. Green, and W. Ribarsky. Defining insight for visual analytics. *IEEE Computer Graphics and Applications*, 29(2):14–17, March/April 2009.

12 Mark A. Changizi, Andrew Hsieh, Romi Nijhawan, Ryota Kanai, and Shinsuke Shimojo. Perceiving the present and a systematization of illusions. *Cognitive Science: A Multidisciplinary Journal*, 32(3):459–503, 2008.

13 M. C. Chuah and S. F. Roth. On the semantics of interactive visualizations. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 29–36, San Francisco, CA, 1996.

14 H. H. Clark and S. E. Brennan. Grounding in communication. In L. B. Resnick, J. Levine, and S. D. Teasley, editors, *Perspectives on socially shared cognition*, pages 127–ñ149. Washington, DC, 1991. Reprinted in Groupware and computer-supported cooperative work: Assisting human-human collaboration, R. M. Baecker (ed.), Morgan Kaufman Publishers, Inc., San Mateo, CA, pp. 222ñ-233, (1992).

15 M. Csikszentmihalyi. *Flow*. Simon & Schuster, 1994.

16 Stephen G. Eick and Graham J. Wills. High interaction graphics. *European Journal of Operations Research*, 81(3):445–459, March 1995.

17 B. Fisher, S. Fels, K. MacLean, T. Munzner, and R. Rensink. Seeing, hearing, and touching: putting it all together. In *ACM SIGGRAPH 2004 Course Notes*, page 8, Los Angeles, CA, August 08–12 2004. SIGGRAPH, ACM Press, New York, NY.

18 Mohammad Ghoniem, Dongning Luo, Jing Yang, and William Ribarsky. Newslab: Exploratory Broadcast News Video Analysis. In *IEEE Symp. on Visual Analytics Science and Technology (VAST)*, pages 123–130, Sacramento, CA, Oct. 30–Nov. 1 2007.

19 J. Giesen, K. Mueller, E. Schuberth, L. Wang, and P. Zolliker. Conjoint analysis to measure the perceived quality in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1664–1671, 2007.

20 M. Good. Participatory design of a portable torque-feedback device. In *Proceedings of the ACM Conference on Human Factors*, Computing Systems, pages 439–446, New York, NY, 1992. ACM Press.

21 W. D. Gray and W. Fu. Soft constraints in interactive behavior: The case of ignoring perfect knowledge in-the-world for imperfect knowledge in-the-head. *Cognitive Science*, 28(3):359–382, 2004.

22 Tera Green and William Ribarsky. Using a human cognition model in the creation of collaborative knowledge visualizations. In *Proceedings of SPIE Defense & Security Conference, vol. 6983*, pages C1–C10, 2008.

23 Tera Green, William Ribarsky, and Brian Fisher. Building and applying a human cognition model for visual analytics. *Information Visualization Journal*, 8(1):1–13, 2009.

24 Jonathan Grudin. Why groupware applications fail: Problems in design and evaluation, Office: Technology and People. 4(3):245ñ–264, 1989.

25 Y. Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, 19(4):486–517, 1987.

26 A. Gustafsson, A. Herrmann, and F. Huber. *Conjoint analysis as an instrument of market research practice*, pages 5ñ–45. Conjoint Measurement. Methods and Applications. 2000.

27 K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell. Passive real-world interface props for neurosurgical visualization. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 452–458, Boston, MA, 1994.

28 Daniel Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 8(1):1–8, January–March 2002.

29 A. Kobsa and J. Fink. An LDAP-based user modeling server and its evaluation. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, 16(2):129–169, 2006.

30 A. Kobsa and W. Wahlster. Computational linguistics. *Special Issue on User Modeling*, 14(3), 1988.

31 A. Kobsa and W. Wahlster. *User Models in Dialog Systems*. Symbolic Computation. Springer, New York, etc., 1989.

32 Bill Kules. User modeling for adaptive and adaptable software systems, `http://www.otal.umd.edu/UUGuide/wmk/` (accessed May 18, 2009), University of Maryland, College Park, MD, April 19 2000.

33 Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

34 C. North. Toward measuring visualization insight. *IEEE Computer Graphics and Applications*, 26(3):6–9, 2006.

35 W. A. Pike. Science of interaction. *Information Visualization Journal, Special Issue on The Future of Visual Analytics*, 8(4), 2009.

36 B. Po, B. Fisher, and Booth K. S. Pointing and visual feedback for spatial interaction in large-screen display environments. In A. Butz, A. Krüger, and P. Olivier, editors, *Smart Graphics 2003*, Springer Lecture Notes in Computer Science, Heidelberg, Germany, 2003. Lexicle Ltd, York, UK.

37 Z. Pylyshyn. Is vision continuous with cognition? The case for cognitive impenetrability of visual perception. *The Behavioral and Brain Sciences*, 22(3):341–365, discussion 366–423, 1999.

38 Jef Raskin. *The Intelligent User Interface*. Addison-Wesley, 2001.

39 William Ribarsky, Brian Fisher, A. E. Turner, and William M. Pottenger. The science of analytic methods and reasoning. *Information Visualization Journal*, 2009. To be published.

40 Felix Ritter, Bernhard Preim, Oliver Deussen, and Thomas. Strothotte. Using a 3d puzzle as a metaphor for learning spatial relations. In *Proceedings of Graphics Interface*, pages 171–178, Montreal, Canada, May 15–17 2000. Morgan Kaufmann Publishers.

41 I. Rock. *The logic of perception*. MIT Press, Cambridge, MA, 1983. ISBN 0-2621-8109-6.

42 I. Rock. *Indirect perception*. MIT Press, Cambridge, MA, 1997. ISBN 0-2621-8177-0.

43 Mary Beth Rosson and John Millar Carroll. *Usability Engineering*. Morgan Kaufmann, 2002.

44 P. Saraiya, C. North, and K. Duca. An insight-based methodology for evaluating bioinformatics visualizations. 11(4):443–456, 2005.

45 M. Schneider-Hufschmidt, T. Kühme, and U. Malinowski. *Adaptive user interfaces: Principles and Practice*. Amsterdam: North-Holland, 1993.

46 Roger N. Shepard. Recognition memory for words, sentences, and pictures. *Journal of Verbal Learning and Behavior*, 6:156–163, 1967.

47 B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages*, pages 336–343, Boulder, CO, 1996.

48 Lionel Standing. Learning 10,000 pictures. *Quarterly Journal of Experimental Psychology*, 25(2):207–222, 1973.

49 Lucy Suchman. *Plans and Situated Action. The Problem of Human-Machine Communication*. Cambridge University Press, NY, 1987.

50 Lucy Suchman. *Human-Machine Reconfigurations*. Cambridge University Press, NY, 2007.

51 J. Thomas and K. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Center. IEEE CS Press, 2005.

52 E. R. Tufte. *Envisioning information*. Graphics Press, 1990.

53 L. A. Tweedie, R. Spence, D. Williams, and R. Bhogal. The attribute explorer. In *Proceedings of the Video Track of the ACM Conference on Human Factors in Computing Systems*, pages 435–436, Boston, MA, April 1994.

54 R. van Liere, Mulder J. D., and J. J. van Wijk R. Computational steering. *Future Generation Computer Systems*, 12(5):41–450, 1997.

55 B. Z. Veksler, W. D. Gray, and M. J. Schoelles. From 1000 ms to 650 ms: Why interleaving, soft constraints, and milliseconds matter. In *Proceedings of the 8th International Conference on Cognitive Modeling*, Ann Arbor, MN, 2007.

56 J. S. Yi, Y. ah Kang, J. T. Stasko, and J. A. Jacko. Toward a deeper understanding of the role of interaction in information visualization. 13(6):1224–1231, November/December 2007.

57 J. S. Yi, Y. ah Kang, J. T. Stasko, and J. A. Jacko. Understanding and characterizing insights: How do people gain insights using information visualization? In *BELIV: Proceedings of the 2008 Conference on Beyond Time and Errors*, pages 1–6, New York, NY, 2008. ACM.

Visual Simulation of Flow

Arie Kaufman[1] **and Ye Zhao**[1]

1 **Department of Computer Science**
Stony Brook University
Stony Brook, NY 11794, USA
{ari,yezhao}@cs.sunysb.edu

Abstract

We have adopted a numerical method from computational fluid dynamics, the Lattice Boltzmann Method (LBM), for real-time simulation and visualization of flow and amorphous phenomena, such as clouds, smoke, fire, haze, dust, radioactive plumes, and air-borne biological or chemical agents. Unlike other approaches, LBM discretizes the micro-physics of local interactions and can handle very complex boundary conditions, such as deep urban canyons, curved walls, indoors, and dynamic boundaries of moving objects. Due to its discrete nature, LBM lends itself to multi-resolution approaches, and its computational pattern, which is similar to cellular automata, is easily parallelizable. We have accelerated LBM on commodity graphics processing units (GPUs), achieving real-time or even accelerated real-time on a single GPU or on a GPU cluster. We have implemented a 3D urban navigation system and applied it in New York City with real-time live sensor data. In addition to a pivotal application in simulation of airborne contaminants in urban environments, this approach will enable the development of other superior prediction simulation capabilities, computer graphics and games, and a novel technology for computational science and engineering.

1998 ACM Subject Classification I.3.1 Hardware Architecture, I.3.8 Applications, J.2 Physical Sciences and Engineering

Keywords and phrases Lattice Boltzmann Method, Amorphous phenomena, GPU Acceleration, Computational Fluid Dynamics, Urban Security

Digital Object Identifier 10.4230/DFU.SciViz.2010.246

1 Introduction

Visually reproducing flow phenomena in all its richness and complexity has been an exciting endeavor in computer graphics and visualization. The purpose can be categorized into two main topics. One is to enrich the virtual environment with photorealistic images and animations of natural scenery. We have seen astounding appearances in the movies and games of streaming water, flaming fire, turbulent smoke, and so on. In these applications, the requirement is to visually convince the observers that they have seen duplicated reality. The other topic is the correct flow simulation, which demands the accuracy and precision besides the visual authenticity. Such simulations have a broader scope of applications in scientific, environmental and security areas with its reasonable behavior replication and believable picture making.

We have now entered a new era in computer graphics with the advent of hardware accelerated programmable rendering and shading. With the programmability of the graphics processing unit (GPU), combined with the increased performance of CPUs, we can now start to simulate flow phenomena at interactive rates. Direct computational fluid dynamics (CFD)

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 246–258

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

solvers of the complex Navier-Stokes (NS) equations have been introduced to benefit the graphics and visualization applications.

Procedure techniques have been applied for many years to model gaseous phenomena. They [4, 3, 27, 29, 20, 13] use mathematical functions and algorithms to define the appearance of the objects. Particle-based Methods [26, 23, 25, 19, 9, 28] have also been widely used in amorphous phenomena modeling because they are computationally inexpensive, flexible to control the behaviors and easy to fit into the user-interaction paradigm.

Over the past decades, the application of CFD methods for solving the NS equations has led to significant advances in the modeling of fluid phenomena. Foster and Metaxas [10, 11] developed physically-based methods for the realistic animation of fluids by solving the NS equations. Based on a stable fluid solver [30], various realistically looking smoke [8], water [5], flame [21], viscoelastic fluids [12] and flows on surfaces [31] were generated.

A promising and relatively new CFD method, Lattice Boltzmann Method (LBM), has been introduced by our group to the graphics and visualization community in 2002 [33]. Extensive research has been conducted on using the LBM to model various flow phenomena [34, 1, 7, 24, 35, 36, 40, 39]. The LBM solves the fluid dynamics within the framework of statistical mechanics, where the microscopic physics of fluid particles are modelled and the macroscopic averaged properties obey the desired NS equations. Its microscopic kinetic equation provides many advantages, including easy implementation of boundary conditions and fully parallel algorithms. The LBM can handle any arbitrarily-shaped inside objects and even dynamic boundaries of moving objects. The computation of the LBM is inherently local and explicitly parallel, which is easy to accelerate on the modern graphics processing unit (GPU) to achieve great simulation performance. Meanwhile, multi-resolution LBM approaches have been applied to optimize the use of computational resources for large-scale simulations. The LBM has been successfully used in modeling various amorphous phenomena and simulating contaminants dispersion in urban environments. It has the great potential to benefit researchers and end-users in a variety of applications in education, entertainment and scientific simulations.

2 Lattice Boltzmann Method

The LBM is a relatively new approach in CFD, inspired by cellular automata, that models Boltzmann particle dynamics on a lattice. In the case of a fluid, for example, the Boltzmann equation expresses how the average number of flow particles with a given velocity changes between neighboring lattice sites due to inter-particle interactions and ballistic motion. In the LBM, the variables associated with each lattice site are the particle velocity distributions that represent the probability of the presence of a flow particle with a given velocity. The set of velocities in the model is discrete, being defined by the number, orientation, and length of lattice links. Particles stream between neighboring sites synchronously in discrete time steps. Between streaming steps, they undergo collision. The Bhatnager, Gross, Krook (BGK) model [32] is commonly employed to represent the collisions as a statistical redistribution of momentum, which locally drives the system toward equilibrium. As in kinetic theory, the collisions conserve mass and momentum.

The LBM simulation of 3D flow field is generally performed on a 3D lattice where each lattice site has a number of links representing the velocity vectors (including the zero velocity) to its neighbors. As illustrated in Figure 1, this lattice cell is part of a 3D lattice called D3Q13, which includes the center site with zero velocity and the twelve minor-diagonal neighbor links. Stored at each lattice site are 13 particle distributions associated with the

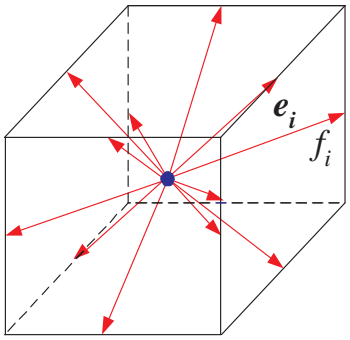


Figure 1 The D3Q13 lattice geometry. The particle distribution f_i is associated with the link corresponding to the $\mathbf{e}_i$ velocity vector.

13 velocity vectors. We denote these as f_i where i identifies a particular velocity vector $\mathbf{e_i}$ among the 13. The macroscopic flow density, ρ, and the macroscopic flow velocity, $\mathbf{u}$, are computed from the particle distributions:

$$\rho(\mathbf{r},t) = \sum_i f_i(\mathbf{r},t) \tag{1}$$

$$\mathbf{u}(\mathbf{r},t) = \frac{1}{\rho(\mathbf{r},t)} \sum_i f_i(\mathbf{r},t)\mathbf{e}_i \tag{2}$$

Using the BGK model, the Boltzmann dynamics can be represented as a two-step process of collision and ballistic streaming:

$$f_i(\mathbf{r},t^+) = f_i(\mathbf{r},t) - \frac{1}{\tau}(f_i(\mathbf{r},t) - f_i^{eq}(\mathbf{r},t)) \tag{3}$$

$$f_i(\mathbf{r}+\mathbf{e}_i,t+1) = f_i(\mathbf{r},t^+) \tag{4}$$

Note that we use the notation t^+ to denote the post-collision particle distribution. Also, f_i^{eq} represents the local equilibrium particle distribution, which is given by

$$f_i^{eq}(\rho,\mathbf{u}) = \rho(A + B(\mathbf{e}_i \cdot \mathbf{u}) + C(\mathbf{e}_i \cdot \mathbf{u})^2 + D\mathbf{u}^2). \tag{5}$$

The constant τ represents the relaxation time scale that determines the viscosity of the flow, while A through D are constant coefficients specific to the chosen lattice geometry. The equilibrium particle distribution comes in as a consequence of the BGK collision model. It is a local particle distribution whose value depends only on conserved quantities - the macroscopic mass ρ and momentum $\rho\mathbf{u}$. Its form may be recognized as the Taylor expansion of the 3D Maxwell velocity distribution to second order. Since the evaluation of Equations 1 through 5 requires only local particle distributions, their acceleration on graphics hardware is efficient (see Section 3). Only one parameter, τ, is used to control the flow behavior in collision (Equation 3). Therefore, this primary LBM is called the single-relaxation-time LBM (SRTLBM). A Smagorinsky subgrid model [36] can be applied to achieve high Reynolds numbers flows without incurring numerical instability, which only involves local particle distribution values and retains the LBM parallelizability.

2.1 Multiple-relaxation-time LBM

Even with the subgrid method, the SRTLBM is prone to unstable numerical computation when used for low viscosity fluids (high turbulent fluids) or incorporated with temperatures or body forces. Multiple-relaxation-time Lattice Boltzmann Model (MRTLBM) is a new general version of LBM developed by d'Humiéres et al. [2]. This collision model abandons SRTLBM to achieve better numerical stability and greater flexibility in selecting the transport coefficients. The essential idea is to make a change of basis from phase space (i.e., the space of the distributions f_i) to the space of hydrodynamic moments (i.e., density, momentum, energy, etc.) and to perform the collision step in the latter space. As in the BGK model, collisions are implemented via a relaxation, but in the moment space each moment is allowed to relax individually. Although the relaxation rates are not all independent, the additional flexibility allows one to maneuver the model into regions of higher stability while decoupling some of the transport coefficients. After relaxation, the inverse transformation is applied to return to phase space where streaming, boundary update rules, and additional micro-physics are implemented as before.

Mathematically, the change of basis from the space of distributions to the space of moments is given by:

$$|m\rangle = M|f\rangle, \qquad |f\rangle = M^{-1}|m\rangle \tag{6}$$

$$|f\rangle = (f_0, f_1, \ldots, f_{12})^T, \tag{7}$$

$$|m\rangle = (m_0, m_1, \ldots, m_{12})^T, \tag{8}$$

where T denotes the transpose. Each of the 13 moments $\{m_i|(i = 0, 1, \ldots, 12)\}$ has a physical meaning. For example, m_0 is the mass density ρ, $m_{1,2,3}$ are the components of the momentum vector $\mathbf{j}$, m_4 is the energy, and the other higher order moments are components of the stress tensor and other high order tensors. The rows of the matrix M relate the distributions to the moments. For example, since $\rho = \sum_i f_i$, the first row of M consists of all ones. Although the values of the distributions and the moments vary over the nodes of the lattice, the matrix M is constant for a given lattice.

In MRTLBM, the two step process of collision and streaming becomes:

$$|f(\mathbf{r}, t^+)\rangle = |f(\mathbf{r}, t)\rangle - M^{-1}S[|m(\mathbf{r}, t)\rangle - |m^{eq}(\mathbf{r}, t)\rangle] \tag{9}$$

$$|f(\mathbf{r} + \mathbf{e_i}, t + 1)\rangle = |f(\mathbf{r}, t^+)\rangle \tag{10}$$

The components of the vector $|m^{eq}\rangle$ are the local equilibrium values of the moments. Among them, the mass density and the momentum (m_0 to m_4) are conserved. Expressions for the nonconserved moments depend only on local values of the conserved moments [15]. The matrix S in the collision equation is a diagonal matrix whose elements are the relaxation rates, $\{s_i|(i = 0, 1, \ldots, 12)\}$. Their values are directly related to the kinematic shear and bulk viscosities, ν and ξ, respectively:

$$\nu = \frac{1}{4}(\frac{1}{s_6} - \frac{1}{2}), \tag{11}$$

$$\xi = (\frac{2}{3} - \gamma c_{s0}^2)(\frac{1}{s_5} - \frac{1}{2}), \tag{12}$$

where γ is the specific heat and c_{s0} is the isothermal speed of sound. The user has the freedom to choose the flow parameters to define characteristics of the fluid being modeled. This choice then determines the relaxation rates.

MRTLBM can also accommodate a body force due to gravity, sensor readings or some other external field. This is implemented by adding the force $\mathbf{F}$ to the momentum, $\mathbf{j}' = \mathbf{j} + \mathbf{F}\delta t$

(typically, $\delta t = 1$). In practice, for stability, the force term is executed in two steps, one-half before the relaxation step and one-half after.

To capture thermal effects, temperature is coupled to MRTLBM through the energy moment that the model exposes. For the D3Q13 lattice, the energy equilibrium is modified as follows:

$$(m_4^{eq})' = m_4^{eq} + cT, \tag{13}$$

where the temperature T is coupled with a constant coefficient c. The heat transfer here is modeled separately with a standard diffusion-advection equation, given rise to the LBM:

$$\partial_t T + \mathbf{u} \cdot \nabla T = \kappa \Delta T, \tag{14}$$

where κ is the thermal diffusivity of the fluid.

Note that SRTLBM can be seen as a special case of MRTLBM associated with a specific choice of parameter values in the equilibria of the moments so that only one single relaxation rate, $1/\tau$, remains free.

2.2 Boundary Conditions

Interactions between the LBM flow field and the interacting objects result from the exchange of momentum at their shared boundaries. Generally, there exist two types of boundaries in the LBM: (1) the surrounding walls of the LBM simulating space; (2) internal objects. The treatment of boundary conditions of both the SRTLBM and MRTLBM are handled in the discrete velocity space after a general streaming simulating step.

For the surrounding walls, the boundary conditions are usually treated as periodic, bounce back (forward), or out flow boundaries [32]. For the inside objects, bounce-back rule can be easily applied. For no-slip boundaries that can move, or involve complex geometries, the bounce back rule has been substantially improved [35, 18] to accommodate curved and moving boundaries.

2.3 Multi-resolution LBM

Physically-based flow modeling methods usually employ a uniform grid to discretize the simulation domain, and then apply numerical computations to solve the NS equations. For large-scale simulations, it is inefficient to maintain a uniform grid with high resolution spanning the entire domain. To achieve interactive performance and to optimize the use of resources, we have applied a multi-resolution LBM [38] that offers high resolution computation around areas of interest (for example, near a solid body) and places low resolution grids on other areas or faraway boundaries. Interfaces between the grids with different resolutions are properly treated to satisfy the continuity of mass and momentum.

This level-of-detail scheme is implemented by a 3D block-based grid structure consisting of a coarse grid and one or more fine grids. The global flow behavior in the whole simulation space is roughly modeled by the LBM simulation on the coarse grid with relatively low consumption of resources. For regions of interest, the LBM computation performs on the fine grids superposed on the coarse one. These grids are implemented as separate blocks instead of tree-style recursive structures. The global simulation on the coarse grid determines the flow properties on interfaces and then defines boundary conditions of the fine grids at each time step. Therefore, the simulation on the fine grids obeys the correct global flow behavior. Meanwhile, it supplies rich visual details and accuracy in the regions of interest by utilizing

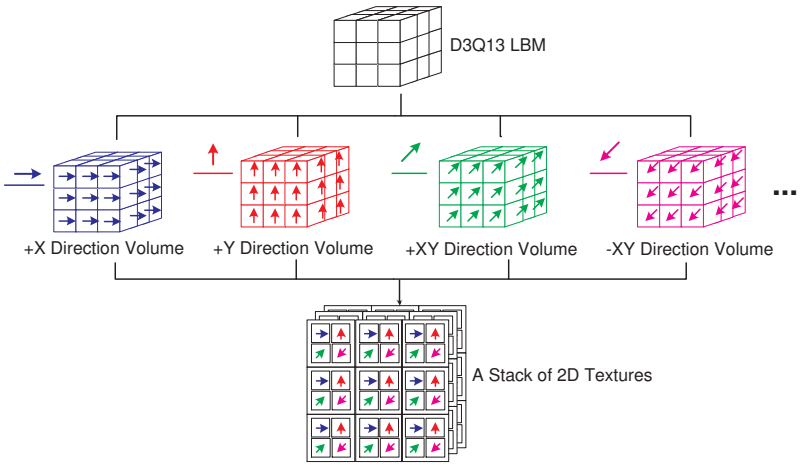


Figure 2 *Each set of particle distributions having the same velocity direction is grouped into a volume and every four volumes are packed into one stack of 2D textures*

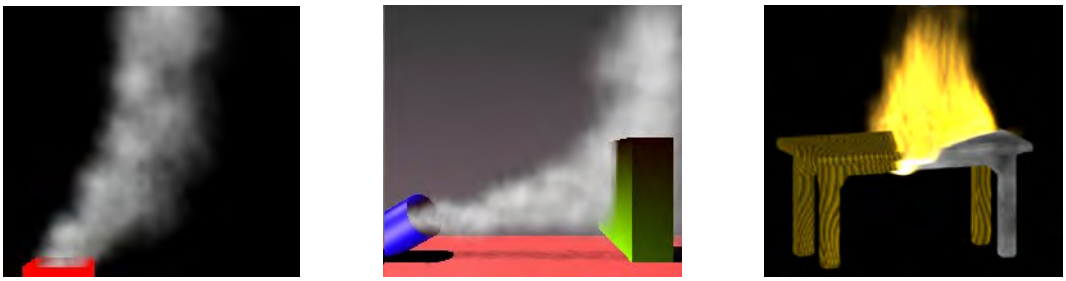
small grid spacing, small time intervals, and introducing vorticity confinement. A fine grid is easily initiated and terminated at any time while the global simulation is running. Moreover, a fine grid is able to move along with a moving object, to model small-scale turbulence caused by the object-fluid interactions.

3 GPU Acceleration

As a direct result of recent advances in modern GPU's multiple-pipeline SIMD architecture, stream processing model, and high memory bandwidth, the computational power of the GPU has outpaced that of the CPU. This gap between their computational powers is foreseen to widen, propelled by the booming game industry. As a result, using the GPUs for general-purpose computation has become very attractive. We refer the reader to a further survey on General Purpose computation on the GPU (GPGPU) [22]. However, note that not all computations can take advantage of the GPU computation. Some requirements, such as data parallelism and the locality in memory access, are essential. Therefore, the implicit fluid solvers cannot be naturally plugged into the GPU due to its obligation to solve a linear system from the Poisson equation of the pressure.

An attractive feature of the LBM is that the computation is inherently local and explicitly parallel. This feature allows us to accelerate the LBM computations on the low-cost SIMD processor of contemporary GPUs, and achieve great performance for the flow phenomena simulation. We have accelerated the flow computation on a GPU [16, 17] with moving boundaries, and extended it for large-scale simulations on a GPU cluster [6].

As shown in Figure 2, all the particle distributions f_i associated with the same velocity direction are grouped into a volume and every four volumes are packed into one stack of textures, since a texel has four RGBA components. The equilibrium distributions f_i^{eq} are stored in the same way. The macroscopic density ρ along with the three components of the velocity u, are stored in one stack of textures similarly. Generally, the simulation of the flow field on the GPU is implemented in several steps: (1) Compute the equilibrium distribution values; (2) Compute collision; (3) Apply the boundary conditions at the boundary links; (4) Stream the particle distributions; (5) Compute the macroscopic density and velocity.



(a) (b) (c)

Figure 3 (a) Smoke coming out of a chimney; (b) Smoke interacting with a green moving obstacle. (c) Fire spreading on a table.

Involving only local operations, each step is implemented as a fragment program that calculates the corresponding equation. The fragment programs fetch the input variables from the textures, execute the computation, and render the output results. Comparing the SRTLBM and the MRTLBM, the hardware acceleration is similar, except that for the MRTLBM, extra matrix multiplications are executed. The matrices used are constant and no inversion is required, and therefore, the computation fits well on the GPU. The speedup factor of the GPU acceleration compared with the CPU version depends on the type of the GPU, the optimization of the code, and the resolution of the simulation lattice.

4 Amorphous Phenomena

Natural amorphous phenomena play an important role in graphics and visualization simulations. A good flow model for these phenomena should not only describe the flow, but also model the interaction between the flow and the surrounding environment in a physically correct manner. We have applied the LBM to simulate various flow phenomena, including smoke [36], fire [33, 40], light objects floating in the air [34, 35], solid melting [39], and thermal flow phenomena [37], such as heat shimmering and mirage.

4.1 Smoke and Fire

The movements of smoke and fire are highly complex and have many rotational and turbulent details at various levels. While the LBM simulation models the flow field and takes care of the large-scale interactions of the flow with the scene, our rendering approach can add the small-scale interactions and visual details on-the-fly during the interactive viewing process. A key component of our approach are textured splats [33], which can be efficiently rendered on any commodity hardware board. Textured splats allow us to model both the visual detail of the natural phenomena itself as well as the volumetric shadows cast onto objects in the scene.

Besides the texture splatting method, we have also implemented the volume rendering on the GPU for flow visualization. When a fluid source (for example, a smoke inlet) is positioned and begins to release smoke, the smoke density constructs a scalar volumetric dataset. The evolution of this density volume is modeled by an advection-diffusion equation and computed by a back-tracing algorithm based on the method of characteristics [30]. We use the monotonic cubic interpolation [8] for computing the back-tracing density values at positions not on the regular grid sites. Figure 3 shows the rendering results of fire and smoke.

(a) (b) (c)

Figure 4 (a) Soap bubble deformation and dynamics in response to the spray from the spray can; (b) Multiple bubble blown by a flow; (c) Streamlines show the flow pattern originating from a plane cutting through the most active flow region.

4.2 Floating Objects

Using the LBM with boundary conditions appropriate for moving objects, we have simulated the natural dynamics that emerge from the interaction between a flow field and immersed floating objects [34, 35]. If the boundary is fixed in space and located between two neighboring lattice sites, the no-slip condition is implemented with the bounce-back rule of Section 2.2. To match the velocity at the boundary, in the case of a moving object, the bounced-back particle distribution must also be adjusted to account for the momentum imparted by the boundary. Following the approach of Ladd [14], the essential idea is to enforce the no-slip boundary condition at a moving boundary, while maintaining the conservation of mass and momentum. Based on an SRTLBM simulation, the effect of the boundary velocity is to transfer some momentum across the boundary so that the streamed distribution is:

$$f_i'(\mathbf{r}, t+1) = f_i(\mathbf{r}, t^+) - 2W\frac{3}{c^2}\rho(\mathbf{e_i} \cdot \mathbf{u_b}) \tag{15}$$

where the prime indicates the velocity distribution associated with $-\mathbf{e}_i$, $\mathbf{u_b}$ is the object velocity, and W is a constant associated with the lattice velocity.

We demonstrate our approach using soap bubbles. The soap bubbles in Figure 4 illustrate Fresnel reflection, reveal the dynamics of the unseen flow field in which they travel, and display spherical harmonics in their undulations. Our bubble simulation allows the user to directly interact with the flow field to influence the dynamics in real time.

4.3 Thermal Flow

Various realistic phenomena involve hot objects, dynamic flows and heat transfers, such as melting, dissolving, shimmering and mirage, which are of great interest to computer graphicists and visualization experts. For simulating these phenomena, it is imperative to provide a correct and efficient modeling of the heat transfer as well as the interaction between the objects and the flow. Based on the LBM, we developed a physically-based method that provides a basic framework for modeling these thermal flow phenomena [39, 37].

Our method includes conduction, convection and radiation, which are the three basic types of heat transfer in the real world. Heat sources are defined as any arbitrarily shaped objects interacting with the surrounding air. The temperature distribution on the objects can be calculated from radiators (e.g., the sun), or defined by the user with other physical or nonphysical methods. Such temperature distribution is applied to the surface geometry by a novel mechanism, a *temperature texture*. Therefore, we model the heat transfer from the heat sources to the ambient flow. The different heat exchange behaviors are determined

(a) (b) (c)

Figure 5 (a) Heat shimmering from a hot bagel; (b) Mirage in a desert; (c) Melting of a volumetric wax foot.

Figure 6 Contaminant propagation in the West Village of New York City.

by material and flow properties that are controlled by physically meaningful parameters, such as thermal conductivity, Prandtl number, and flow velocity. In the flow region, a hybrid thermal LBM, which couples the MRTLBM with a finite difference discretization of a diffusion-advection equation for temperature, is used for modeling the thermal flow dynamics.

Heat shimmering and mirage appear when the heated air has a different refractive index from that of the cooler surrounding air, resulting in an altered light direction through the hot air compared to that of the cooler air. That is, the changes in the index of refraction are attributed to temperature variation. Once the dynamic temperature distribution is computed by our physically-based modeling framework, we apply a nonlinear ray tracing method to render the resulting visual effects [37]. Figure 5a illustrates the shimmering effects from a hot bagel with a distorted background. In Figure 5b, a phantom body of water appears in the desert due to total reflection. We have also presented a method to simulate the melting and flowing phenomena with different materials in multiple phases [39]. In such a multiphase environment, solid objects are melted because of heating and the melted liquid flows while interacting with the ambient air flow. Figure 5c shows the melting effects of a volumetric foot. When the skin and other soft parts are melted as wax and begin to flow downwards, the bones are revealed.

Figure 7 A closeup view of buildings and smoke.

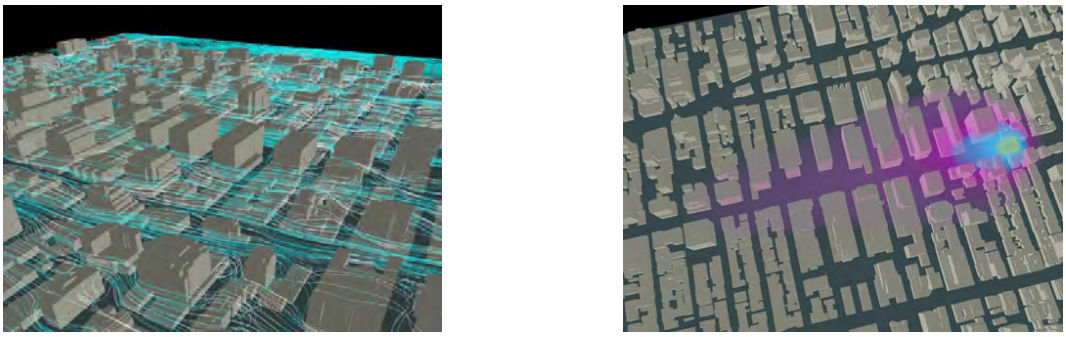
5 Urban Security

The LBM can accurately model air flow and contaminant transport and mixing in geometrically complex environments, such as urban canyons, with the inclusion of thermal effects due to surface heating. Our simulation work [24] is directly relevant to: (1) dispersion models that predict the path and spread of the hazardous agent; (2) interaction with emergency responders who use the information provided by the models. By exploiting the inherent parallelizability of the LBM and implementing the computation on the GPU or a GPU cluster, it is possible to build large scale simulations that span a whole city.

Traditionally, the airborne dispersion of toxic contaminants in open environments is modeled via mesoscale Gaussian plume models which completely disregard the small-scale complex flow dynamics around buildings defining the intricate boundary conditions in deep urban canyons. This may cause the local contamination to be vastly misjudged with devastating consequences for evacuation and mediation efforts. In contrast, our approach uses the MRTLBM to accurately model air flow, contaminant transport and temperature fields within a complex GIS city geometry with a resolution sufficient to accurately simulate its dispersion along the canyons.

To provide an efficient visual interface of the simulation, we first render building facades with textures acquired from real photographs. Because the simulation is executed on the GPU and most of the texture memory is used to store simulation data, there is little space to store textures for buildings. We use noise textures and a smart shader to help texturing the buildings [24]. Then, we render contaminant smoke with self-shadows in real-time with textured splatting method.

To study the behavior of smoke particles, gases, aerosols, and other plumes in a pervasive urban environment, we initially used a 3×3 block area around the Environmental Measurements Laboratory (EML) building in the West Village of New York City. Those sensors record meteorological data (e.g., the wind velocity, temperature, etc.) at a real-time speed. Currently, there are 3 sensors installed in this exercise. Once the live-sensor input is communicated over network links, we adapt the simulation numerical models to accommodate it. The effect of the sensor data is incorporated as a body force [24]. Figure 6 shows the simulation result of a 10-block area rendered by our visualization system. Figure 7 shows the closeup views of the buildings and smoke during the simulation. The LBM model consists of $90 \times 30 \times 60$ lattice sites with the spacing between two neighboring sites less than 5 meters. The building GIS models are at 1 meter resolution. Measured on a computer with a 2.53 GHz Intel Pentium 4 CPU and an NVIDIA GeForce FX 5950 Ultra GPU, our GPU



(a) (b)

Figure 8 Airborne dispersion in the Time Square area. (a) Wind field streamlines; (b) Contaminant density rendered with colors.

implementation has achieved speedup factor of 8 over a CPU imiplementation and can be run at over 12 steps per second.

For large-scale simulations, we have developed a parallel LBM flow simulation on a GPU cluster and simulated the dispersion of airborne contaminants in the Times Square area of New York City [6]. Using 30 GPU nodes, our simulation can compute a $480 \times 400 \times 80$ LBM (the spacing between two lattice sites is 3.8 meters) in 0.31 second/step, a speed which is 4.6 times faster than that of our CPU cluster implementation on 30 CPU nodes. Figure 8 shows the contaminant dispersion in the Time Square area with wind field streamlines and airborne dispersion distribution.

6 Conclusions

We have proposed a solution for the visual simulation of flow phenomena based on a promising microscopic fluid solver, the Lattice Boltzmann Method. The LBM is powerful and flexible due to its great ability for handling complex geometries and accelerating the computation on parallel machines. Our visual simulation system has great benefits in modeling the complex and non-linear flow behaviors with its convincing visual results and accurate prediction simulation. This approach will enable the development of other superior prediction simulation in computer graphics and games, as well as in computational science and engineering.

Acknowledgement

This work has been partially supported by NSF grant CCR-0306438 and from Department of Homeland Security, Environment Measurement Laboratory.

References

1 N. Chu and C. Tai. Moxi: real-time ink dispersion in absorbent paper. *ACM Trans. Graph.*, 24(3):504–511, 2005.

2 D. d'Humiéres, I. Ginzburg, M. Krafczyk, P. Lallemand, and L. Luo. Multiple-relaxation-time lattice Boltzmann models in three-dimensions. *Philosophical Transactions of Royal Society of London*, 360(1792):437–451, 2002.

3 D. Ebert, F. Musgrave, D Peachey, K. Perlin, and S. Worley. *Texturing and Modeling, A Procedural Approach.* Morgan Kaufmann Press, third edition edition, 2002.

4 D. Ebert and R. Parent. Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques. *Proceedings of SIGGRAPH*, pages 357–366, 1990.

5 D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *Proceedings of SIGGRAPH*, pages 736–744, 2002.

6 Z. Fan, F. Qiu, A. Kaufman, and S. Yoakum-Stover. GPU cluster for high performance computing. *Proceedings of ACM/IEEE Supercomputing Conference*, 2004.

7 Z. Fan, Y. Zhao, A. Kaufman, and Y. He. Adapted unstructured lbm for flow simulation on curved surfaces. *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 245–254, 2005.

8 R. Fedkiw, J. Stam, and H. Jensen. Visual simulation of smoke. *Proceedings of SIGGRAPH*, pages 15–22, 2001.

9 B. Feldman, J. O'Brien, and O. Arikan. Animating suspended particle explosions. *ACM Trans. Graph.*, 22(3):708–715, 2003.

10 N. Foster and D. Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

11 N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. *Proceedings of SIGGRAPH*, pages 181–188, 1997.

12 T. Goktekin, A. Bargteil, and J. O'Brien. A method for animating viscoelastic fluids. *ACM Trans. Graph.*, 23(3):463–468, 2004.

13 S. A. King, R. A. Crawfis, and W. Reid. Fast volume rendering and animation of amorphous phenomena. *Proceedings of Volume Graphics*, pages 229–242, 2000.

14 A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part I. Theoretical foundation. *J. Fluid Mech.*, 271:285–309, 1994.

15 P. Lallemand and L. Luo. Theory of the lattice Boltzmann method: Acoustic and thermal properties in two and three dimensions. *Physical Review E*, 68:036706, 2003.

16 W. Li, Z. Fan, X. Wei, and A. Kaufman. *Flow Simulation with Complex Boundaries*, chapter 47, pages 747–764. Addison-Wesley, 2005.

17 W. Li, X. Wei, and A. Kaufman. Implementing lattice Boltzmann computation on graphics hardware. *The Visual Computer*, 19(7-8):444–456, December 2003.

18 R. Mei, L. S. Luo, and W. Shyy. An accurate curved boundary treatment in the lattice Boltzmann method. *Journal of Comp. Phys.*, 155:307–330, June 1999.

19 M. Mueller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 154–159, 2003.

20 M. Nelson and R. Crawfis. Visualizing wind velocities by advecting cloud textures. *Proceedings of Visualization*, pages 179–184, 1992.

21 D. Nguyen, R. Fedkiw, and H. Jensen. Physically based modeling and animation of fire. *Proceeding of SIGGRAPH 2002*, pages 721–728, 2002.

22 John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell. A survey of general-purpose computation on graphics hardware. *Proceedings of Eurographics*, pages 21–51, 2005.

23 S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R.Whitaker. Particle-based simulation of fluids. *Proceeding of Eurographics*, pages 401–410, 2003.

24 F. Qiu, Y. Zhao, Z. Fan, X. Wei, H. Lorenz, J. Wang, S. Yoakum-Stover, A. Kaufman, and K. Mueller. Dispersion simulation and visualization for urban security. *Proceedings of Visualization*, pages 553–560, October 2004.

25 N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 193–202, 2004.

26 W. T. Reeves. Particle system - a technique for modeling a class of fuzzy objects. *Proceedings of SIGGRAPH*, 17(3):359–376, 1983.

27 G. Sakas. Modeling and animating turbulent gaseous phenomena using spectral synthesis. *The Visual Computer*, pages 200–212, 1993.

28 A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph.*, 24(3):910–914, 2005.

29 M. Shinya and A. Fournier. Stochastic motion - motion under the influence of wind. *Computer Graphics Forum*, 11(3):C119–128, September 1992.

30 J. Stam. Stable fluids. *Proceedings of SIGGRAPH*, pages 121–128, 1999.

31 J. Stam. Flows on surfaces of arbitrary topology. *ACM Trans. Graph.*, 22(3):724–731, 2003.

32 S. Succi. *The lattice Boltzmann equation for fluid dynamics and beyond.* Numerical Mathematics and Scientific Computation. Oxford University Press, 2001.

33 X. Wei, W. Li, K. Mueller, and A. Kaufman. Simulating fire with texture splats. *Proceedings of Visualization*, pages 227–237, 2002.

34 X. Wei, Y. Zhao, Z. Fan, W. Li, F. Qiu, S. Yoakum-Stover, and A. Kaufman. Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):719–729, November 2004.

35 X. Wei, Y. Zhao, Z. Fan, W. Li, S. Yoakum-Stover, and A. Kaufman. Blowing in the wind. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 75–85, July 2003.

36 Xiaoming Wei, Wei Li, Klaus Mueller, and Arie E. Kaufman. The lattice-boltzmann method for simulating gaseous phenomena. *IEEE Trans. on Visualization and Computer Graphics*, 10(3):164–176, March/April 2004.

37 Y. Zhao, Y. Han, Z. Fan, F. Qiu, Y. Kuo, A. Kaufman, and K. Mueller. Visual simulation of heat shimmering and mirage. *Stony Brook CVC Technical Report No. 20051219, Submitted*, 2005.

38 Y. Zhao, F. Qiu, Z. Fan, and A. Kaufman. Interactive fluid simulation with multi-resolution LBM. *Stony Brook CVC Technical Report No. 20060206, Submitted*, 2006.

39 Y. Zhao, L. Wang, F. Qiu, A. Kaufman, and K. Mueller. Melting and flowing in multiphase environment. *Computers & Graphics, Special Issue on Natural Phenomena*, 2006.

40 Y. Zhao, X. Wei, Z. Fan, A. Kaufman, and H. Qin. Voxels on fire. *Proceedings of Visualization*, pages 271–278, October 2003.

Local and Global Illumination in the Volume Rendering Integral

Nelson Max[1] and Min Chen[2]

1 **University of California, Davis, USA**
`max@cs.ucdavis.edu`
2 **Swansea University, Swansea, UK**
`m.chen@swansea.ac.uk`

Abstract

This article is intended as an update of the major survey by Max [40] on optical models for direct volume rendering. It provides a brief overview of the subject scope covered by [40], and brings recent developments, such as new shadow algorithms and refraction rendering, into the perspective. In particular, we examine three fundamentals aspects of direct volume rendering, namely *the volume rendering integral*, *local illumination models* and *global illumination models*, in a wavelength-independent manner. We review the developments on *spectral volume rendering*, in which visible light are considered as a form of electromagnetic radiation, optical models are implemented in conjunction with representations of spectral power distribution. This survey can provide a basis for, and encourage, new efforts for developing and using complex illumination models to achieve better realism and perception through optical correctness.

1998 ACM Subject Classification I.3.7 Three-Dimensional Graphics and Realism

Keywords and phrases Volume Rendering, Illumination Model

Digital Object Identifier 10.4230/DFU.SciViz.2010.259

1 Introduction

The process of volume rendering maps a 3D scalar field into a 3D field of optical properties, usually color c and extinction coefficient τ, via so-called transfer functions, and then approximates its visual appearance by integrating along viewing rays.

Max provided a comprehensive survey of optical models for direct volume rendering in 1995 [40], and the survey has been extensively referenced in the volume rendering literature. Since then, there have been some new developments in the context of optical and illumination models for volume rendering, such as spectral volume rendering, shadow algorithms and refraction rendering. The authors believe that it is beneficial to re-visit this topic and bring these new developments into the perspective. On one hand, this survey is written as an update of [40], so it does not repeat the details of many technical aspects that were presented in [40]. On the other hand, it is also intended to be a self-contained work, so some fundamentals have been included.

In Sections 2, 3 and 4, we will examine the volume rendering integral, and local and global illumination models used in volume rendering, respectively. In these sections, the discussions on lights and colors are wavelength-independent, and the optical models presented were often applied directly to RGB-based implementations in practice. In Section 5, we review the developments on spectral volume rendering, in which optical models were implemented in a wavelength-dependent manner.

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 259–274

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

2 The Volume Rendering Integral

2.1 Basic Optical Model

The simplest physical model is that of a collection of small non-reflecting opaque particles, glowing with color c, which are the only light sources in the scene. The extinction coefficient τ represents the differential probability of a viewing ray hitting a particle, so that the probability of the ray hitting a particle along an infinitesimal length ds is $\tau\, ds$. Max [40] gives a derivation of $\tau = \pi r^2 N$ in terms of the radius r and number density N of the particles. In particular, τ is proportional to N. For particles of constant radius and material, τ is then also proportional to the mass density ρ of the particles, so that $\tau = \rho\kappa$, where κ depends on the material.

The transparency $T(s)$ represents the probability that the viewing ray travels a distance s away from the viewpoint without hitting any particles. Since the events "not hitting between distances 0 and s" and "not hitting between distances s and $s+ds$" are independent, their probabilities multiply, so

$$T(s+ds) = T(s)\big(1-\tau(s)ds\big).$$

Thus

$$dT = T(s+ds) - T(s) = -T(s)\tau(s)ds,$$

so

$$dT/T = -\tau(s)ds.$$

Using the initial condition $T(0) = 1$,

$$\ln(T(s)) = -\int_0^s \tau(u)\,du,$$

and

$$T(s) = \exp\left(-\int_0^s \tau(u)\,du\right). \tag{1}$$

By the same independent event argument, the probability of not hitting a particle between distances 0 and s, and then hitting one between distances s and $s+ds$, is $T(s)\tau(s)ds$. In this case, the color will be $c(s)$, the color of the particles at position s. The values of τ and c at a given 3D position are usually specified by transfer functions of the scalar variable being visualized. The ray may pass through the complete volume, and hit an opaque background at distance D with probability $T(D)$, in which case it will have the background color B (which may depend on the point hit). The expected color E for the ray is gotten by averaging the colors for all these hitting events, weighted by their probabilities:

$$E = \int_0^D T(s)\tau(s)c(s)\,ds + T(D)B. \tag{2}$$

(For another derivation of this, see [40].) The product $\tau(s)c(s)$ can be thought of as a spatially varying color $C(s)$ which already has the effects of the particle density factored in. In the compositing literature, C is called a "premultiplied" color, and c is called a "non-premultiplied" color. In volume rendering it is also common for the transfer function to specify a premultiplied color, in which case

$$E = \int_0^D T(s)C(s)\,ds + T(D)B. \tag{3}$$

2.2 Integration by Discrete Sampling

The simplest way of estimating the integral in Equation (2) is to subdivide the interval from 0 to D into n segments of equal length $\Delta s = D/n$ and approximate the integral as a Riemann sum:

$$\int_0^D T(s)\tau(s)c(s)\,ds \cong \sum_{i=0}^{n-1} T(s_i)\tau(s_i)c(s_i)\Delta s,$$

where s_i is a sample in the i^{th} segment. If $s_i = i\Delta s$ is at the left hand end of the i^{th} segment,

$$\begin{aligned} T(s_i) &= \exp\left(-\int_0^{s_i} \tau(u)\,du\right) \\ &= \prod_{j=1}^{i} \exp\left(-\int_{s_{j-1}}^{s_j} \tau(u)\,du\right). \end{aligned}$$

If we assume $\tau(s)$ is constant on each segment (a further approximation),

$$T(s_i) \cong \prod_{j=1}^{i} \exp\left(-\tau(s_j)\Delta s\right).$$

Thus we can compute E by the following Iteration A:

```
E = 0.
T = 1.
for i = 0 to n-1 do
    E = E + T*tau[i]c[i]
    T = T*exp(-tau[i]*DeltaS)
E = E + T*B
```

Early ray termination stops this computation when T becomes zero, or small enough so that further terms have negligible effect. A similar iteration can be written for back to front accumulation of the contributions, which does not need the temporary variable T, but cannot do early ray termination.

A further approximation is to estimate $\exp(x)$ by the first two terms in its Taylor series:

$$\exp(x) = 1 + x + x^2/2 + \cdots$$

so that

$$\exp(-\tau(s)\Delta s) \cong 1 - \tau(s)\Delta s. \tag{4}$$

This approximation is only good when the product $\tau(s)\Delta s$ is small; otherwise it may cause visible artifacts. If it is made, the steps in the above iteration can be achieved in simple compositing hardware. The data volume is sliced by a series of equally-spaced planes into textured polygons that can be scan converted and composited in hardware.

On a regular cubic or rectilinear grid, the data at s_i can be interpolated from the eight surrounding grid values using 3D texture mapping hardware. If only 2D texture mapping is available, the slices can be taken along a set of parallel coordinate planes in the grid, passing through the aligned data samples. It is more accurate to interpolate the scalar data and then apply the transfer functions for color and opacity, rather than sample the color and opacity at the grid and then interpolate them. This is easily done in hardware using dependent textures.

2.3 Integration along Ray Segments

Garrity [15] showed how to trace a ray through a tetrahedral grid, by determining through which face it exits a tetrahedron, and following a neighbor pointer to the next adjacent tetrahedron. Weiler *et al.* [56] have implemented this on a GPU.

Assume that the i^{th} segment is bounded by ray distances s_i and s_{i+1} and thus has length $\Delta s_i = s_{i+1} - s_i$, and that within its tetrahedron, the optical properties c_i and τ_i are constant. Then substituting Equation (1) into Equation (2), separating out the transparency up to s_i, and then integrating, the contribution E_i of the i^{th} segment to E is

$$\begin{aligned}
E_i &= \int_{s_i}^{s_{i+1}} \tau_i c_i T(s)\, ds \\
&= \int_{s_i}^{s_{i+1}} \tau_i c_i \exp\left(-\int_0^s \tau(u)\, du\right) ds \\
&= \tau_i c_i \int_{s_i}^{s_{i+1}} \exp\left(-\int_0^{s_i} \tau(u)\, du\right) \cdot \exp\left(-\int_{s_i}^{s} \tau_i\, du\right) ds \\
&= T(s_i)\tau_i c_i \int_{s_i}^{s_{i+1}} \exp(-\tau_i(s-s_i)\, ds \\
&= T(s_i)\tau_i c_i \left.\frac{\exp(-\tau_i(s-s_i))}{-\tau_i}\right|_{s_i}^{s_{i+1}} \\
&= T(s_i) c_i \big(1 - \exp(-\tau_i \Delta s_i)\big).
\end{aligned}$$

The following Iteration B can then compute the radiance along the ray with floating point accuracy:

```
E = 0.
T = 1.
for i = 0 to n-1 do
    F = exp(-tau[i]*DeltaS[i])
    E = E + T*c[i]*(1.  - F)
    T = T*F
E = E + T*B
```

Even for the discrete sampling in the previous section, Iteration B will be more accurate than the Iteration A derived from the Riemann sum, if $\tau \Delta s$ is large, so that T varies significantly between samples.

Shirley and Tuchman [52] divided the projection of a tetrahedron by its projected edges into triangles in software, and used polygon rendering and compositing to render them in hardware. Wylie *et al.* [60] have now done all steps in hardware, except the global visibility sorting required to determine the compositing order. The same principles apply to a mesh of arbitrary convex polyhedral cells, as shown by Williams *et al.* [59], which also presents a visibility sort based on a method proposed by Martin Newell.

This visibility sorting remains a bottleneck, and in fact is not always possible, because there may be a cycle of tetrahedra, each overlapping the next. Visibility sorting has its own extensive literature, which is beyond the scope of this survey.

2.4 Analytic Integration

So far we have assumed that the optical properties are constant within each cell. Another case which can be handled analytically is when they vary linearly along each ray. For example, this is the case if the scalar function is linearly interpolated from its values at the four vertices of a tetrahedron, and the transfer functions mapping it to τ and c or C are also linear. In such cases, applying a symbolic

integration package to the integral in Equation (2) produces a rather complicated analytic expression, first presented by Williams and Max [58]. As shown in [59], this method also applies to piecewise linear transfer functions, by dividing the tetrahedra into sub-polyhedra in which the transfer function is linear. That paper also improves the numerical stability of the software evaluation. Moreland and Angel [43] have used a GPU to evaluate this analytic expression in hardware, by precomputing part of it into texture tables.

2.5 Other Integration Methods

If the volume data is determined at grid points, the discrete sampling methods in Section 2.2 are appropriate, but if the data comes from a finite element grid, integration along the ray segments in each grid cell may be more accurate and efficient. However, only the constant case in Section 2.3 and the linear case in Section 2.4 admit analytic solutions. The transfer function may not be linear or even piecewise linear. Also, the interpolated scalar value along a ray may not be linear. For example, for trilinear interpolation on hexahedral elements, the scalar function is a cubic polynomial along general rays. In addition, higher order finite elements may be used in the physical simulation for better convergence.

A further problem with finite elements is that they may be deformed, that is, the mapping from a standard element shape like a cube or regular tetrahedron into the physical simulation space may be non-linear. Often, the same higher order polynomials used to define the scalar function on the standard element are also used to map the standard element into physical space. Thus, an inverse map is needed before the scalar function can even be evaluated at a point on the ray. In fact, a ray can intersect a deformed element in more than one segment. Wiley *et al.* [57] show how to determine these segments for the case of deformed quadratic tetrahedra, and approximate the inverse mapping along them by polynomial splines.

Once the ray segments are known, there are several ways that the integral can be approximated. In [57], the equally spaced sampling method of Section 2.2 was used. But for smoothly varying transfer functions, Gaussian integration [48] can give greater accuracy for the same number of sample points, by approximating the integral as an unequally weighted sum of the integrand at n unequally spaced sample points. The sample locations and sample weights are chosen to give the exact integral for polynomials of the maximum degree, $2n-1$, allowed by the $2n$ degrees of freedom in these positions and weights. So if the integrand is well approximated by such polynomials, Gaussian integration will give a good approximation. But this is not always the case, especially for non-smooth transfer functions that are selected to emphasize certain contour surfaces of the scalar function. For this application, discussed in more detail in the next section, it may be more appropriate to find the exact intersection of the ray with any contour surface within the volume element. Williams *et al.* [59] used the quadratic formula to intersect rays with contour surfaces within undeformed tetrahedra with quadratically varying scalar functions. Kirby and Nelson [28] estimate the scaler function along the inverse-mapped ray within a higher order deformed element by a high degree polynomial, and then use a general root finding procedure to find its first root along the ray.

Another possibility is to precompute and store the integrals. For a linear variation of the scalar function along a ray segment and arbitrary transfer functions, the integral along the segment depends only on the segment length, and the scalar values at its two endpoints. Röttger *et al.* [50] propose storing the integrals in a 3D texture, indexed by these three variables. Weiler *et al.* [56] used this in their hardware ray tracing, and proposed an efficient incremental method of doing the precomputation. For the hardware implementation by plane slicing given at the end of Section 2.2, the ray segment length is constant in an orthogonal view, and approximately constant in a perspective view, so only a 2D texture is required for the precomputed integrals. A good reference on hardware implementation of the methods in this and the next two sections is [13].

3 Local Illumination

The volume rendering integrals given in Equations (2) and (3) represent a typical optical model which is referred to as the *absorption plus emission model* [40] and is perhaps the most widely-used optical model in volume visualization. The non-premultiplied color $c(s)$ can simply be a function of emission property at a sample point s by considering the volume as a light-emitting medium. Another approach is to involve one or more light sources in the computation of $c(s)$. In both cases, the illumination at s depends not only the optical properties sampled at s and the intensity of each light source, but also indirect light reflected towards s from other part of the medium (i.e., scattering) as well as the absorptivity of the medium that determines how much light can eventually arrived at s (i.e., shadows). Such an illumination model is referred to as *global illumination*, which will be discussed in detail in Section 4.

To avoid costly computation with a global illumination model, it is common to adopt a *local illumination model* where $c(s)$ is estimated based only on the optical properties sampled at s and the intensity of each light source. This allows us to rewrite Equation (2) as:

$$E = \int_0^D T(s)\tau(s)\left(c_g(s) + \sum_{i=1}^{k} c_r(s, L_i)\right) ds + T(D)B,$$

where $c_g(s)$ is the sampled intensity of self-glowing at s, and $c_r(s, L_i)$ defines the light reflected due to light source $L_i\,(i = 1, \ldots, k)$. In many applications, a local illumination model is normally adequate for rendering a single isosurface within a volume. When handling multiple isosurfaces, or amorphous regions, one needs to aware the limitation of such a model and the potential perceptual discrepancy due to the omission of shadows and indirect lighting.

3.1 Classic Illumination Models

Given a light source L, one can estimate c_r at s locally by using one of the empirical or physically-based illumination models designed for surface geometry, such as the Phong, Phong-Blinn, and Cook-Torrance models [14]. When such a model is used in volume rendering, it is assumed that each sampling position, s, is associated with a surface or microfacet. This assumption allows us to compute the surface normal at s, which is required by almost all surface-based illumination models.

In volume models, surface geometry is normally not explicitly defined, and in many situations, models do not even assume the existence of a surface. Hence, the computation of surface normals is usually substituted by that of gradient vectors. While for some parametric or procedurally-defined volume models, it is possible to derive gradient vectors analytically, in most applications, especially where discrete volumetric models are used, gradient vectors are estimated, for example, using the finite differences method for rectangular grids [41], and 4D linear regression for both regular and irregular grids [44]. The commonly used central differences method is a reduced form of finite differences based on the first two terms of the Taylor series. There are many other gradient estimation methods, including schemes that involve more or fewer neighboring samples (e.g., [47, 63]), and schemes where the discrete volume models are first convolved using a high-order interpolation function, and gradients are computed as the first derivative of the interpolation function (e.g., [6, 45]). Möller *et al.* compared a few normal estimation schemes in the context of volume visualization [42].

Usually the local illumination models are only applied at or near a presumed surface within the volume, so a surface presence indicator is used to weight (i.e. multiply) the computed local illumination. Levoy [36] describes two methods of computing this surface weight, using formulas involving the scalar value and its gradient magnitude, and Kindleman and Durkin [27] give a method which uses a 2D texture table indexed by these two quantities. Kraus [32] points out that if the task is to determine only if a contour surfaces is intersected by a ray segment, this can be indicated by a

2D texture table indexed by the two scalar values at the endpoints of the segment. Alternatively, ray tracing can be used to locate points exactly on an isosurface, as described in Section 2.5, and the local illumination can be applied only at these points.

3.2 Measured and Precomputed BRDFs

The light reflected from a point on a surface can be described by a *bi-directional reflection distribution function* (BRDF). Hence, it is feasible to obtain a BRDF in sampled form by either measurement or computer simulation [16]. The measurements of a BRDF are usually made using a goniophotometer in a large number of directions, in terms of polar and azimuth angles, uniformly distributed on a hemisphere about a source [20]. In computer graphics, it is also common to precompute discrete samples of a BRDF on a hemisphere surrounding a surface element (e.g., [25, 4]).

Given n sampling points on a hemisphere, and n possible incident directions of light, a BRDF can be represented by an $n \times n$ matrix. Given an arbitrary incident light vector, and an arbitrary viewing vector, one can determine the local illuminance along the viewing vector by performing two look-up operations and interpolating up to 16 samples.

One major advantage of using measured or pre-computed BRDFs is that the computation of $c(s)$ in Equation (2) or $C(s)$ in Equation (3) no longer needs to rely on an illumination model that can easily be defined and computed. One can use measured data to compensate for the lack of an appropriate model that accounts for a range of physical attributes, or use precomputed data for a complicated and computationally intensive model (e.g., an anisotropic model as in [25]).

Similar to a BRDF, the light transmitted at a point on a surface can be described by a *bi-directional transmittance distribution function* (BTDF). Hanrahan and Krueger [17] considered both BRDF and BTDF in a multi-layered surface model, which can be viewed as a simplified volume model. Baranoski and Rokne [3] applied this approach to the modeling of foliar scattering. Wang *et al.* [55] obtained their BRDFs and BTDFs by fitting parametric models to measured reflectance and transmittance data.

3.3 Phase Functions

A phase function, $r(\omega, \omega')$, defines a probability distribution of scattering in direction ω with respect to the direction of the incident light, ω' [10]. More precisely, $r(\omega, \omega')d\omega$ represents the probability that light flowing in direction ω' that scatters from a particle scatters into the solid angle $d\omega$ about the direction ω. Blinn [10] gives the phase function for a spherical diffusely reflecting particle. The Henyey-Greenstein phase function [19] is also popular in computer graphics. A discussion of the phase function in the context of multiple scattering will be given in Section 4.1. Here we briefly describe its use as a local illumination model.

The fundamental difference between such an illumination model and those in 3.1 and 3.2 is that it is entirely volumetric and does not assume the existence of a surface or microfacet at every visible point in space. While phase functions are largely used in the context of global illumination, they can be used as for local illumination in a perhaps rather simplified manner. Despite the omission of the multiple scattering feature in the context of local illumination, phase functions allow a volumetric point to be lit by light from any direction. On the contrary, classic illumination models and BRDFs consider only light in front of the assumed surface or microfacet defined at the point concerned.

3.4 Other Related Developments

Many local illumination methods developed for surface rendering have been, or can be, used in conjunction with the volume rendering integral. These include Blinn and Newell's environment map [9], Arvo's illumination map [2], Heckbert's radiosity texture [18]. However, the application of these

methods in volume rendering has so far been largely limited to the rendering of a single iso-surface [31].

The discrete sampling process described in Section 2.2 facilitates a scattering event at each sample. In such a process, refraction can be realized as a local illumination feature by altering the ray path. Rodgman and Chen examined several sampling methods for rendering refraction in conjunction with the volume rendering integral [49], and employed nonlinear diffusion filters to improve the quality of refraction rendering. Li and Mueller studied the use of different interpolation filters in a surface-based approach to rendering refraction [38, 37].

In many visualization applications, it is appropriate, and often desirable, to use *non-photorealistic lighting* to enhance the perception of synthesized visualizations. Recently, Stewart employed local occuluders and uniform diffuse illumination to render pseudo-shadows in depressions and crevices [54]. Lee *et al.* used globally inconsistent lighting to enhance perception of shapes [35]. Lum *et al.* applied transfer functions to incoming light to provide better perception of material thickness and boundaries [39].

4 Global Illumination

4.1 The Basic Equation for Multiple Scattering

If the particles in the optical model scatter as well as emit light, the mathematical situation is more complicated than in section 2. For a complete solution, we must determine $I(x,\omega)$, the radiance (light intensity) flowing through every 3D point x in the volume, in every direction ω on the unit sphere, taking into account the effects of multiple scattering.

The probability that light hitting a particle scatters instead of being absorbed is called the albedo a. The scattering depends on the direction of the incoming and scattering rays according to the phase function $r(\omega,\omega')$.

Let the source function $g(x,\omega)$ represent the light emitted or inscattered into direction ω by a particle at position x. Integrating the scattered incoming light over all incoming directions ω' in the unit sphere Ω, taking into account the effects of the albedo and the phase function, and finally adding on the glow $c_g(x)$, we get

$$g(x,\omega) = \int_\Omega a(x) r(\omega,\omega') I(x,\omega')\, d\omega' + c_g(x). \tag{5}$$

In practice, a usually does not depend on x, and the phase function $r(\omega,\omega')$ depends only on the scattering angle between the unit direction vectors ω and ω'.

In order to write the multiple scattering version of Equation (2), giving an expression for $I(x,\omega)$, we substitute $g(x,\omega)$ for $c(s)$. We evaluate the optical properties at points $x(s) = x - \omega s$, since the integral in (2) is along a "viewing ray" in the direction opposite to the light flow. Thus we have

$$I(x,\omega) = \int_0^D T(s)\tau(x(s)) g(x(s),\omega)\, ds + T(D)B. \tag{6}$$

This equation is difficult to solve, since $g(x(s),\omega)$ depends via Equation (5) on $I(x,\omega')$ at all the points $x(s)$ on the viewing ray and in all directions ω' in the unit sphere, so that all the $I(x,\omega)$ must be solved for simultaneously. Surveys of techniques for the solution are given in Pérez *et al.* [46] and Max [40].

4.2 Single Scattering Approximation

One simplifying assumption is that the albedo and/or density is low, so that the probability of light scattering more than once is small, and only single scattering need be considered. For further

simplicity here, we will also assume that there is no glow c_g, and only one parallel light source with intensity L_0 at an infinite distance in the single direction ω'. Then we can precompute the shadowing effects of the volume opacity, and determine the $I(x,\omega')$ inside the integral in Equation (5) by using Equation (1) to obtain

$$I(x,\omega') = L_0 \exp\left(-\int_0^D \tau(x-s\omega')\,ds\right).$$

Kajiya and von Herzen [26] did this in a first pass through the volume, essentially computing $I(x,\omega')$ by the part of Iteration A involving only T. Bahrens and Rattering [5] give an accurate and efficient method for this shadow pre-computation on gridded data, using texture mapping hardware, by moving a sampling plane perpendicular to ω' in discrete steps along the light flow direction. This shadow computation can be done in the same pass as the volume rendering if the sampling plane instead bisects the angle between the viewing and illumination rays, as in [29].

4.3 Diffusion Approximation

Another simplifying assumption is that the albedo is high, and that the size of the volume features is large compared to the mean free path $1/\tau$. Then almost every ray seen at the viewpoint will have been scattered so many times that all the directional properties of $r(\omega,\omega')$ will be washed out by multiple spherical convolutions, and the scattering will be effectively diffuse, equal in all directions. In this case, the flow of light can be modeled by a second order partial differential equation for $I(x)$ as a function of position only. Stam [53] first introduced this equation to computer graphics, and solved it using the multi-grid method. (His equation is off by a factor of 3 in a couple of the terms. For a correct derivation see Ishimaru [21].)

Jensen *et al.* [24] introduced an approximate solution to this equation based on fitting the analytic solution for the diffusion from two virtual point light sources, one below a planar surface bounding the participating medium, and one above it, outside the medium. This approximation is only valid for a semi-infinite domain with constant optical properties, bounded by a flat plane, but it has been applied to give realistic renderings for other geometries. Jensen and Buhler [22] used an octree hierarchy to account for illumination at all the surface points, and this idea has now been refined by various authors, for example Dachsbacher and Stamminger [11], to give real time performance on graphics hardware. However, the basic technique of Jensen *et al.* [24] is only applicable to homogenous materials, and is less useful for volume rendering of spatially varying data.

4.4 Other Multiple Scattering Methods

For non-homogeneous materials, there are several methods not covered in the survey of Perez *et al.* [46]. Jensen and Christiansen [23] extend photon mapping to participating media. They do a Monte Carlo simulation of photon transport from the light sources, which can take into account spatially varying optical properties and general phase functions. They record each scattering event in a spatial data structure called a photon map. In a final gather pass from the viewpoint, they collect the scattering events relevant to a viewing ray, to account for the inscattered photons. Of course, this can be slow, since many photons are required for accurate convergence.

Kniss *et al.* [30] have generalized the single pass shadow algorithms given at the end of Section 4.2 to multiple forward scattering, by gathering the accumulated light from several sampled directions in the previous sampling plane. Zhang *et al.* [62] have added backward scattering from the next couple of sampling planes.

5 Spectral Volume Rendering

We now consider visible light as a form of electromagnetic radiation. The radiative power emitted by an object is typically defined as a function known as the *spectral power distribution (SPD)*, $F(\lambda)$, where λ is the wavelength within the radiation band concerned. In color computation, it is common to limit this range to the visible spectrum, $\lambda \in [380nm, 770nm]$, or often a narrower range, $\lambda \in [400nm, 700nm]$, to which human eyes are more sensitive.

5.1 Basic Optical Models

The transmission of light of a single wavelength λ through a homogeneous isotropic absorption filter (such as glass and gelatin) is governed by the Bouguer's or Lambert's law (1727, 1760), which states that the intensity of an incoming light, $L_0(\lambda)$ decreases exponentially with the path length s in the filter medium, that is:

$$L(\lambda) = L_0(\lambda) \cdot \exp\big(-s \cdot \tau(\lambda)\big),$$

where $\tau(\lambda)$ is the spectral extinction coefficient (commonly referred to as absorptivity) of the medium. Beer (1983) extended the Lambert-Bouguer law to a liquid solution with a low or moderate concentration of an absorbing solute, as:

$$L(\lambda) = L_0(\lambda) \cdot \exp\big(-s \cdot \nu \cdot \kappa(\lambda)\big),$$

where ν represents the concentration of the solute and $\kappa(\lambda)$ is the extinction coefficient of the solute [61].

When the transparency $T(\lambda, \Delta) = \exp\big(-\Delta \cdot \tau(\lambda)\big)$ is known for a standard path length Δ, we can obtain the transparency for an arbitrary path length δ as:

$$T(\lambda, \delta) = T(\lambda, \Delta)^{\frac{\delta}{\Delta}}.$$

This is more often written in the form of a depth correction formula for opacity, $\alpha(\lambda, \delta)$, as:

$$\alpha(\lambda, \delta) = 1 - T(\lambda, \delta) = 1 - \big(1 - \alpha(\lambda, \Delta)\big)^{\frac{\delta}{\Delta}}.$$

For a homogeneous medium, the Lambert-Bouguer law, which was derived from experimentation, is consistent with Equation (1), which was derived independently based on the notion of absorbing particles. Comparing with the early discussion in Section 2.1, the Lambert-Bouguer law is a special case of Equation (1), while transparency $T(s)$ in Equation (1) can be considered as an approximated extension of the Lambert-Bouguer law by removing the homogeneity condition and assuming the same refractive index for materials with different $\tau(s)$.

The basic optical model proposed by Bouguer and Lambert, and a wavelength-dependent version of Equation (1) form the basis of the two spectral volume rendering integrals in 5.2 and 5.3.

5.2 A One-pass Rendering Integral

Bergner *et al.* [7, 8] developed a spectral volume rendering integral, partially based on the multiple scattering model described in Section 4.1. In order to facilitate ray casting with local illumination, they simplified the multiple scattering model by removing all the global illumination elements in the model. This includes (*i*) assigning volumetric shadow ratio to constant 1 for all voxels and directions, (*ii*) considering only irradiance and radiance with the same direction as the viewing ray, and (*iii*) approximating the radiance at each point by the reflectance of the local materials.

In particular, Bergner *et al.* successfully factored the light source out in the computation of their volume rendering integral. This enables interactive data integration using different light sources during post illumination.

Let $L(\lambda)$ be a light source of a single wavelength, and $I_i(x,\omega,\lambda)$ be the irradiance at a sampling position x reached from a light ray of direction ω. I_i is computed as:

$$I_i(x,\omega,\lambda) = L(\lambda)\cdot\tilde{I}_i(x,\omega,\lambda,D),$$

where D is the distance between x and the point d where the light ray first enters the volume. $\tilde{I}_i(x,\omega,\lambda)$ indicates the proportion of $L(\lambda)$ that has arrived at x from d, which is:

$$\tilde{I}_i(x,\omega,\lambda) = \tilde{I}_i(x-D\omega,\omega,\lambda)\cdot T(x,\omega,\lambda,D) + \int_0^D \tilde{I}_r(x-s\omega,\omega,\lambda)\cdot T(x,\omega,\lambda,s)\,ds,$$

where $\tilde{I}_i(x-D\omega,\omega,\lambda)$ is the irradiance at d, $T(x,\omega,\lambda,s)$ is the spectral version of the extinction function in Equation (1), i.e.,

$$T(x,\omega,\lambda,s) = \exp\left(-\int_0^s \tau(x-t\omega,\lambda)dt\right),$$

and $\tilde{I}_r(x-s\omega,\omega,\lambda)$ indicates the scattered radiance from the external illumination. Because of the abovementioned simplification, the term $\tilde{I}_r(x-s\omega,\omega,\lambda)$ is computed with a local illumination model based on the Phong model.

5.3 A Two-pass Spectral Rendering Integral

Noordmans *et al.* considered a spectral volume rendering integral which features shadow computation. They adopted the Kajiya and von Herzen's two pass algorithm [26] for computing volumetric shadows and adapted the notion of opacity in the traditional RGBα-based volume rendering. Let $\rho_m(x)$ be the mass density of a material m at x, and $\kappa_m(\lambda)$ be the absorption attribute of m at wavelength λ. Consider the position x features M materials. The spectral representation of opacity at x is defined as:

$$\tau(x,\lambda) = \sum_{m=1}^{M} \rho_m(x)\cdot\kappa_m(\lambda).$$

In the *illumination* phase, the flux of the light is propagated through the volume using discrete ray casting. At each sampling position x, the incident light I_i is related to the transmitted light I_t (which subsequently becomes the incident light at the next sample) simply as:

$$I_i(x+\Delta x,\lambda) = I_t(x,\lambda) = I_i\cdot\big(1-\Delta x\cdot\tau(x,\lambda)\big).$$

In the *radiation* phase, the flux of radiance is accumulated also using back-to-front ray casting. At each sampling position x, the radiance I_r is related to the irradiance I_b arriving from the previous sample, and the local emission I_e as:

$$I_b(x+\Delta x,\lambda) = I_r(x,\lambda) = I_e(x,\lambda) + I_b(x,\lambda)\cdot\big(1-\Delta x\cdot\tau(x,\lambda)\big).$$

Note that as mentioned in conjunction with Equation (4), the term $1-\Delta x\cdot\tau(x,\lambda)$ used in both phases is only an approximation.

One particular interesting aspect of the work by Noordmans *et al.* is the design of its emission function, which is split into two parts, namely elastic scattering and inelastic scattering. The former features scattering in a chromatic medium, with each material m is associated with a specific spectral band, facilitating a spectral transfer function. The latter enables materials to absorb the incident light at one wavelength and re-emit the energy at another, facilitating a simulation of fluorescence and phosphorescence materials.

5.4 One-dimensional Radiosity

Abdul-Rahman and Chen [1] presented a spectral volume rendering integral based on the optical model proposed by Kubelka and Munk [34], commonly referred to as the Kubelka and Munk theory. The theory, which was built upon Schuster's two flux concept [51], differs from the Lambert-Bouguer law in at least two respects. (i) It models both absorption and scattering but only in the directions of an *incident flux* and a *reflected flux*. (ii) It assumes that a volumetric colorant layer can be divided into a large number of homogeneous elementary layers, the optical properties of the volume thus depend on one direction, say x. The two fluxes I_i and I_r flow in opposite directions, x and $-x$ respectively.

Consider an isotropic elementary layer of thickness dx, which is associated with an absorbing coefficient $K(\lambda)$ and a scattering coefficient $S(\lambda)$. With the incident flux $I_i(\lambda)$, the passage of a light beam through the layer will have its energy decreased through absorption, by an amount of $K(\lambda)I_i(\lambda)\,dx$, and through scattering, by an amount of $S(\lambda)I_i(\lambda)\,dx$. At the same time, because of the radiation reflected by the reflected flux $I_r(\lambda)$, the energy is also increased due to backscatter, by an amount of $S(\lambda)I_r(\lambda)\,dx$. This gives the total changes of the $I_i(\lambda)$ as:

$$dI_i(\lambda) = -K(\lambda)I_i(\lambda)\,dx - S(\lambda)I_i(\lambda)\,dx + S(\lambda)I_r(\lambda)\,dx. \tag{7}$$

On the other hand, the passage of the reflected flux, $I_r(\lambda)$, in the opposite direction, is also subject to similar changes, that is:

$$-dI_r(\lambda) = -K(\lambda)I_r(\lambda)\,dx - S(\lambda)I_r(\lambda)\,dx + S(\lambda)I_i(\lambda)\,dx. \tag{8}$$

Note that when $S(\lambda) = 0$, the incident flux, on integrating, follows the Lambert-Bouguer law discussed in 5.1. Also note that without the back-scattering, we would have $K(\lambda) + S(\lambda) = \tau(\lambda)$, where $\tau(\lambda)$ is the extinction coefficient in the Lambert-Bouguer law. As placement of the colorant layer and its elementary layers are normally drawn in a horizontal manner, the incident and reflected fluxes are commonly referred to as downward and upward fluxes respectively.

Dividing both sides of Equations (7) and (8) by dx, we have two simultaneous differential equations. The solution to the equations leads to several useful formulae. Consider a single homogeneous layer of thickness X. Let $R(\lambda)$ and $T(\lambda)$ be the reflectance and transmittance of the layer respectively. We have:

$$R(\lambda) = \frac{\sinh(b(\lambda)S(\lambda)X)}{a(\lambda)\sinh(b(\lambda)S(\lambda)X) + b(\lambda)\cosh(b(\lambda)S(\lambda)X)} \tag{9}$$

$$T(\lambda) = \frac{b(\lambda)}{a(\lambda)\sinh(b(\lambda)S(\lambda)X) + b(\lambda)\cosh(b(\lambda)S(\lambda)X)}, \tag{10}$$

where

$$a(\lambda) = \frac{S(\lambda) + K(\lambda)}{S(\lambda)}, \quad b(\lambda) = \sqrt{a(\lambda)^2 - 1}.$$

Here $a(\lambda)$ is essentially a spectral version of the albedo defined in 4.1, but only the probability of hitting back-scatters is considered. The reflectance of an opaque medium can thereby be computed by making $X \to \infty$ in Equation (9), resulting in:

$$R_\infty(\lambda) = 1 + \left(\frac{K(\lambda)}{S(\lambda)}\right) - \left[\left(\frac{K(\lambda)}{S(\lambda)}\right)^2 + 2\left(\frac{K(\lambda)}{S(\lambda)}\right)\right]^{\frac{1}{2}}.$$

Kubelka later extended the Kubelka-Munk theory to inhomogeneous layers [33]. Given the reflectance and transmittance of two different layers, R_1, T_1, R_2 and T_2, considering an infinite process of interaction between the two layers. A light flux passes the first layer with the portion T_1, and then the

second layer T_1T_2. Meanwhile, a portion of $T_1R_2T_1$ will be reflected from the second layer and pass though the first layer again. Continuing this process leads to two infinite series:

$$\begin{array}{llll} R_1, & T_1R_2T_1, & T_1R_2R_1R_2T_1, & \ldots \\ T_1T_2, & T_1R_2R_1T_2, & T_1R_2R_1R_2R_1T_2, & \ldots \end{array}$$

The infinite process of interaction can be considered as one-dimensional radiosity [12] — a limited form of global illumination. The sums of the two series give the compositing reflectance and transmittance as:

$$R(\lambda) = R_1(\lambda) + \frac{T_1(\lambda)^2 R_2(\lambda)}{1 - R_1(\lambda)R_2(\lambda)}$$
$$T(\lambda) = \frac{T_1(\lambda)T_2(\lambda)}{1 - R_1(\lambda)R_2(\lambda)}.$$

A discrete spectral volume rendering integral can thus be formulated based on this multi-layer model. Using the ray-casting method, one can approximate the intersection volume between each ray and a volume object as a series of homogeneous layers with a thickness equal to the sampling distance.

The work by Abdul-Rahman and Chen [1] highlighted some relative merits of the Kubelka-Munk theory over the Lambert-Bouguer law, especially in terms of its built-in distance attenuation, and its capability of determining the opacity and transparency optically according to the absorption and scattering properties. They have also experimented with captured optical properties of some real world materials, and the design for spectral transfer function for post-illumination.

6 Summary and Conclusions

Volume rendering is an indispensable tool for synthesizing images involved volumetric models. Much of the theoretic foundation was laid down in late 1980's and early 1990's [40]. Until recent years, however, most volume rendering systems employed only basic local illumination models. Due to the rapid increase of computation power, a collection of complex illumination features, such as shadows and refraction, have started to appear in some recent developments. This survey has provided an overview of optical and illumination models for volume rendering, while highlighting some new developments including spectral volume rendering, shadow algorithms and refraction rendering. We hope that this survey will encourage further research into the development and use of complex illumination models to achieve better realism and perception through optical correctness.

Acknowledgments

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract number W-7405-ENG-48. It was also partly funded by a UK-EPSRC grant with reference GR/S44198. The authors wish to thank Dr. Martin Kraus, Universität Stuttgart, for his comments and suggestions in the early stage of preparing this survey.

References

1 A. Abdul-Rahman and M. Chen. Spectral volume rendering based on the Kubelka-Munk theory. *Computer Graphics Forum*, 24(3), 2005.

2 James Arvo. Backward ray tracing. In *SIGGRAPH '86 Developments in Ray Tracing, Seminar Notes*, August 1986.

3 G. V. G. Baranoski and J. G. Rokne. An algorithmic reflectance and transmittance model for plant tissue. *Computer Graphics Forum*, 16(3):141–150, 1997.

4 B. G. Becker and N. L. Max. Smooth transitions between bump rendering algorithms. *ACM SIGGRAPH Computer Graphics*, 27(3):193–190, 1993.

5 Uwe Behrens and Ralf Ratering. Adding shadows to a texture-based volume renderer. In *Proceedings ACM SIGGRAPH Symposium on Volume Visualization*, pages 39–46, 1998.

6 M. J. Bentum, B. B. A. Lichtenbelt, and T. Malzbender. Frequency analysis of gradient estimators in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):242–254, 1996.

7 S. Bergner, T. Möller, M. S. Drew, and G. D. Finlayson. Internative spectral volume rendering. In *Proc. IEEE Visualization*, pages 101–108, 2002.

8 S. Bergner, T. Möller, M. Tory, and M.S. Drew. A practical approach to spectral volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):207–216, 2005.

9 James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.

10 J.F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics*, 16(3):21–29, 1982.

11 Carsten Dachsbacher and Marc Stamminger. Translucent shadow maps. In *Rendering Techniques 2003*, pages 197–201. Eurographics, 2003.

12 J. Dorsey and P. Hanrahan. Modeling and rendering of metallic patinas. In *Proc. 23rd Annual ACM SIGGRAPH Conference*, pages 387–396, 1996.

13 Klaus Engel, Markus Hadwiger, Joe Kniss, Christof Rezk-Salama, and Daniel Weiskopf. *Real-Time Volume Graphics*. A. K. Peters, 2006.

14 J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, 2nd edition, 1990.

15 Michael P. Garrity. Ray tracing irregular volume data. *ACM SIGGRAPH Computer Graphics*, 24(5):35–40, 1990.

16 D. P. Greenberg, J. Arvo, E. Lafortune, K. E. Torrance, J. A. Ferwerda, B. Walter, B. Trumbore, P. Shieley, S. Pattanaik, and S. Foo. A framework for realistic image synthesis. In *Proc. ACM SIGGRAPH Annual Conference Series*, pages 477–494, 1997.

17 Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of ACM SIGGRAPH Annual Conference*, pages 165–174, 1993.

18 Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics (SIGGRAPH '90)*, 24(4):145–154, 1990.

19 G. L. Henyey and J. L. Greenstein. Diffuse radiation in the galaxy. *Astrophysical Journal*, 88:70–73, 1940.

20 R. S. Hunter and R. W. Harold. *The Measurement of Appearance*. John Wiley & Sons, 2nd edition, 1987.

21 Akira Ishimaru. *Wave Propagation and Scattering in Random Media, Volume I: Single Scattering and Transport Theory*. Academic Press, New York, 1978.

22 Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics (SIGGRAPH '2002)*, 21(3):721–728, 2002.

23 Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of ACM SIGGRAPH Annual Conference*, pages 311–320, 1998.

24 Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of ACM SIGGRAPH Annual Conference*, pages 511–518, 2001.

25 J. T. Kajiya. Anisotropic reflection models. *ACM SIGGRAPH Computer Graphics*, 19(3):15–21, 1985.

26 J. T. Kajiya and B. P. von Herzen. Ray tracing volume densities. *ACM SIGGRAPH Computer Graphics*, 18(3):165–174, 1984.

27 Gordon Kindleman and James Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Proceedings, IEEE/ACM Symposium on Volume Visualization*, pages 79–86, 1998.

28 Robert M. Kirby and Blake Nelson. Ray-tracing polymorphic multi-domain spectral/hp elements for isosurface rendering. *IEEE Transcations on Visualization and Computer Graphics*, 12(1), 2006.

29 Joe Kniss, Gordon Kindlmann, and Charles Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

30 Joe Kniss, Simon Premoze, Charles Hansen, and David Ebert. Interactive translucent volume rendering and procedural modeling. In *Proceedings, IEEE Visualization*, pages 109–116, Washington, DC, USA, 2002. IEEE Computer Society.

31 Joe Kniss, Christof Rezk-Salama, Klaus Engel, and Markus Hadwiger. High-quality volume graphics on consumer PC hardware. In *SIGGRAPH '2002 Course*, 2002.

32 Martin Kraus. Scale-invariant volume rendering. In *Proceedings, IEEE Visualization*, pages 295–302, 2005.

33 P. Kubelka. New contributions to the optics of intensely light-scattering materials, Part II. nonhomogeneous layers. *Journal of the Optical Society of America*, 44:330 – 355, 1954.

34 P. Kubelka and F. Munk. Ein Beitrag zur Optik der Farbanstriche. *Zeitschrift für Technische Physik*, 12:593–601, 1931.

35 Chang Ha Lee, Xuejun Hao, and Amitabh Varshney. Light collages: Lighting design for effective visualization. In *Proc. IEEE Visualization*, pages 281–288, 2004.

36 M. Levoy. Volume rendering: display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

37 Shengying Li and Klaus Mueller. Accelerated, high-quality refraction computations for volume graphics. In *Proc. Volume Graphics*, pages 73–81. Eurographics, 2005.

38 Shengying Li and Klaus Mueller. Spline-based gradient filters for high-quality refraction computation in discrete datasets. In *Proc. EuroVis*, pages 215–222. Eurographics, 2005.

39 Eric B. Lum and Kwan-Liu Ma. Lighting transfer functions using gradient aligned sampling. In *Proc. IEEE Visualization*, pages 289–296, 2004.

40 N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

41 T. H. Meyer, M. Eriksson, and R. C. Maggio. Gradient estimation from irregularly spaced data sets. *Mathematical Geology*, 33(6):693–717, 2001.

42 Torsten Möller, Raghu Machiraju, Klaus Mueller, and Roni Yagel. A comparison of normal estimation schemes. In *Proc. IEEE Visualization '97*, pages 19–26, Phoenix, AZ, November 1997.

43 Kenneth Moreland and Edward Angel. A fast high accuracy volume renderer for unstructured data. In *Proceedings of the IEEE/SIGGRAPH Symposium on Volume Visualization*, pages 9–16, 2004.

44 L. Neumann, B. Csébfalvi, A. König, and E. Gröller. Gradient estimation in volume data using 4D linear regression. *Computer Graphics Forum*, 19(3):C351–C357, 2000.

45 S. K. Park and R. A. Schowengerdt. Image reconstruction by parametric cubic convolution. *Computer Vision, Graphics and Image Processing*, 23:258–272, 1983.

46 Frederic Pérez, Xaview Pueyo, and François Sillion. Global illumination techniques for the simulation of participating media. In Julie Dorsey and Philipp Slusallek, editors, *Rendering Techniques '97*, pages 309–320, Vienna, 1997. Springer.

47 A. Pommert, U. Tiede, G. Wiebecke, and K. H. Hohne. Surface shading in tomographic volume visualization. In *Proc. 1st Conference on Visualization in Biomedical Computing*, volume 1, pages 19–26, 1990.

48 William Press, Saul Teukolosky, William Vettering, and Brian Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2002.

49 David Rodgman and Min Chen. Refraction in discrete raytracing. In Klaus Mueller and Arie Kaufman, editors, *Proc. Volume Graphics 2001*, pages 3–17. Springer, 2001.

50 Stefan Röttger, Martin Kraus, and Thomas Ertl. Hardware-accelerated volume and isosurfaces rendering based on cell projection. In *Proc. IEEE Visualization*, pages 109–116, 2000.

51 A. Schuster. Radiation through a foggy atmosphere. *Journal of Astrophysics*, 21:1–22, 1905.

52 Peter Shirley and Alan Tuchman. A polygonal approximation to direct scalar volume rendering. *ACM SIGGRAPH Computer Graphics*, 24(5):63–70, 1990.

53 Jos Stam. Multiple scattering as a diffusion process. In Patrick Hanrahan and Werner Purgathofer, editors, *Proc. Eurographics Workshop on Rendering Techniques*, pages 41–50. Springer, 1995.

54 A. J. Stewart. Vicinity shading for enhanced perception of volumetric data. In *Proc. IEEE Visualization*, pages 355–362, 2003.

55 Lifeng Wang, Wenle Wang, Julie Dorsey, Xu Yang, Baining Guo, and Heung-Yeung Shum. Real-time rendering of plant leaves. *ACM Transactions on Graphics (SIGGRAPH '2005)*, 24(3):712–719, 2005.

56 Manfred Weiler, Martin Kraus, Markus Merts, , and Thomas Ertl. Hardware-based ray casting for tetrahedral meshes. In *Proc. IEEE Visualization*, pages 333–340, 2003.

57 David Wiley, Hank Childs, Bernd Hamann, and Kenneth Joy. Ray casting curved-quadratic elements. In *Proc. Eurographics/IEEE TCVG Data Visualization*, pages 201–209, 2003.

58 Peter Williams and Nelson Max. A volume density optical model. In *Proceedings, ACM SIGGRAPH Workshop on Volume Visualization*, pages 61–68, 1992.

59 Peter Williams, Nelson Max, and Clifford Stein. A high accuracy renderer for unstructured data. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):37–54, 1998.

60 Brian Wylie, Kenneth Moreland, Lee Ann Fisk, and Patricia Crossno. Tetrahedral projection using vertex shaders. In *Proceedings, ACM SIGGRAPH Workshop on Volume Visualization*, pages 7–12, 2002.

61 G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & Sons, 1982.

62 Ciaxia Zhang, Daqing Xue, and Roger Crawfis. Light propagation for mixed polygonal and volumetric data. In *Proceedings, Computer Graphics International*, pages 249–256. IEEE, 2005.

63 S. W. Zucker and R. A. Hummel. A three-dimensional edge operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(3):324–331, 1981.

Real-time Terrain Mapping

Tony Bernardin[1], Eric Cowgil[2], Ryan Gold[2], Bernd Hamann[3], Oliver Kreylos[3], and Alfred Schmitt[1]

1 **Institut für Betriebs- und Dialogsysteme, Universität Karlsruhe**
{ug1g,aschmitt}@rz.uni-karlsruhe.de
2 **Department of Geology, University of California, Davis**
{cowgill,gold}@geology.ucdavis.edu
3 **Institute for Data Analysis and Visualization, University of California, Davis**
{hamann,kreylos}@cs.ucdavis.edu

Abstract

We present an interactive, real-time mapping system for digital elevation maps (DEMs), which allows Earth scientists to map and therefore understand the deformation of the continental crust at length scales of 10 m to 1000 km. Our system visualizes the surface of the Earth as a 3D surface generated from a DEM, with a color texture generated from a registered multispectral image and vector-based mapping elements draped over it. We use a quadtree-based multiresolution method to be able to render high-resolution terrain mapping data sets of large spatial regions in real time. The main strength of our system is the combination of interactive rendering and interactive mapping directly onto the 3D surface, with the ability to navigate the terrain and to change viewpoints arbitrarily during mapping. User studies and comparisons with commercially available mapping software show that our system improves mapping accuracy and efficiency, and also enables qualitatively different observations that are not possible to make with existing systems.

1998 ACM Subject Classification I.3.7 Three-Dimensional Graphics and Realism, J.2 Physical Sciences and Engineering

Keywords and phrases Earth, Space, and Environmental Sciences Visualization, Interaction, Terrain Visualization, Multiresolution Visualization

Digital Object Identifier 10.4230/DFU.SciViz.2010.275

1 Introduction

Understanding how continents deform is a fundamental problem in Earth science [2]. Although the plate tectonic paradigm provides an accurate description of the behavior of oceanic crust, the theory fails to fully explain strain distribution within continents. There are currently two end-member views of the problem [3]. In one, continental deformation is spatially distributed over thousands of kilometers and thus, fundamentally differs from the plate-like behavior of the oceanic crust. In a second view, continents are mosaics of strong, rigid blocks bounded by weak faults, and thus mimic the behavior of oceanic plates. Distinguishing between these two views centers on determining the geometric and mechanical evolution of first-order ($\approx$ 1000 km long) intracontinental structural systems [3]. Do these systems of faults and folds remain stable in space and time for tens of millions of years (oceanic-plate like), or do they migrate, eventually producing spatially distributed deformation zones (diffuse pattern)?

Addressing this problem centers on determining how these 1000 km-long structural systems form and evolve over geologically intermediate time scales of a few tens of thousands to a few million years. At its core, developing this understanding requires mapping these

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 275–288

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

structural systems. Specifically, mapping means identifying and measuring the 3D orientation of surfaces such as faults and folded layers of rock, along with the various topographic features by which these structures may be identified, such as contorted drainage networks or displaced ridges. Short-term deformation of a few seconds to a few thousands of years is straightforward to characterize using active, historical or paleo earthquakes as well as geodetic techniques such as VLBI, GPS, or InSAR. Likewise, long-term deformation that accumulates over tens of millions of years can be measured using thermochronology and geochronology. But understanding the intermediate record has proven difficult because it has been difficult to map.

At such intermediate time scales, active deformation of the Earth's surface produces detailed (10 m–100 m) topographic features by which active structures may be identified and mapped. However, such mapping has been hampered by the lack of both data and tools that permit efficient analysis of those data over the 1000 km × 1000 km areas that span regions of active continental deformation. In the last few years the first of these problems has been addressed and geologists now have access to non-classified intermediate (10 m–20 m) and high (1 m–10 m) pixel resolution DEMs and multispectral satellite or photographic imagery. The sudden availability of these new datasets has amplified the need within the earth sciences community for straightforward tools that provide both efficient visualization of gigabyte to terabyte datasets and geological mapping within an interactive, 3D visualization environment. The problem of interactive, 3D visualization of large datasets has been previously addressed using multi-resolution and level of detail techniques [8, 7]. We expand that environment to allow users to map on the 3D surface and compare our new application with recently developed alternative approaches based on 3D photogrammetric techniques.

Although active structures typically deform the Earth's surface, this surface is also under constant attack by geomorphic processes that either destroy it via erosion or bury it via deposition. The competition between the rates of surface deformation and destruction results in a characteristic length scale for preserved deformed geomorphic markers, such that markers that are a few thousand to a few million years old will typically have spatial dimensions of a few tens to a few thousands of meters. This spatial range requires geologists to make numerous detailed observations to fully map active structures, and therefore places a fundamental limit on the data resolution needed to study these markers. In particular, multispectral imagery and digital terrain data must have resolutions not coarser than 15 m and 30 m, respectively. However, because the first-order structural systems are typically on the order of 1000 km long, geologists must also make these detailed observations over very large areas. In addition, many of the areas of active continental deformation lie in Africa, Asia and the Middle East, where data are incomplete and/or of variable quality.

Geologists have struggled with the dilemma of making detailed, remote observations over large areas for some time. One compromise is to use low-resolution imagery, and a second is to conduct detailed investigations of small (10 km × 10 km) areas at a few, widely spaced localities. Both methods give a strongly filtered view of the active deformation field. As a result, the geomorphic record of neotectonic deformation, and thus our understanding of how major structural systems evolve at intermediate time scales, remains largely unexplored. In response, we have developed RIMS, a Real-time, Interactive, Mapping System. RIMS allows geologists to visualize, and map in 3D space, structures that are active at intermediate time scales, both in detail but also over thousand-kilometer wide zones of active continental deformation.

2 Related Work

Prior to the development of RIMS, there were two basic methods for analysis of high resolution, multispectral imagery and digital terrain data available to geologists. One option was perspective views of texture data such as imagery draped over a DEM. A large number of widely available tools allow interactive manipulation of such displays [6, 9, 10, 11, 12, 13]. Most of them do not yet appear to make use of multiresolution techniques [14]. More importantly, it is not possible to map directly on the 3D scene. The second approach has been to perform 3D feature extraction using a StereoGraphics Z-Screen and a photogrammetry package [15, 16, 17].

2.1 Terrain Level-of-Detail (LOD) Algorithms

Multi-resolution visualization of large-scale 3D terrain models is an active area of research. [1] have recently presented a technique using triangulated irregular network (TIN) patches as drawing primitives in place of triangles. TIN patches represent terrain highly accurately and are optimized in a preprocessing step for efficient storage and rendering and are batched to the graphics hardware by a view-dependent algorithm. In addition, a texture tile hierarchy is constructed to allow for multi-resolution imagery to be projected onto the terrain geometry. [4] and [8] work with regular gridded data, considerably reducing the preprocessing requirement, and present data management and view-dependent algorithms focusing on the real-time generation of the triangulated mesh representation of the terrain. The former method focuses on attaining the best fidelity in the generated approximation using complex algorithms, whereas the latter method concentrates on the simplicity and ease of implementation of the data management and simplification techniques. All the mentioned techniques feature out-of-core data management enabling them to process the large data sets associated with terrain visualization.

The terrain visualization component of our system is based on the algorithm of [7]. Quadtrees are a well-understood data structuring concept we use to uniformly represent and manage the different components of our system (geometry, texture and mappings).

2.2 Vector Mapping and Display Systems

One of us (Cowgill) has recently developed a method for building digital stereo models (DSMs) from 15 m-pixel resolution ASTER stereoscopic imagery using only the ground control point information provided with the Level 1A data. Individual DSMs comprise 4200×4200 pixel ($\approx 60\,\text{km} \times 60\,\text{km}$) images that can be mapped in 3D using StereoAnalyst (SA) [15] at scales up to 1:20,000 ($\approx 6^2$ screen pixels to 1 image pixel). This system is limited to a plan or bird's eye view and does not permit a user to view the surface along a vertical cross-section, which is a perspective that geologists rely upon heavily for analysis. In addition, the DSMs typically have lateral variations in X and Y parallax, resulting in eye strain after a few hours of analysis with SA. Finally, the lack of an ASTER-specific sensor model and external ground control information restricts the DSM to a single scene, thus only a $60\,\text{km} \times 60\,\text{km}$ area can be mapped at once using SA.

To address the visualization problem of draping 2D vector data over a multi-resolution 3D terrain representation, [18] propose algorithms for rendering geometric lines adapted to surface tessellation. Their method handles sophisticated restricted quadtree triangulations where the representation is not fixed for a given node. Our approach employs fixed regular triangulations per node and allows for the more straightforward method presented in Sect. 4.1.2.

As an alternative to the polyline-as-geometry approach mentioned above, [5] present techniques to render vector data using textures generated on-demand. In hopes of being less sensitive to vector data quantity, we would like to add such a polyline-as-texture approach to our system in the future.

3 Terrain Visualization

To be useful for netotectonic studies, a 3D mapping system must provide interactive textured height field rendering of very large terrain data sets (above 35 k × 35 k samples) with full roaming and viewpoint manipulation, and at the same time must provide interactive mapping of attributed points, polylines and polygons directly on the 3D terrain model. When zoomed in to large magnifications, the system must display height field and texture at full resolution. The system must be able to import and export georeferenced mapping data from and to standard GIS applications. Ideally, the system should be able to manipulate the viewpoint at any time while placing mapping elements.

3.1 Terrain Representation

Unprocessed terrain data sets are commonly maintained as two-dimensional single or multi-band images. In our case, one gray-scale DEM represents the height measurements for a given longitude/latitude sampling, and other three-channel images contain the corresponding color-information – typically either false-colors generated from the height or spectral-band interpretations, or actual aerial photographs. The first main task of our system is to facilitate visual exploration of such data sets by constructing and rendering appropriate 3D surface representations.

3.1.1 Quadtrees

For rendering purposes, a terrain data set's surface is reconstructed using triangle drawing primitives. When rendered directly, the large and high-resolution terrain data sets required by the application would far exceed current graphics hardware capabilities, prohibiting real-time exploration. To address this problem, we use multi-resolution representations based on a quadtree subdivision. A quadtree's lowest-level nodes correspond to a tiling of a given data set at its native resolution. Higher-level nodes contain successively coarser representations, subsampled by a factor of two between levels. All nodes in the same tree have a fixed size of some power of two in each dimension. By sending only appropriate nodes to the graphics hardware (see Sect. 3.1.2), we can render very large data sets in real time while maintaining sufficient detail for mapping purposes. When treating each tree node as an atomic entity, the quadtree representation enables efficient frame processing, e. g., hierarchical view culling (see Sect. 3.1.3, selecting appropriate detail level (see Sect. 3.1.2), and computing mapping element representations (see Sect. 4.1.2).

In our system, terrain models are represented by a set of correlated quadtrees, the first one containing the terrain's height values, and one or more additional ones containing texture data. These trees are generated from the unprocessed data in a preprocessing step (see Sect. 3.2). Considering that the original height and texture data are tightly correlated, the height and texture trees will present a node-to-node matching if generated with appropriate node sizes. For example, using a 30 m resolution DEM with a 15 m resolution texture draped over it would require a texture quad twice the size of a height quad, e. g., 128 and 64, respectively. This tight coupling prompted us to merge the two trees leaving a single terrain

tree that needs to be maintained and processed each frame: each of its nodes contains references to corresponding height and texture data (see Fig. 1 left). We expect this structure to also facilitate future out-of-core management/caching and allow for quick overlaying of compatible preprocessed textures.

3.1.2 Level-of-Detail Calculation

To effectively exploit the multi-resolution terrain representation, we need a means of evaluating the *appropriateness* of a node based on a set of frame conditions. We consider a continuous LOD value characterizing a node as perfectly fitting the conditions if it is zero. Negative values progressively indicate the node's resolution to be too coarse and positive values indicitate it being too fine. Additionally, we choose a *split-threshold* below which nodes will be considered for subdivision and a *merge-threshold* above which merging is suggested (see Sect. 3.1.3). We make the following considerations for our LOD evaluation:

Focus and Context

When mapping, users operate locally on the terrain data, effectively defining the region of interest. We consider each evaluated node's distance from the manipulation cursor as a first LOD value: at the cursor location, the finest detail is desired and the farther away from the focus point, the coarser we are allowed to represent the terrain (see Fig. 1, top-right).

View-Distance Dependency

To explore the data, users are constantly manipulating view parameters that in turn affect the projection of terrain tree nodes into screen space. To account for such view-specific characteristics we consider a node's projected pixel coverage using [8] estimation of perspective projection dependent on the distance between the node and the viewing camera. The node's world-space edge-size is projected and compared to a chosen optimum for a second LOD value: for a given view, the finest detail level discernable on screen (as specified by the optimum) is chosen (see Fig. 1, middle-right).

In the end, the final LOD value is computed by combining the two previous ones: the *focus-LOD* sets the coarsest bound, meaning that although the view would allow for more detail to be displayed on the screen, for a node away from the focus point this does not currently interest the user. On the other hand, the *view-LOD* specifies the finest bound, in that even for the node directly under the focus point we need only render as much detail as will be discernable in the screen projection (see Fig. 1, bottom-right).

3.1.3 Tree Maintenance

Ideally, the *active* tree nodes chosen for a terrain approximation would be those with a LOD value evaluated between the split and merge thresholds. Refining recursively by starting with the root node and subdividing nodes whose LOD values lie below the split-threshold would in fact result in a set of leaf nodes defining a gap-less, overlay-free tiling of the terrain area. However, cracks might appear between neighboring nodes of different resolutions due to hanging triangle vertices (see Fig. 2). To address this problem, we modify the LOD criterion such that direct neighbors in the active set differ at most by one level of resolution. If this property holds for an entire tree, cracks can be removed by simple *stitching* at the edges of affected nodes.

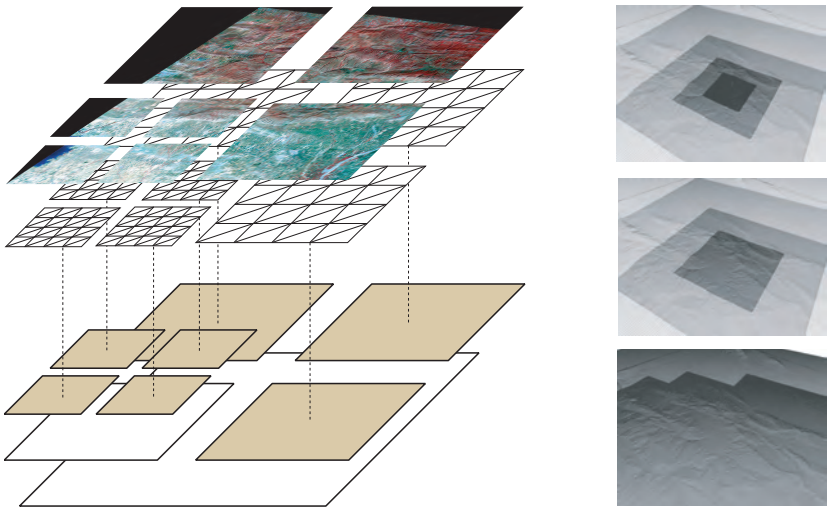


Figure 1 Left: Terrain tree with height and texture data. *Active* tiles belonging to the current approximation are colored, and their geometry and texture data quads are shown. Right: Level-of-detail computation. Top: LOD is calculated based solely on the focus point (note the overly detailed square in the middle). Middle: LOD is determined only using the view parameters. Bottom: Focus and view LODs are combined

Splitting and Merging

Since users exploring terrain roam interactively, view parameters change little from one frame to the next. A high inter-frame coherence can be expected and an approach similar to the one followed by [4] is worthwhile. Instead of generating the appropriate representation by recursive subdivision of the root (with the necessary balancing performed on the fly), the previous representation is *tweaked* to conform to the new frame conditions. Our current implementation dedicates a first traversal pass to tree maintenance using split and merge operations, but the task could easily be left to the care of an independent thread.

The tree maintenace pass initially evaluates the LOD values of all previously active nodes. Their new LOD values are then compared to the thresholds in order to decide whether nodes should be kept as they are, split into their children, or merged with their siblings. Both the split and merge operations assume a valid one-level difference terrain tree and result in a similarly valid one. The split operation does so by recursively forcing the considered node's neighborhood to subdivide appropriately, whereas the merge operation only succeeds if the siblings and neighborhood allow for the parent to become a leaf. This scheme favors showing detail over hiding it. Additionally, before either operation can be completed, the stitching attributes for the inserted node(s) and the neighborhood have to be corrected, i. e., coarser nodes have to be adapted to neighboring finer ones. We use four bit-flags, one for each edge, specifying if the corresponding edge connects to a more detailed one. We describe how these bit-flags relate to actual mesh approximations in Sect. 3.3.

View Frustum Culling

By taking advantage of the quadtree structure, we can further reduce maintenance costs (and later rendering time) using hierarchical view frustum culling. The tree maintenance pass traverses the terrain tree depth-first from the root until it encounters a node that is outside the current view frustum or a leaf node in the current approximation (as determined by the

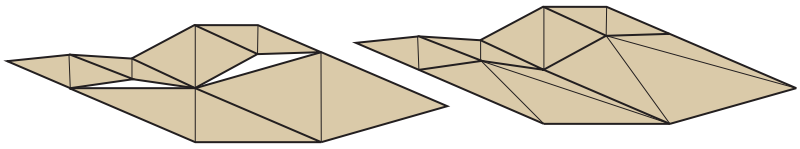


Figure 2 Stitching between nodes. Left: Cracks can appear at the edge of neighboring nodes of different resolutions. Right: Stitching adapts lower-resolution nodes to higher-resolution ones for smooth transitions.

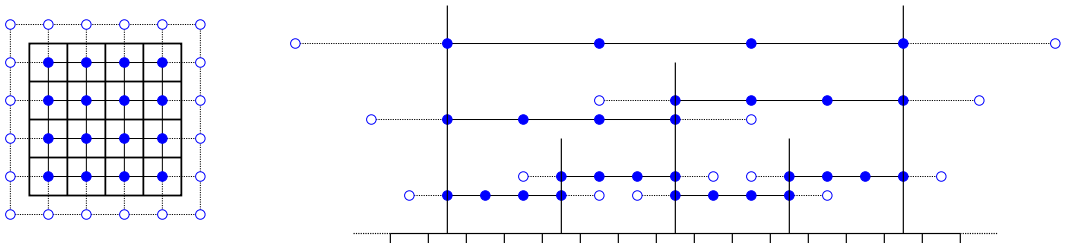


Figure 3 Quadbase preprocessing. Left: Alignment of vertex positions in a height quad (blue dots) and texels in a texture quad (bold squares). Right: Alignment between input height data, height quads in the same level, and height quads between levels. Black tick marks denote pixel borders of input data, blue dots denote vertex positions in height quads, hollow dots denote "ghost vertices" around height quads.

split and merge criteria described above). Computing the visibility of a node is done by intersecting its bounding sphere with the view frustum. This check is very inexpensive, and enables efficient culling of entire subtrees from the maintenance traversal if an upper-level node is invisible. We maintain a visibility flag in each node to forward the results obtained here to the rendering phase (see Sect. 3.3).

3.2 Quadbase Preprocessing

Available terrain data sets usually describe a continuous area at a given resolution, whereas our program requires a multi-resolution hierarchical tiling of that area. We generate the needed tiles off-line with our preprocessing tool and store them in a binary *quadbase* file. The preprocessor first constructs a skeletal quadtree with the property that its leaf nodes tile the input data set at its native resolution, the root node entirely covers the input data set's domain, and only those nodes intersecting the domain are retained. The skeletal tree is then traversed in a bottom-up, breadth-first fashion. At each level, each node crops out the data associated with it and appends it to the quadbase file. After all nodes in a level are processed, the input data is resampled to the resolution appropriate for the next higher level. To associate the image data with mesh geometry we place vertices at the centers of pixels (see Fig. 3, left). Therefore, care must be taken to duplicate quad edge pixels where vertices are shared for rendering. Moreover, quads produced to store height information should additionally store border pixels to facilitate generation of vertex normals later on (see Fig. 3, right).

Descriptive information for both the quadtree, e. g., quad resolution and number of quads, and the contained data, e. g., upper-left corner longitude/latitude and data resolution, is stored in an additional quadbase header file.

3.3 Rendering

Mesh Representation

Whereas the preprocessed texture quads can be used directly as sources for texture objects, the height quads have to be converted into triangulated patches of 3D vertices. The vertex positions and texture coordinates are generated by creating a planar regular grid where (x, y)-points are elevated using the appropriate pixel value of the height quad. Vertex texture coordinates are calculated by linearly mapping (x, y)-coordinates into the associated texture quad's texture rectangle, which is identical for each texture node. The only considerable computation comes from running a filter on a height pixel's neighborhood to obtain vertex normals for rendering. After being computed, the position, texture coordinates and normals are stored in memory compactly as an interleaved vertex array. We have chosen to omit this step from the preprocessing to keep the input data as general and independent of internal geometric representation as possible. This approach also reduces I/O volume, making us less dependent on slow reads from disk.

We employ a simple caching scheme for node geometry and texture data, to circumvent having to wait for disk I/O when a previously active node becomes activated again. This caching scheme can be enhanced for full data size-independent out-of-core rendering for very large terrain data and limited main memory.

Rendering pass

The image corresponding to the current terrain representation is produced in a second pass through the terrain tree. Similar to tree maintenance, a separate thread could be assigned this task, refreshing at the graphics hardware's rate instead of the I/O-bound update thread's rate. A depth-first traversal from the root finds the active nodes of the current approximation and renders them, exploiting the hierarchical view culling maintained in the visibility bit-flag (see Sect. 3.1.3). Additionally, we could use the quadtree structure to always draw the nodes in front-to-back order and take advantage of the graphics hardware's depth buffer culling.

In Sect. 3.1.3, we mention the need for neighboring rendered quads to align without cracks, even when they do not represent the same level of detail. This affects the triangulations that have to be generated: with one level of difference maximally possible between neighbors, we can identify fifteen different stitching cases. For each case, we pre-compute a static index array defining appropriate triangle strips over the vertex grid. To render a node, its vertex data can then efficiently be sent to the graphics hardware with the appropriate index array for the node's stitching flag computed by the tree maintenance traversal (see Sect. 3.1.3).

4 Mapping

The real-time rendering provided by RIMS constitutes a highly valuable tool for terrain data exploration. However, textured 3D representations are already available in common commercial software (albeit not using multi-resolution approaches yet) and many advanced techniques have been published. More important for our purposes is the use of the 3D terrain model to directly and efficiently specify and edit georeferenced mapping elements. The following section presents our program's mapping capabilities.

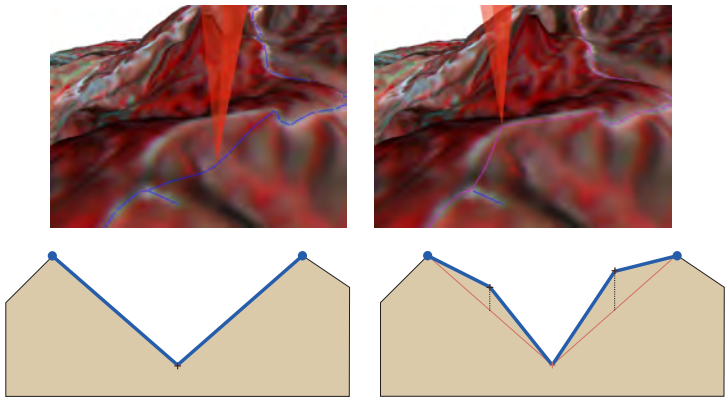


Figure 4 Top: Refining a polyline by inserting a new control point. Bottom: 3D polyline representation. Left: Line strips connecting the segment control points follow the terrain topology. Right: Mesh representation has changed showing more detail, thus hiding the old line strips (red). A new line strip has to be generated connecting the same control points.

Specifying 2.5D Mappings

Typically, mapping data is two-dimensional, e. g., a polyline would be specified as a list of (longitude, latitude) control points. Our mapping tools conceptually operate on a 2D plane by keeping this representation and dynamically assigning appropriate height values to all control points. This approach allows for mappings to be defined independently of the current 3D terrain approximation which, in our case, is constantly changing. Interfacing with common GIS packages can then also be realized easily: our system supports the ASCII ARC/INFO interchange file format for imports and exports.

Most commonly, geologists highlight features using a connected sequences of line segments, i. e., polylines. Our system supports mapping with this primitive: controlling a cursor bound to the terrain surface as a spatial reference, users can perform various actions such as creating, selecting, moving and deleting control points (see Fig. 4, top).

4.1 Polyline Rendering

To display polylines we take a line-as-geometry approach similar to [18]. Combining such an approach with the multi-resolution 3D terrain representation requires “lifting” polylines to the 3D terrain model appropriatedly to avoid clipping with the terrain geometry (see Fig. 4, bottom). In the following, we describe processing the polyline approximation in detail.

4.1.1 General Handling

Geometric lines, our display primitives, can only accurately follow flat surfaces, like those defined by the triangles of the 3D terrain representation. Thus, each 2D polyline segment – specified by a pair of 2D control points – has to be represented by a sequence of 3D line segments, one for each triangle intersected by the 2D polyline segment. Re-computing the appropriate 3D vertices for each frame would dramatically reduce the amount of segments that can be visualized interactively. To address this limitation, we exploit the locality of polyline manipulations (moving an inner control point, for example, modifies at most two segments) and the strong frame-to-frame coherence (triangulations will only change for few quads in each frame) by storing 3D representations for all polylines, and *tweaking* previously valid representations when polyline segments are edited, or the terrain approximation changes.

Table 1 Data set sizes (in pixels for DEM and texture), preprocessing times (in seconds) and frame rates (in frames per second) for the three test data sets.

Data Set	DEM Size	Tex Size	Build	Min. fps	Avg. fps	Max. fps
Aksai	1850×900	3700×1800	6 s	41.2	141.6	285.7
Mosul	2558×2447	5115×4901	22 s	60.6	130.0	400.0
Puget Sound	8193×8193	16384×16384	750 s	30.1	94.2	285.7

A polyline is represented as a list of subsegments, such that each subsegment is contained in a single currently active quadtree node. When a polyline is created or manipulated, the sequence of subsegments is computed by clipping the 2D polyline against the domains of all active nodes it intersects. Each active node also stores a list of subsegments associates with it, such that when a node splits or is merged with its neighbors, the affected polyline subsegments can be determined efficiently, and replaced with new ones appropriate for the changed set of active nodes.

4.1.2 Subsegment Computation

The dominating computational cost of visualizing a polyline lies in the generation of line strips for each of its subsegments, i. e., for each polyline part contained in an active quad of the current terrain approximation; thus, a fast technique is required to maintain high frame rates. In our case, this is facilitated by the regular triangulations within each quad. Moreover, computing the subsegment vertices is effectively only to a 2D problem: since all vertices of the 3D terrain approximation are extruded from the (x, y)-plane along the same direction, we "flatten" them back onto the plane containing the 2D polylines. Intersection points can then be computed and subsequently extruded appropriately. Thus, a very simple algorithm similar to those used to rasterize lines to a regular pixel grid can be used with few modifications.

5 Results

To evaluate RIMS' performance, we simulated mapping usage on three test data sets of different sizes. The data set sizes (as DEM size and texture size) and the preprocessing times necessary to create the hierarchical quadbases from the input data sets are given in Table 1, as well as the minimum, average and maximum frame rates achieved during mapping.

To evaluate the utility of RIMS, we conducted two comparison tests between the RIMS and StereoAnalyst (SA) [15] mapping methods. The first test (see Fig.5 and 6) compares the maximum level of geological detail that can be extracted from the data to identify the mapping system with the highest sensitivity to detail. Geologists seek the most sensitive system because it allows them to extract the largest amount of information and thus develop the most sophisticated geological analysis. For this test, a user spent as much time as needed to extract the maximum number of features over the same area. The second test (see Fig. 7) compares the number and quality of geologic observations that can be collected in the same finite period of time to identify the most efficient mapping system. Geologists prefer highly efficient systems that allow them to process their data as quickly as possible. For this test, a user spent two hours mapping the same area. In both tests, the study areas were mapped first with SA, then with RIMS. This approach was admittedly biased, because the users had the benefit of already having mapped the scene once at the start of their RIMS sessions. However, both users are significantly more familiar with the SA navigation/mapping environment;

thus, their lack of familiarity with the RIMS controls likely offsets any advantage their prior SA mapping provided.

The tests indicated that RIMS provides a number of user benefits, including reduced eye strain, faster zoom and pan speeds, and slight advantages in the navigation. More importantly, the tests also revealed five key differences that make RIMS more useful for geological applications. Relative to SA, RIMS provided greater 1) understanding of the mapped structural geometry and thus pattern of active deformation; 2) confidence in feature identification and location; 3) numbers of mapped features (i. e., a larger number of mapping elements); 4) mapping accuracy (i. e., a larger number of vertices per mapping element); and 5) ability to locate and identify small features. Specific examples of each difference are provided in the following sections, highlighting the utility of RIMS.

1. The most important difference revealed by the tests is that RIMS allowed both users to obtain a more sophisticated understanding of the structural geometry of their areas. For example, in Fig. 5, arrow A' on the right-hand side of the figure points to a structure that was obvious in the RIMS environment. The lack of a structure at arrow A on the left-hand side of the figure indicates that the user was not able to see and interpret this feature using SA. A RIMS screen shot (see Fig. 6, right half) clearly shows the structure mapped at A', and also demonstrates that it appears as an uninterpretable bump in a plan-view stereo pair that replicates the view from SA (see Fig. 6, left half). Likewise, additional structures were discovered at B' and C' using RIMS while the corresponding points B and C indicate that the user missed these features when using SA. In summary, the plan (bird's eye) view and grayscale imagery of SA made it difficult to identify the topographic and textural variations that indicated the existence of these subtle features.
2. RIMS provided both users with higher confidence in their vector mapping, as indicated by the type of lines selected to represent mapped features. Geologists express their confidence in their ability to accurately locate a mapped feature by using solid, dashed, or dotted lines (in order of decreasing confidence). Fig. 7 shows that the RIMS project contains 20 boundaries mapped using solid lines, 2 using dashed, and 2 with dotted. In contrast, the SA project has only 2 boundaries defined with solid lines, 21 with dashed lines, and 1 with a dotted line.
3. Both users were able to identify a larger number of features using RIMS than SA. The RIMS output shown in Fig. 5 has 289 mapped features whereas only 172 features were extracted using SA. Likewise, Fig. 7 indicates that 14 major structures were defined using RIMS, in contrast to 8 structures on the SA map.
4. RIMS allows users to more accurately locate features and then map them using more vertices per feature because it does not demand constant manual parallax adjustments. Because the polylines have more vertices in the RIMS outputs, they better track short wavelength variations in the feature geometry and thus more accurately follow subtle changes in the boundaries between geologic units. In contrast, the maps generated from SA show a prevalence of long straight line segments. Differences in detail are especially evident in Fig. 7 at comparison points A–A', B–B' and C–C' in SA and RIMS, respectively.
5. Finally, RIMS is more effective for locating small geologic features. For example, a series of river terraces located at point D' in the RIMS output were not located at point A using SA (see Fig. 7). Likewise, points E–E' indicate a small outcrop that was not seen in SA at E but that was mappable using RIMS at E'. Although these features are small, their identification has important implications regarding the geometry of active deformation in the mapped area.

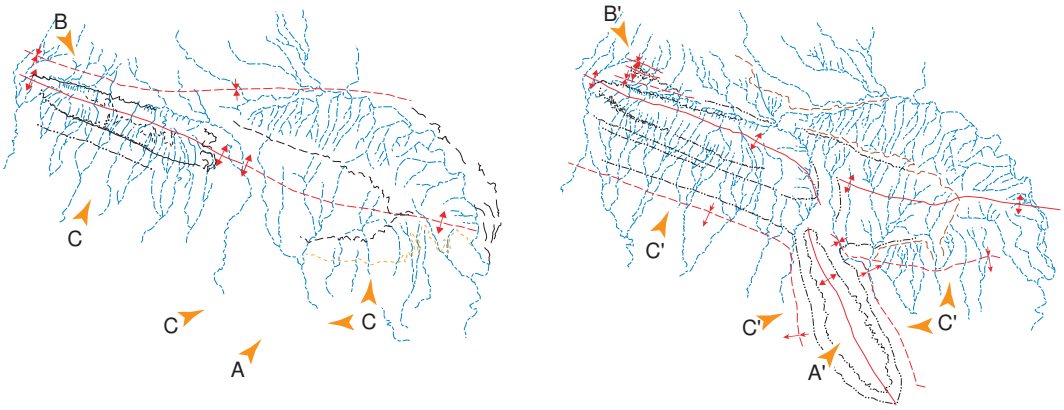


Figure 5 Results of sensitivity test. Gold arrows highlight points where the maps differ significantly, as discussed in the text. Red lines are fold hingelines and are solid where confidently located and dashed where their position is less clear. Blue lines denote drainages. Broken black lines indicate contacts between two different geologic units, dotted black lines are marker beds. Dashed yellow line denotes the edge of a geomorphic surface. Brown lines indicate drainage divides. Left: Map generated using StereoAnalyst. Right: Map generated using RIMS.

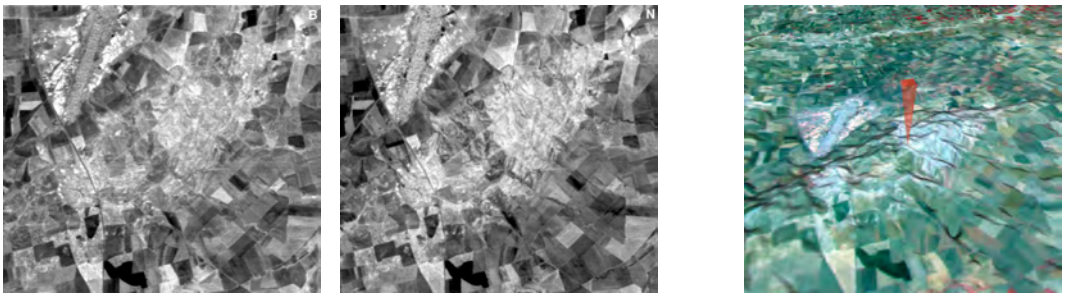


Figure 6 Subtle ridge appearing at location A–A' in Fig. 5. Left and center: Cross-eye stereo pair reconstructing the plan view provided by StereoAnalyst. Right: Screen shot from RIMS.

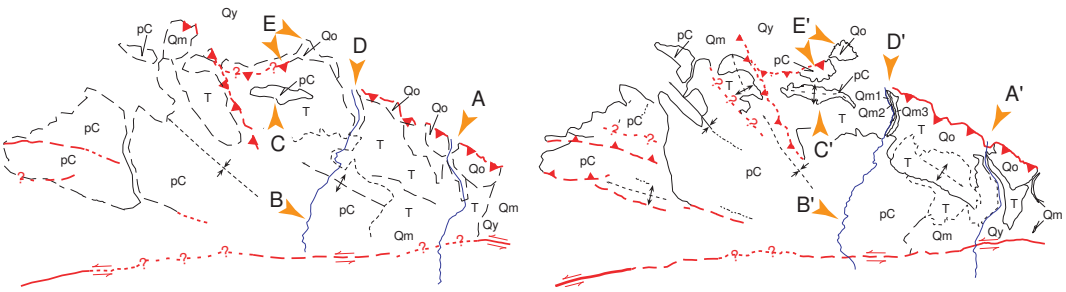


Figure 7 Results of efficiency test. Decorated red lines are various types of active faults. Black lines represent folds and contacts between two different geologic units. Red and black lines are solid where features are confidently located and dashed, dotted, or querried where position is increasingly less clear. Solid blue lines are drainages. Text labels (pC, T, Qo, Qm#, Qy) denote units of different apparent ages. Left: Map generated using StereoAnalyst. Right: Map generated using RIMS.

6 Conclusions and Future Work

In summary, while the tests described above show that the maps generated using both utilities capture many of the same major geologic features, it is clear that RIMS is both a more sensitive and a more efficient mapping utility, and thus greatly advances geologists' ability to remotely map patterns of active defomation in fine detail while also spanning continental collision zones that are thousands of kilometers wide and often inaccessible for field study. The advantages of RIMS over the previously used system are mostly due to RIMS' interactive visualization of large textured 3D terrain models, and its ability to map directly onto the 3D terrain in real-time.

Our future efforts will focus on moving the terrain maintenance out-of-core to allow for more scalability. We are also looking into on-demand textures to support a higher quantity of mappings with a more varied appearance (as seen in the results figures produced with ArcMap). In addition, mapping capabilities are to be extended providing geologist with more tools and help so as to more efficiently extract interesting features from the data sets.

Acknowledgments

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, through the National Partnership for Advanced Computational Infrastructure (NPACI), and a large Information Technology Research (ITR) grant. This work was partially supported by the W. M. Keck Center for Active Visualization in the Earth Sciences (Keck CAVES) and NASA grant EOS/03–0663–0306. We have benefited from conversations with Magali Billen, Louise Kellogg, and Nickolas Raterman. We thank the members of the Visualization and Graphics Research Group at the Center for Image Processing and Integrated Computing (IDAV) at the University of California, Davis.

References

1 P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. BDAM – batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22(3), 2003.

2 E. Cowgill, J Ramón Arrowsmith, A. Yin, X.-F. Wang, and Z. Chen. The Akato Tagh bend along the Altyn Tagh fault, NW Tibet 2. Active deformation and the importance of transpression and strain-hardening within the Altyn Tagh system. *Geological Society of America Bulletin*, 2004.

3 E. Cowgill, A. Yin, J Ramón Arrowsmith, Xiao-Feng Wang, and Shuanghong Zhang. The Akato Tagh bend along the Altyn Tagh fault, NW Tibet 1. Smoothing by vertical-axis rotation and the effect of topographic stresses on bend-flanking faults. *Geological Society of America Bulletin*, 2004.

4 Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In *Proceedings of the 8th conference on Visualization '97*, pages 81–88. IEEE Computer Society Press, 1997.

5 Oliver Kersting and Jürgen Döllner. Interactive 3D visualization of vector data in GIS. In *Proceedings of the tenth ACM international symposium on Advances in geographic information systems*, pages 107–112. ACM Press, 2002.

6 J. M. Lees. Geotouch: Software for three- and four-dimensional GIS in the Earth sciences. *Computers and Geosciences*, 26:751–761, 2000.

7 Peter Lindstrom, David Koller, Larry F. Hodges, William Ribarsky, Nickolas Faust, and Gregory Turner. Level-of-detail management for real-time rendering of phototextured terrain. Technical Report 6, 1995.

8 Peter Lindstrom and Valerio Pascucci. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239–254, 2002.

9 GRASS GIS. http://grass.baylor.edu.

10 ArcScene utility in 3D Analyst extension of ArcGIS. http://www.esri.com/software/arcgis/arcgisxtensions/3danalyst/.

11 3D SurfaceView utility in ENVI. http://www.rsinc.com/envi/.

12 VirtualGIS module in ERDAS IMAGINE. http://gis.leica-geosystems.com/Products/Imagine/.

13 FLY! http://www.pcigeomatics.com/product_ind/fly.html.

14 ArcGlobe, to be released in Spring 2004. http://www.esri.com/news/arcnews/summer03articles/introducing-arcglobe.html.

15 Stereo Analyst for ArcGIS. http://gis.leica-geosystems.com/Products/StereoAnalyst/.

16 OrthoEngine add-on for Geomatica. http://www.pcigeomatics.com/product_ind/add_on_oe.html.

17 SOCET SET. http://www.vitec.com/products/socetset/.

18 Zachary Wartell, Eunjung Kang, Tony Wasilewski, William Ribarsky, and Nickolas Faust. Rendering vector data over global, multi-resolution 3D terrain. In *Proceedings of the symposium on Data visualisation 2003*, pages 213–222. Eurographics Association, 2003.

A Survey of Visualization Methods for Special Relativity

Daniel Weiskopf[1]

1 **Visualization Research Center (VISUS), Universität Stuttgart**
weiskopf@visus.uni-stuttgart.de

Abstract

This paper provides a survey of approaches for special relativistic visualization. Visualization techniques are classified into three categories: Minkowski spacetime diagrams, depictions of spatial slices at a constant time, and virtual camera methods that simulate image generation in a relativistic scenario. The paper covers the historical outline from early hand-drawn visualizations to state-of-the-art computer-based visualization methods. This paper also provides a concise presentation of the mathematics of special relativity, making use of the geometric nature of spacetime and relating it to geometric concepts such vectors and linear transformations.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, J.2 Physical Sciences and Engineering

Keywords and phrases Special Relativity, Minkowski, Spacetime, Virtual Camera

Digital Object Identifier 10.4230/DFU.SciViz.2010.289

1 Introduction

Einstein's special theory of relativity has been attracting a lot of attention from the general public and physicists alike. In 2005, the 100$^{\text{th}}$ anniversary of the publication of special relativity [7] was the reason for numerous exhibitions, popular-science publications, or TV shows on modern physics in general and Einstein and his work in particular. In special relativistic physics, properties of space, time, and light are dramatically different from those of our familiar environment governed by classical physics. We do not experience relativistic effects and thus do not have an intuitive understanding for those effects because we, in our daily live, do not travel at velocities close to the speed of light.

Special relativity is usually described in terms of mathematical models such as spacetime and Lorentz transformations. Since special relativity has a strong geometric component, visualization can play a crucial role in making those geometric aspects visible without relying on symbolic notation. This paper gives an overview of different approaches for visualizing various aspects of special relativity to a range of different audiences. One application is the support of visual communication for a general public, for example, by means of illustrations in popular-science publications, exhibitions, or TV shows. Another audience are students because visualization can be used to improve the learning experience in high-school or university courses. For example, depictions help to motivate, interactive computer experiments allow for exploration and active participation, and visual explanations can enrich a symbolic description of mathematical ideas.

A third group of people are experts in physics and relativity. Although they do not need visualization to learn and understand the mathematics of special relativity, visualization may engage them in a different way of thinking. Edwin F. Taylor, a renowned teacher of special relativity [33], expressed his experience with special relativistic visualization as follows [32,

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 289–302

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

Section 6]: "[...] I have come to think about relativity quite differently [...] My current view of the subject is much more visual, more fluid, more process oriented, covering a wider range of phenomena [...] In short, both my professional life and my view of physics have been transformed [by relativistic visualization]."

These different types of audiences have a quite varying background knowledge. Therefore, different visualization approaches may be employed for different audiences and purposes. This paper distinguishes three classes of basic approaches (see Section 2): A direct visualization of spacetime by Minkowski diagrams, a visualization of a subset of spacetime for a fixed time, and the simulation of images as taken by a fast moving camera. Technical aspects and algorithms for these different approaches are discussed in Sections 5–7. All three approaches have in common that they may benefit from interactive computer implementations that facilitate trial-and-error explorations through the user.

This paper has several goals. First, it provides a survey of state-of-the-art computer-based methods for special relativistic visualization. The methods are roughly structured along the aforementioned classification of approaches. Because several alternative algorithms are available for the camera metaphor, these algorithms are further classified in subcategories (see Section 7). In addition, this paper provides a concise presentation of special relativity in Section 3, making use of the geometric nature of spacetime and relating it to geometric concepts such vectors and linear transformations. This paper also provides a historical outline of the development of special relativistic visualization (Section 4).

2 Types of Visualization Approaches

This paper adopts a geometric point of view on special relativity. Key elements are the concepts of spacetime and Lorentz transformations, which relate different coordinate systems in spacetime. Section 3 explains these concepts on a mathematical level. Spacetime is the combination of 3D space and a single temporal dimension, leading to a joint 4D description. Physical experiments, even if they are only virtual, are represented in form of 4D spacetime coordinates. For example, a point-like object that moves through space and time leaves a trace in spacetime—a so-called worldline. Similarly, light rays can be represented as lines through spacetime. Therefore, spacetime and traces therein are sufficient to describe the physical scenarios that are relevant for this paper.

Lorentz transformations represent changes between coordinate systems—transformations between different frames of reference. Due to the Lorentz transformation, observers in different frames of reference typically provide different coordinate descriptions for the very same physical object. In other words, both spatial and temporal positions are dependent on the reference frame—space and time are not addressed by absolute coordinates, but they are relative. The structure of spacetime and the Lorentz transformation can be derived from two postulates: the principle of relativity (i.e., physical laws are valid and unchanged in any inertial reference frame) and the invariance of the speed of light (i.e., the speed of light in vacuo has a finite and constant value, regardless of the reference frame). This derivation can be found in textbooks, such as [22].

The three visualization approaches discussed in this paper can be related to spacetime in the following ways. All approaches have in common a reduction of dimensionality of 4D spacetime.

(a) *Minkowki diagrams.* Minkowski diagrams are spacetime diagrams. They depict spacetime by graphically representing both temporal and spatial dimensions in a single image. The dimensionality of the spatial domain is reduced to either one or two (by taking a slice

through space), which leads to a total number of two or three dimensions for the spacetime diagram. In this way, graphical representations of the 2D or 3D diagram are feasible in an image. The advantage of Minkowski diagrams is their direct visualization of spacetime itself—Minkowski diagrams are the visual pendant to the mathematical geometry of special relativity. Figures 1 and 2 show typical examples of Minkowski diagrams.

(b) *Spatial slices.* Another way of reducing dimensionality is to construct a spatial slice of constant time. This slicing corresponds physically to a simultaneous measurement of positions in 3D space. Time and simultaneity depend on the frame of reference, i.e., a spatial slice is always defined with respect to a reference frame. Spatial slices are a natural metaphor because they model measurements in 3D space for a "frozen" time. Figure 3 provides an example of several spatial slices taken at different times.

(c) *Virtual camera model.* The virtual camera model simulates a physical experiment: what kind of image would a camera produce in a special relativistic setting? This approach simulates what we would see and, therefore, is the special relativistic analog of standard image synthesis by rendering non-relativistic scenes. Figure 4 illustrates the virtual camera view for high-speed travel toward the Brandenburg Gate.

In a non-relativistic setting, where the speed of light is assumed to be infinite, approaches (b) and (c) are identical. Special relativity, however, requires us to make a clear distinction between seeing and measuring. Measurements are made at sample points simultaneously with respect to the reference frame of the observer. In contrast, seeing is based on the photons that arrive simultaneously at the camera of the observer. These photons are usually not emitted simultaneously (with respect to the observer's reference frame) due to the finite speed of light. Following [41], approaches (a) and (b) can be regarded exocentric visualization, which present an outside view, whereas approach (c) can be considered an egocentric visualization, which is produced from the perspective of the user.

3 Elements of Special Relativity

This section provides a brief introduction to the mathematics of special relativity, discussing the concepts of spacetime, Lorentz transformation, four-vectors, and the Minkowski metric. More details on these concepts can be found in textbooks like [21, 22], or in a visualization-orientated presentation of special relativity [40].

Spacetime consists of three spatial dimensions and one temporal dimension. Analogously to a vector in Euclidean 3D space, a vector in spacetime can be described by four components

$$x^\mu = (t, x, y, z) = (x^0, x^1, x^2, x^3) \ , \quad \mu = 0, 1, 2, 3 \ .$$

The Greek indices $\{1, 2, 3\}$ refer to three spatial components and the index 0 refers to the temporal dimension. To simplify the notation in this paper, natural units are used, in which the speed of light $c = 1$. The geometry of spacetime is described by the Minkowski metric

$$\eta_{\mu\nu} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \ , \tag{1}$$

which is used to compute distances and inner products in spacetime. The inner product between two vectors, a^μ and b^μ, is

$$a^\mu \cdot b^\mu = \eta_{\mu\nu} a^\mu b^\nu \ .$$

The Einsteinian sum convention is applied: terms with duplicate indices are implicitly summed, e.g., the indices μ and ν are implicitly summed over all entries $0 \ldots 3$. The inner product is used to define the squared length of a spacetime vector—a so-called four-vector. For example, the position vector x^μ has squared length

$$x^\mu \cdot x^\mu = \eta_{\mu\nu} x^\mu x^\nu = t^2 - x^2 - y^2 - z^2 \ .$$

The Lorentz transformation is a linear and homogeneous transformation of spacetime vectors

$$x^{\mu\prime} = \Lambda^\mu{}_\nu x^\nu \ , \tag{2}$$

where $\Lambda^\mu{}_\nu$ is the matrix representation of the transformation. A Lorentz transformation is defined as a transformation that does not change the inner product or the squared length of four-vectors—it does not affect the geometry of spacetime as described by the Minkowski metric. The collection of Lorentz transformations forms the Lorentz group. A Lorentz transformation connects two inertial frames of reference and leaves the speed of light invariant. Lorentz transformations can be interpreted as a combination of spatial rotations of the two reference frames and a so-called Lorentz boost. The Lorentz boost is a velocity transformation between two reference frames that move at different relative speeds, given by (cf. [21, p.69])

$$\Lambda^\mu{}_\nu = \begin{pmatrix} \gamma & -\beta\,\gamma\,n_x & -\beta\,\gamma\,n_y & -\beta\,\gamma\,n_z \\ -\beta\,\gamma\,n_x & (\gamma-1)\,{n_x}^2+1 & (\gamma-1)\,n_x\,n_y & (\gamma-1)\,n_x\,n_z \\ -\beta\,\gamma\,n_y & (\gamma-1)\,n_x\,n_y & (\gamma-1)\,{n_y}^2+1 & (\gamma-1)\,n_y\,n_z \\ -\beta\,\gamma\,n_z & (\gamma-1)\,n_x\,n_z & (\gamma-1)\,n_y\,n_z & (\gamma-1)\,{n_z}^2+1 \end{pmatrix} \ . \tag{3}$$

Here, $\vec{n} = (n^1, n^2, n^3)$ is the normalized direction of motion, β is the velocity relative to the speed of light, and $\gamma = 1/\sqrt{1-\beta^2}$.

By including translations between reference frames, the Lorentz group is extended to the Poincaré group. The Lorentz group transforms four-vectors in spacetime, while the Poincaré group transforms points in spacetime. A spacetime point is usually called an event. For comparison, an analogous distinction has to be made between points and vectors in Euclidean 3D space—linear transformations are applied to vectors while affine transformations are applied to points. Events and four-vectors are tightly connected: the difference between two spacetime events is described by a four-vector that connects both events.

In general, the squared length of a four-vector is independent of the frame of reference because it is not changed by Lorentz transformations. Therefore, a vector can be characterized by its length. A vector is lightlike when it has vanishing length. Lightlike difference vectors are important for image synthesis because they connect light emission and absorbtion events. Accordingly, the propagation of a photon can be described by its lightlike four-wavevector, which combines circular frequency and 3D wavevector:

$$k^\mu = (\omega, \vec{k}) \ . \tag{4}$$

The 3D wavevector $\vec{k}$ points into the light direction and has length $k = 2\pi\lambda^{-1}$, where λ is the wavelength. The circular frequency ω is related to the frequency ν of the photon by $\omega = 2\pi\nu$. Wavelength and frequency are related by $\lambda = \nu^{-1}$.

By applying a Lorentz transformation to the four-wavevector, the aberration of light and the Doppler effect can be immediately computed. The relativistic aberration of light

describes the change of light direction caused by the Lorentz transformation. The aberration can be expressed as

$$\cos\theta' = \frac{\cos\theta - \beta}{1 - \beta\cos\theta} \ , \qquad \phi' = \phi \ , \tag{5}$$

where directions are given by spherical coordinates (θ, ϕ) and (θ', ϕ'). The Doppler effect accounts for the transformation of wavelength from one inertial frame of reference to another and is described by

$$\omega' = \frac{\omega}{D} \ , \tag{6}$$

with the Doppler factor

$$D = \frac{1}{\gamma\,(1 - \beta\cos\theta)} \ . \tag{7}$$

Expressed in terms of wavelength, the Doppler effect is

$$\lambda' = D\,\lambda \ . \tag{8}$$

Finally, the wavelength-dependent radiance L_λ is transformed according to

$$L'_\lambda(\lambda', \theta', \phi') = \frac{1}{D^5} L_\lambda(\lambda, \theta, \phi) \ . \tag{9}$$

A derivation of this relation is described by Weiskopf et al. [43] in the context of special relativistic visualization. The transformation of radiance leads to the so-called searchlight effect because light that is incident from the direction of motion is increased in radiance.

4 Historic Remarks and Recent Developments

This section discusses the timeline of developments in special relativistic visualization. The presentation is in approximately chronological order, from early work on special relativity to recent improvements on computer-based visualization. Although this overview of the literature covers papers from both physics and computer science, the focus is on a survey of special relativistic visualization based on modern computer graphics.

Einstein's original article on special relativity [7] was published in 1905. However, the contemporary understanding of special relativity as a description of spacetime associated with a certain metric goes back to Hermann Minkowski, as laid out in his 1908 lecture on "Raum und Zeit" ("space and time") [20]. The accompanying four-vector formalism may be attributed to Arnold Sommerfeld and his publications [29, 30] from 1910. These developments form the basis for a geometric point of view on special relativity that is widely adopted in most contemporary presentations and textbooks. Moreover, this geometric interpretation is vital for the extension to general relativity. More historical details on the early developments in special relativity can be found in an article by Walter [36].

A direct visualization of the spacetime geometry is typically based on a Minkowski diagram, which is a most popular type of visualization in textbooks. Minkowski introduced his diagrams in his lecture on space and time [20].

The second fundamental visualization approach—using spatial slices—was also used early on. For example, such a visualization appears in the movie "Die Grundlagen der Einsteinschen Relativitäts-Theorie", which is an early popular-science documentary film and probably the first animated visualization of special relativity. This film, with a length of over

two hours, had its premiere in 1922. Unfortunately, no copy of the original movie is known to have survived until today [37]. However, an abridged and modified American version that was released in 1923 as a 20-minute film with the title "The Einstein Theory of Relativity" is available, including an accompanying booklet [27].

The third visualization approach—the virtual camera model—took much longer before it was developed. Remarkably, the issue of image generation in the context in special relativity was ignored for a long time, or wrong interpretations were given. For example, Gamow equates the Lorentz contraction and the visual appearance of moving bodies in his book "Mr Tompkins in Wonderland" [8], neglecting the difference between seeing and simultaneously measuring. Apart from a previously disregarded article by Lampa [18] in 1924 about the invisibility of the Lorentz contraction, it was only in 1959 that the first correct solutions to the issue of image generation within special relativity were presented by Terrell [34] and Penrose [23]. Later, more detailed descriptions of the geometrical appearance of fast moving objects were discussed in the physics literature, e.g., by Weisskopf [45], Boas [3], Scott and Viner [26], Scott and van Driel [25], Hollenbach [11], Hickey [10], Suffern [31], Burke and Strode [4], Sheldon [28], Terrell [35], and Kraus [17].

In the 1980s, computers were gradually used for advanced special relativistic visualization. The earliest published example known to the author is a software by Taylor [32], which was used for teaching courses at MIT starting 1986. This software included both Minkowski diagrams and the virtual camera model, and rendering was based on line graphics. Besides this published work, there might have been other implementations of that era that may be overlooked today because their results were not disseminated publicly.

In 1989, Hsiung and Dunn [12] were the first to publish an advanced rendering technique for the virtual camera approach. Their rendering technique is based on an extension of standard 3D ray tracing. Within the following year, Hsiung and co-workers extended their work to include the visualization of the Doppler effect [14], relativistic time dilation [13], and the time-buffer method [15]. In 1991, Gekelman et al. [9] described a rendering method related to the time-buffer. Later developments included extended illumination models by Chang et al. [5] and Betts [2], which were subsequently improved by Weiskopf and co-workers [43, 44]. Other papers addressed the issue of acceleration in the context of the virtual camera approach [9, 44, 16, 24, 39]. As alternative rendering methods, texture-based rendering was introduced by Weiskopf [38] and image-based rendering by Weiskopf et al. [42]. Weiskopf [40] described these techniques in more detail and proposed a special relativistic version of the radiosity method. Li et al. [19] extended special relativistic ray tracing to include reflection and transmission phenomena. Finally, Weiskopf et al. [41] reported on their experiences with relativistic visualization for popular-science and educational presentations.

5 Minkowski Diagrams

A Minkowski diagram is the standard way of depicting the spacetime of special relativity. Figure 1 shows a typical example. Such a diagram is a direct visualization of the mathematical concepts described in Section 3: spacetime events are visualized as points (dots), four-vectors are shown as connecting lines, the wordlines of objects as lines, and reference frames are indicated by their respective coordinate axes. The light rays absorbed by the observer form a light cone that is also illustrated by a line (or a collection of lines).

The popularity of Minkowski diagrams in textbooks and other scientific presentations of special relativity is rooted in the fact that those diagrams provide a geometric visualization of spacetime and its important constituents. In other words, a Minkowski diagram goes

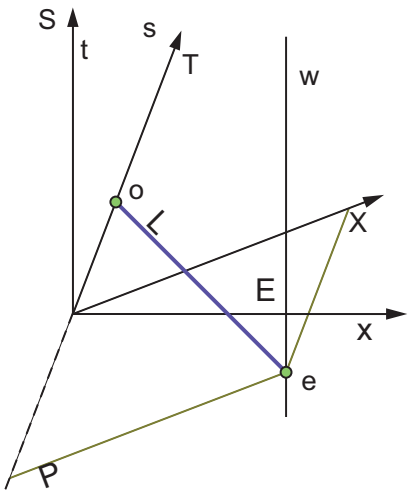


Figure 1 1+1D Minkowski diagram showing light emission from a static object and light absorption by a moving camera.

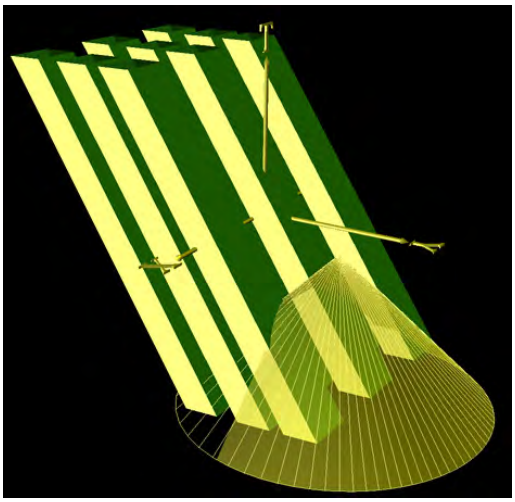


Figure 2 2+1D Minkowski diagram with several worldtubes and a backward light cone originating from a virtual camera.

hand-in-hand with the mathematical description from Section 3. The superposition of inertial reference frames using a Minkowski diagram allows us to visualize the Lorentz transformation of an event. Figure 1 illustrates the Lorentz transformation of the event E to frame S' by projection lines (light solid lines).

Unfortunately, there are a few issues connected with Minkowski diagrams. One problem is the reduction of dimensionality: typically, only the temporal dimension and one spatial dimension are shown in a diagram. Sometimes a second spatial dimension is included. However, Minkowski diagrams never show all four dimensions of spacetime but only a subspace thereof. Therefore, these diagrams cannot provide a faithful representation of the complete physical scenario. Another issue is the interpretation of angles and inner products in a Minkowski diagram. Since the Minkowski metric (Eq. (1)) is different from the Euclidean metric, inner products in a Minkowski diagram cannot be inferred intuitively from our experience with Euclidean space, in which the diagram is rendered. For example, the inner product of the t and x axes of any reference frame vanishes even though these axes do not seem to be perpendicular in the rendered diagrams (see the coordinate system S' in Figure 1). However, this interpretation issue is more of a problem in popular-science presentations than in scientific publications due to the different background knowledge of the readers. Therefore, Minkowski diagrams primarily address physicists and researchers as an audience.

Minkowski diagrams are usually not based on complex scene descriptions or other external data because they rather target the visualization of simple spacetime relationships such as a lightlike connection between emission and absorption events. Therefore, Minkowski diagrams are typically generated without specific computer support. For example, they may be hand-drawn illustrations or produced with a generic vector graphics program. The educational software by Taylor [32] is one of the few examples where spacetime diagrams are computer-generated. Another example is the system by Diepstraten et al. [6] that automatically generates Minkowski diagrams from a 3D scene description. This system, as another advanced feature, allows for 2+1D diagrams with two spatial axes. 2+1D diagrams

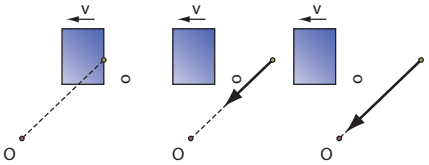


Figure 3 Spatial slices at three different times, taken from the motion of a cube-shaped object.

Figure 4 Virtual camera view for traveling at $\beta = 0.99$ toward the Brandenburg Gate.

have a couple of advantages compared to traditional 1+1D diagrams: extended objects can be shown instead of point particles; the visibility properties between objects can be visualized; objects may move in various directions of motion; angles become apparent in two spatial dimensions; finally, the relativistic aberration of light can only be seen in more than one spatial dimension. Figure 2 shows a 2+1D Minkowski diagram with several worldtubes (i.e., the 2D analog of wordlines) and a backward light cone originating from the observer. The intersections of the backward light cone with the worldtubes indicate the emission of light that is registered by the virtual camera.

6 Spatial Slices

Spatial slices can be constructed for a fixed time, which corresponds to a simultaneous measurement of positions in 3D space. Due to the relativity of time, a spatial slice is always associated with a reference frame—this visualization approach intrinsically depends on coordinate systems. Therefore, one may have to face apparent paradoxes if this frame dependency is not carefully observed. On the other hand, spatial slices are quite intuitive in the sense that they show simultaneous measurements.

Spatial slicing is typically applied to present an "outside" perspective by an omniscient viewer in order to explain relativistic phenomena. Because such phenomena are usually related to some aspects of light propagation, typical visualizations include not only depictions of scene objects but also representations of light rays. Figure 3 shows a 2D example of several spatial slices taken at different times. Here, both a moving box-shaped scene object and a light ray are shown.

Spatial slicing can be implemented by using standard 2D or 3D computer graphics that is based on the assumption of infinite speed of light. Instantaneous light transport is one way of realizing a simultaneous measurement. The only extension, as compared to standard computer graphics, is the need for a Lorentz transformation of scene objects when the reference frame is changed. While the Lorentz transformation could be computed according to the matrix–vector multiplication from Eq. (2), there is a simpler solution for uniformly moving objects: these objects are reduced in length with a factor $1/\gamma$, which is called Lorentz contraction. Therefore, the relativistic effects can be included by a simple scaling along the direction of motion, which can even be done "manually" in any 2D or 3D of-the-shelf graphics tool (e.g. Adobe Illustrator or Maya). The main application of spatial slices is for physics education and popular-science presentations. Examples can be found on the web page [W3].

7 Virtual Camera Model

The virtual camera model is based on a physical experiment: what kind of image would a camera produce in a special relativistic setting? This approach is the special relativistic analog of standard image synthesis. It is also related to spatial slicing from the previous section. The main difference is that the finite speed of light is taken into account for the virtual camera model. The virtual camera model is typically applied to a fast moving camera: how would the world be perceived in a "relativistic flight simulator"? Because this scenario is conceptually simple, it is appropriate for popular-science and educational illustrations [41].

7.1 Special Relativistic Polygon Rendering

The idea of special relativistic polygon rendering is to transform the 3D geometry of a static scene to the apparent shape of the scene objects as seen by a relativistic observer. This approach works in object space, transforming the original 3D geometry to another 3D geometry. The scene objects emit light (either directly or indirectly through light reflection) and, thus, can be related to light emission events. Conversely, the camera is related to a light absorption event. Relativistic polygon rendering relies on the relationship between light emission and absorption events, and the Poincaré transformation of these events in order to allow for moving cameras or objects.

The Minkowski diagram in Figure 1 illustrates these transformations for a single point-like scene object. The line $\{(t, x_e)|t\}$ denotes the worldline of the object in its rest frame S. The intersection of the worldline of the scene object with the backward light cone originating from the camera event O determines the emission event E, i.e., the connection between E and O is lightlike. The time coordinate of E in the frame S is determined by

$$\left(x_o^0 - x_e^0\right) = \sqrt{(x_e^1 - x_o^1)^2 + (x_e^2 - x_o^2)^2 + (x_e^3 - x_o^3)^2} \quad , \tag{10}$$

where x_e^μ denotes the coordinates of E and x_o^μ the coordinates of O. The Poincaré transformation allows us to transfer the coordinates of the emission event from the scene frame to the camera frame. The spatial coordinates of that event can be used to render the object as seen by the fast moving camera. To render a more complex scene, the transformation of events is applied to all vertices of the tessellated scene. Please note that, due to the nonlinear nature of Eq. (10), the combined transformation of vertex locations is nonlinear.

Relativistic polygon rendering fits to the GPU rendering pipeline: vertex coordinates are modified at the first stage of the rendering pipeline, whereas the other stages of the pipeline remain unaffected. The transformation of vertex coordinates can be done either by CPU processing or in a vertex program on the GPU. The object-space approach to special relativistic rendering is used in several papers with sometimes slight variations of the same computational theme [5, 9, 15, 24, 39].

Object-space relativistic rendering has several advantages: it is easy to implement, it is fast due to its direct support by graphics hardware, and it allows for scene objects that move at different speeds. The main disadvantage is caused by the nonlinear transformations of vertex coordinates. The linear connections between vertices through straight edges may lead to artifacts that are most prominent for large, nearby triangles. This problem can be overcome by a view-dependent re-tessellation to refine large triangles.

The transformation of vertex positions only accounts for the apparent geometry as seen by the moving camera. The relativistic effects on illumination can be incorporated by modifying the color and intensity according to the Doppler and searchlight effects (see Eqs. (8) and (9)).

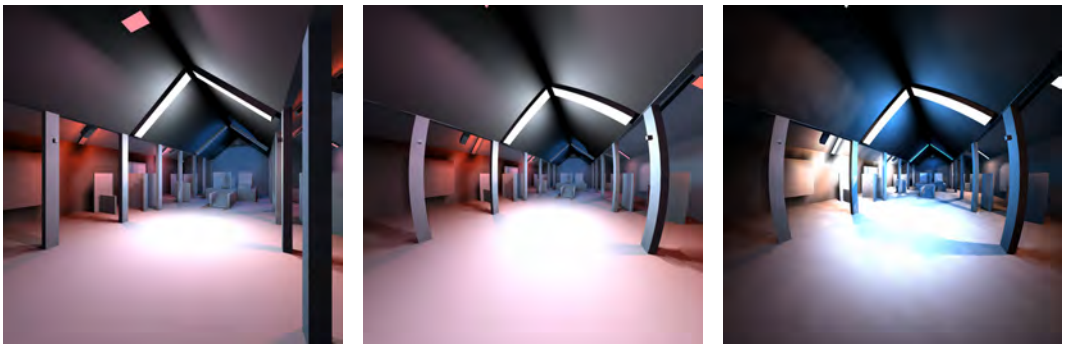


Figure 5 Polygon rendering of relativistic radiosity: non-relativistic view (left), aberration effects for $\beta = 0.6$ (center), and all relativistic effects included (right).

Typically, relativistic polygon rendering considers only direct illumination. However, global illumination is feasible as well. For example, special relativistic radiosity rendering allows for global illumination with diffuse reflections [40]. Like non-relativistic radiosity, the first rendering stage is view-independent and computes the radiosity solution for a static scene. The second stage uses relativistic polygon rendering to construct the image for a relativistic observer. Figure 5 shows an example of relativistic radiosity.

7.2 Image-Based Special Relativistic Rendering

Image-based special relativistic rendering [42] computes images without the need for a 3D scene representation. It uses the plenoptic function, which describes the radiance field depending on light direction, spacetime position, and wavelength [1]. The plenoptic function is first acquired for a static camera in the rest frame of the scene, and then transformed to the frame of a moving camera. Afterwards, non-relativistic rendering methods are applied to generate the final image. The Lorentz transformation of the plenoptic function is governed by the following relativistic effects: the aberration of light changes the direction according to Eq. (5), the Doppler effect modifies the wavelength according to Eq. (8), and the searchlight effect alters the radiance according to Eq. (9).

Often, the plenoptic function is considered only for a single camera location, leading to a panorama image taken from that position. Here, the relativistic aberration just leads to a nonlinear warping of the panorama. The Doppler and searchlight effects additionally change the color and brightness of the panorama. However, all these operations can be realized by simple image manipulations. Image-based rendering can be applied to real-world images [42] or to textures generated on-the-fly by non-relativistic rendering of 3D scenes [38, 41]. Figure 4 shows an example of texture-based rendering where only the geometric effects due to the aberration of light are visualized.

Image-based rendering has the following advantages. First, it is well supported by the graphics pipeline of GPUs—non-relativistic rendering may be used to generate the panorama and only one additional rendering step is required for the image transformation. Second, no view-dependent re-tessellation of the scene is needed because the nonlinear transformations work on a per-pixel basis for the panorama image. Image-based rendering is per-pixel accurate, both for the geometric and illumination effects. A disadvantage of image-based rendering is that the panorama needs to be acquired at high resolution to ensure a sufficient sampling rate after the nonlinear Lorentz transformation. Another shortcoming is the lack of support for objects that move at differing speeds.

7.3 Special Relativistic Ray Tracing

A third class of rendering approaches is based on ray tracing. For scenarios with a single static scene and a moving observer, non-relativistic ray tracing needs only a slight modification to incorporate relativistic effects: the primary ray directions are transformed from the moving camera frame to the frame of the static scene [12]; afterwards, non-relativistic ray tracing is performed. The ray direction and wavelength can be transformed by applying the Lorentz transformation to the corresponding wavevector (see Eq. (4)). The radiance is transformed according to Eq. (9).

Full 4D special relativistic ray tracing is needed to include more advanced effects such as accelerating scene objects and shadowing, reflection, or transmission of light [19, 41]. 4D ray tracing represents light rays, scene objects, and the intersection between rays and objects in 4D spacetime. In this way, animated and accelerated objects can be modeled. To compute the local illumination at an intersection event, the light information needs to be transferred into the rest frame of the object that is hit by the ray. This transformation is accomplished by Eqs. (5), (8), and (9).

One advantage of ray tracing is high image quality and the support for reflection and transmission. Full 4D ray tracing additionally extends the range of effects to include accelerated objects and advanced interreflection between objects that move relative to each other. Therefore, 4D ray tracing is the visualization method with the widest range of supported relativistic effects. The main disadvantage of ray tracing are high computational costs, which typically make interactivity impossible.

7.4 Acceleration

The accelerated motion of a point-like object through spacetime can be computed by solving a corresponding equation of motion (an ordinary differential equation). In this way, accelerating cameras can be modeled [24, 39, 44]. The image generation for an accelerating camera is identical to the relativistic rendering for a co-moving, non-accelerating camera (i.e., a camera that moves with the same instantaneous velocity). Therefore, any of the aforementioned rendering techniques can be employed. The acceleration of extended scene objects is more complex and subject to issues of correct physical modeling to ensure a consistent and physically plausible motion [16, 41]. Here, the rendering is usually accomplished by 4D ray tracing.

8 Open Issues

Most of the aforementioned virtual camera techniques facilitate rendering at interactive rates so that the efficiency of relativistic rendering can be considered a solved problem. A largely unsolved issue, however, is the photorealistic computation of wavelength-dependent radiance in a scene, which is the prerequisite for realistic rendering of the Doppler and searchlight effects. The main challenge is the acquisition or modeling of wavelength-dependent properties of scene objects and light sources (even beyond the visible spectrum because of the Doppler shift).

Although there exist some interaction metaphors for the virtual camera approach [39, 41], the development of appropriate user interfaces is a goal of on-going research. Similarly, the computer support for generating Minkowski diagrams and spatial slicing could be greatly improved by specific interaction models and design interfaces.

Web Links

A wealth of information on special relativistic visualization can be found on the web. Examples are: a list of special relativistic flight simulators [W2], a timeline for the development of computer-based relativistic visualization [W1], and didactics material for teaching relativity [W3].

Web Links

W1 D.V. Black. Visualization of non-intuitive physical phenomena. `http://www.hypervisualization.com` [accessed Feb 20, 2006].

W2 A. Hamilton. Guide to special relativistic flight simulators. `http://casa.colorado.edu/~ajsh/sr/srfs.html` [accessed Feb 20, 2006].

W3 U. Kraus, C. Zahn. Space time travel. `http://www.spacetimetravel.org` [accessed Feb 20, 2006].

References

1 E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In Michael Landy and J. Anthony Movshon, editors, *Computational Models of Visual Processing*, pages 3–20, Cambridge, 1991. MIT Press.

2 Christopher Betts. Fast rendering of relativistic objects. *The Journal of Visualization and Computer Animation*, 9(1):17–31, 1998.

3 Mary L. Boas. Apparent shape of large objects at relativistic speeds. *American Journal of Physics*, 29(5):283–286, May 1961.

4 John Robert Burke and Frank J. Strode. Classroom exercises with the Terrell effect. *American Journal of Physics*, 59(10):912–915, 1991.

5 Meng-Chou Chang, Feipei Lai, and Wei-Chao Chen. Image shading taking into account relativistic effects. *ACM Transactions on Graphics*, 15(4):265–300, October 1996.

6 J. Diepstraten, D. Weiskopf, and T. Ertl. Automatic generation and non-photorealistic rendering of 2+1D Minkowski diagrams. In *Proc. WSCG 2002*, pages 139–146, 2002.

7 Albert Einstein. Zur Elektrodynamik bewegter Körper. *Annalen der Physik*, 17:891–921, 1905.

8 George Gamow. *Mr Tompkins in Wonderland*. University Press, Cambridge, 1939.

9 Walter Gekelman, James Maggs, and Lingyu Xu. Real-time relativity. *Computers in Physics*, 5(4):372–385, July/August 1991.

10 F. R. Hickey. Two-dimensional appearance of a relativistic cube. *American Journal of Physics*, 47(8):711–714, 1979.

11 D. Hollenbach. Appearance of a rapidly moving sphere: A problem for undergraduates. *American Journal of Physics*, 44(1):91–93, 1975.

12 Ping-Kang Hsiung and Robert H. P. Dunn. Visualizing relativistic effects in spacetime. In *Proc. Supercomputing '89*, pages 597–606, 1989.

13 Ping-Kang Hsiung, Robert H. Thibadeau, Christopher B. Cox, and Robert H. P. Dunn. Time dilation visualization in relativity. In *Proc. Supercomputing '90*, pages 835–844, 1990.

14 Ping-Kang Hsiung, Robert H. Thibadeau, Christopher B. Cox, Robert H. P. Dunn, Michael Wu, and Paul Andrew Olbrich. Wide-band relativistic Doppler effect visualization. In *Proc. IEEE Visualization*, pages 83–92, 1990.

15 Ping-Kang Hsiung, Robert H. Thibadeau, and Michael Wu. T-buffer: Fast visualization of relativistic effects in spacetime. *Computer Graphics*, 24(2):83–88, 1990.

16 U. Kraus, H. Ruder, D. Weiskopf, and C. Zahn. Was Einstein noch nicht sehen konnte. Schnelle Computer visualisieren relativistische Effekte. *Physik Journal*, 1(7/8):77–83, July 2002.

17 Ute Kraus. Brightness and color of rapidly moving objects: The visual appearance of a large sphere revisited. *American Journal of Physics*, 68(1):56–60, 2000.

18 Anton Lampa. Wie erscheint nach der Relativitätstheorie ein bewegter Stab einem ruhenden Beobachter? *Zeitschrift für Physik*, 27:138–148, 1924.

19 Jiang Li, Heung-Yeung Shum, and Qunsheng Peng. An improved spacetime ray tracing system for the visualization of relativistic effects. In *Eurographics 2001 Short Presentations*, 2001.

20 Hermann Minkowski. Raum und Zeit. *Physikalische Zeitschrift*, 10:104–111, 1909.

21 Charles W. Misner, Kip S. Thorne, and John Archibald Wheeler. *Gravitation*. Freeman, New York, 1973.

22 C. Møller. *The Theory of Relativity*. Clarendon Press, Oxford, second edition, 1972.

23 Roger Penrose. The apparent shape of a relativistically moving sphere. *Proceedings of the Cambridge Philosophical Society*, 55:137–139, 1959.

24 René T. Rau, Daniel Weiskopf, and Hanns Ruder. Special relativity in virtual reality. In Hans-Christian Hege and Konrad Polthier, editors, *Mathematical Visualization*, pages 269–279. Springer Verlag, Heidelberg, 1998.

25 G. D. Scott and H. J. van Driel. Geometrical appearances at relativistic speeds. *American Journal of Physics*, 38(8):971–977, August 1970.

26 G. D. Scott and R. R. Viner. The geomentrical appearence of large objects moving at relativistic speeds. *American Journal of Physics*, 33(7):534–536, July 1965.

27 Garrett P. Serviss. *The Einstein Theory of Relativity*. E. M. Fadman, New York, 1923.

28 Eric Sheldon. The twists and turns of the Terrell effect. *American Journal of Physics*, 56(3):199–200, 1988.

29 Arnold Sommerfeld. Zur Relativitätstheorie I: Vierdimensionale Vektoralgebra. *Annalen der Physik*, 32:749–776, 1910.

30 Arnold Sommerfeld. Zur Relativitätstheorie II: Vierdimensionale Vektoranalysis. *Annalen der Physik*, 33:649–689, 1910.

31 K. G. Suffern. The apparent shape of a rapidly moving sphere. *American Journal of Physics*, 56(8):729–733, 1988.

32 Edwin F. Taylor. Space-time software: Computer graphics utilities in special relativity. *American Journal of Physics*, 57:508–514, 1989.

33 Edwin F. Taylor and John Archibald Wheeler. *Spacetime Physics*. Freeman, New York, second edition, 1992.

34 James Terrell. Invisibility of the Lorentz contraction. *Physical Review*, 116(4):1041–1045, November 1959.

35 James Terrell. The Terrell effect. *American Journal of Physics*, 57(1):9–10, 1989.

36 Scott Walter. Minkowski, mathematicians, and the mathematical theory of relativity. In H. Goenner, J. Renn, J. Ritter, and T. Sauer, editors, *The Expanding Worlds of General Relativity*, pages 45–86. Birkhäuser, Boston, Basel, 1999.

37 Milena Wazeck and Thomas Bührke. Der verschollene Film. *Bild der Wissenschaft,*, pages 47–49, February 2005.

38 Daniel Weiskopf. Fast visualization of special relativistic effects on geometry and illumination. In *Proc. EG/IEEE TCVG Symposium on Visualization*, pages 219–228, 2000.

39 Daniel Weiskopf. An immersive virtual environment for special relativity. In *Proc. WSCG 2000*, pages 337–344, 2000.

40 Daniel Weiskopf. *Visualization of Four-Dimensional Spacetimes*. PhD thesis, Universität Tübingen, 2001. `http://w210.ub.uni-tuebingen.de/dbt/volltexte/2001/240`.

41 Daniel Weiskopf, Marc Borchers, Thomas Ertl, Martin Falk, Oliver Fechtig, Regine Frank, Frank Grave, Andreas King, Ute Kraus, Thomas Müller, Hans-Peter Nollert, Isabel Rica Mendez, Hanns Ruder, Tobias Schafhitzel, Sonja Schär, Corvin Zahn, and Michael Zatloukal. Explanatory and illustrative visualization of special and general relativity. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):522–534, 2006.

42 Daniel Weiskopf, Daniel Kobras, and Hanns Ruder. Real-world relativity: Image-based special relativistic visualization. In *Proc. IEEE Visualization*, pages 303–310, 2000.

43 Daniel Weiskopf, Ute Kraus, and Hanns Ruder. Searchlight and Doppler effects in the visualization of special relativity: A corrected derivation of the transformation of radiance. *ACM Transactions on Graphics*, 18(3):278–292, July 1999.

44 Daniel Weiskopf, Ute Kraus, and Hanns Ruder. Illumination and acceleration in the visualization of special relativity: A comment on fast rendering of relativistic objects. *The Journal of Visualization and Computer Animation*, 11(4):185–195, 2000.

45 Victor F. Weisskopf. The visual appearance of rapidly moving objects. *Physics Today*, 13(9):24–27, 1960.

Audio-visual Virtual Reality System for Room Acoustics

Eduard Deines[1], Martin Hering-Bertram[2], Jan Mohring[2], Jevgenijs Jegorovs[2], and Hans Hagen[3]

1 **University of California, Davis, USA**
edeines@ucdavis.edu
2 **Fraunhofer ITWM, Kaiserslautern, Germany**
{martin.hering-bertram,jan.mohring,jegorovs}@itwm.fraunhofer.de
3 **Department of Computer Science, University of Kaiserslautern**
hagen@informatik.uni-kl.de

Abstract

We present an audio-visual Virtual Reality display system for simulated sound fields. In addition to the room acoustic simulation by means of phonon tracing and finite element method this system includes the stereoscopic visualization of simulation results using a 3D back projection system as well as auralization by use of a professional sound equipment. For auralization purposes we develop a sound field synthesis approach for accurate control of the loudspeaker system.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, J.2 Physical Sciences and Engineering

Keywords and phrases Special Relativity, Minkowski, Spacetime, Virtual Camera

Digital Object Identifier 10.4230/DFU.SciViz.2010.303

1 Introduction

For architectural planning of class rooms, theaters or concert halls the auditive impression of resulting rooms is very important. Auralization provides such a feasibility. It can be considered as the auditive variant of visualization, providing insight into acoustic properties of a room and its reasons. Virtual Reality environments enable an immersive representation of computer-generated scenes. Integrating acoustic simulation, visualization, and auralization into the design process aims at interactive design and immediate exploration of the virtual model.

In this work we present an audio-visual Virtual Reality system which integrates computer-aided simulation, visualization and auralization of acoustics in a room model. For auralization purposes we combine a FEM based approach and the phonon tracing algorithm [5] in order to obtain a realistic impression of the sound perceived at given listener positions. The wave field synthesis approach enables a correct auditive rendering of the convolved signals on a professional sound equipment. With our system, walkthrough of the listener positions is possible, such that a visual and auditive impression of the scene can be provided.

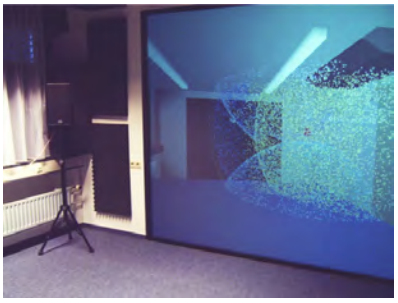


Figure 1 Room acoustic visualization on the VR system.

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 303–320
Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

From our visualization, the effect of different materials on the spectral energy distribution can be observed. The first few reflections already show whether certain frequency bands are rapidly absorbed. The absorbing materials can be identified and replaced in the virtual model, improving the overall acoustic quality of the simulated room. A wave front is the pressure wave corresponding to a unit pulse. In order to visualize the wave fronts spread out from the sound source we use both particles (spheres) and surface elements, color coded using the energy spectra of the corresponding virtual sound particles. After a sufficient time period, a great number of reflections has occurred, such that individual wave fronts cannot be identified, anymore. The darker the color of spheres gets, the more energy has been absorbed by the different materials. Applications of our work include (but are not limited to) the acoustic improvement during architectural design, and equipment phases of class- and congress rooms. sec Our paper is organized as follows. In Section 2 we review related work. Section 3 describes the acoustic virtual reality system. Section 4 outlines our simulation algorithm, implying the phonon tracing approach and the finite element method. Thereafter, we present our synthesis algorithm in Section 5, follows by visualization of the simulation results (Section 6). In Section 7 we show an application of our sound visualization approach, before we conclude our work.

2 Related Work

In room acoustics there are two main approaches simulating the propagation of sound. The first approach is based on the wave equation which is solved numerically, for example by use of the finite element method (FEM). The simulation results are very accurate, but the complexity increases drastically with the highest frequency considered, since a volume grid with $O(n^3)$ cells needs to be constructed where n is proportional to the highest frequency. The time complexity for solving this is typically $O(n^3\ log\ n^3)$. Hence, the wave model is suitable for low frequencies only.

The second approach, known as geometric acoustics, describes the sound propagation by sound particles moving along a directed ray. There exists a variety of such methods for simulating room acoustics. They are mostly based on optical fundamentals, and make use of approaches developed there. Two classical methods for acoustic simulation are the image-source method [1, 6] and the ray tracing method [25, 26]. Due to the shortcomings of the two classical approaches, continuative approaches have been developed in recent years. Mostly, they employ parts of the classical schemes or a combination of them. One approach that makes use of advantages of image-source and ray tracing is introduced in [46]. Here, the visibility check of the image-source algorithm is performed via ray tracing. Beam-tracing methods [12, 13, 32] overcome the aliasing problem of classical ray tracing by recursively tracing pyramidal beams, implying the need for highly complex geometric operations, still ignoring diffraction effects at low frequencies. An approach for calculation of edge diffraction in room acoustics is presented in [45, 30]. To overcome the dependency of the simulation on the receiver position the radiosity method was extended to be used in room acoustics [40, 24]. Due to the computation complexity these methods do not seem practical for large environments. Newer approaches cope with complexity by exploiting GPU hardware accelerating the simulation calculations [17]. Approaches utilizing the photon mapping [18] also exist [20, 5].

The aim of audio simulation is to estimate acoustic properties, such as reverberation time and the reproduction of acoustic benchmarks. Auralization is the process of producing audible impression of a room. It can be realized in two different ways. The first one is the ”direct room impulse response rendering” approach, where the room impulse response (RIR) is measured or modelled and afterwards convolved with the anechoic signal. In the DIVA auralization system [41, 29] a different approach is proposed, the "parametric room impulse response rendering". Here, the RIR is not calculated before the auralization process. Instead, a set of either perception-based or physic-based parameters for auralization is defined. In [34] an audio-rendering system for use in immersive virtual environments

is introduced. It is optimized for efficient rendering of moving sound sources. A survey of existing auralization systems is given in [41].

There are two major sound rendering approaches: wave field synthesis [4] and binaural synthesis [3]. The latter approach uses headphones to reproduce the sound field simulated at the ears of the virtual listener. The advantages are cheap equipment and low number of channels to be computed. However, the listener is bothered by the headphones and synthesis requires deep knowledge of the acoustics of the human head and psycho-acoustics. Therefore, we apply wave field synthesis. The underlying theory is based on Huygens' principle, or more mathematically, on the fact that solutions of the Helmholtz equation correspond to distributions of sound sources on the surface of the domain (Kirchhoff-Helmholtz integral, [14]). This continuous distribution is approximated by a finite number of loudspeakers. The higher the number of speakers and the lower the frequency the larger is the sweet spot where the approximation is accurate. Usually, only the sound field of a 2D listening plane is synthesized as this requires already quite large number of speakers. For instance, Fraunhofer IDMT furnished a cinema with 192 speakers to achieve good reproduction at all seats [22]. For our purpose it is enough to have a good reproduction close to the proband's head the position of which is, except for small movements, fixed. The present system comprises currently 4 full-band speakers and a bass. In a future system additional mid and high-band speakers will be added.

The existing acoustic visualization techniques can be classified into two groups. The methods of the first group consider the propagation of the sound waves or rays inside the room independent of the listener position. Whereas the algorithms in the second group base on the measured or simulated room impulse response for individual listener. Yokota et. al. [47] visualized sound propagation of 2-D sound fields using a difference time domain method. Petrausch and Rabenstein [37] introduce a program for simulation and OpenGL based visualization of 2-D sound wave propagation in real time. Tokita and Yamasaki [44] depicted particle displacements resulted from a wave-based simulation on a rigid 3-D grid in a cube shaped room. Lokki [29] presented a visualization using the image-source method. Lokki and Nenonen [28] utilized cave equipment for immersive visualization of trace paths and particle paths propagating inside closed rooms. In [39] Pulkki and Lokki present an approach visualizing edge diffraction. Funkhouser et. al. [12] used visualization of source points, receiver points, pyramidal beams, reverberation path etc. in order to understand and evaluate their acoustic modeling method. Lauterbach et. al. [27] showed the sound propagation resulting from frustum tracing. Sound visualization approaches utilizing the phonon tracing algorithm as well as comparative visualization of the phonon tracing and an FEM based solver are presented in [10] and [9] respectively. Khoury et. al. [21] represented the sound pressure levels inside the room by means of color maps. Additionally, the precedence effect (or "law of the first wave front") was analyzed by using isosurfaces. Stettner et. al. [43] visualized acoustic metrics such as clarity and definition as well as spatial impression by use of specific icons. Monks et al. [33] introduced an interactive optimization system for acoustic design. The results are presented by means of icons depicting early decay time (EDT), interaural cross-correlation coefficient (IACC), and bass ratio (BS). Furthermore, the sound strength was displayed at room surfaces using color to indicate the sound-level data at different time moments. Several approaches for visualization of measured sound intensity also exist [36, 11, 31].

For room acoustic modeling we combine in our system a FEM based method (for low frequency) and the phonon tracing approach [5] (for middle and high frequency). By this means we can consider the diffraction and interference, which can not be neglected for low frequencies, and manage the complexity problem of FEM by middle and high frequencies. For sound rendering purposes we introduce a wave field synthesis approach. In addition to the auralization we provide a visualization of sound wave propagation inside the virtual room.

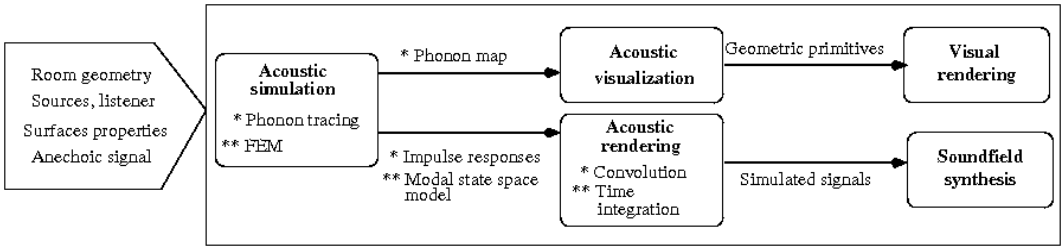


Figure 2 Modules of our audio-visual Virtual Reality system.

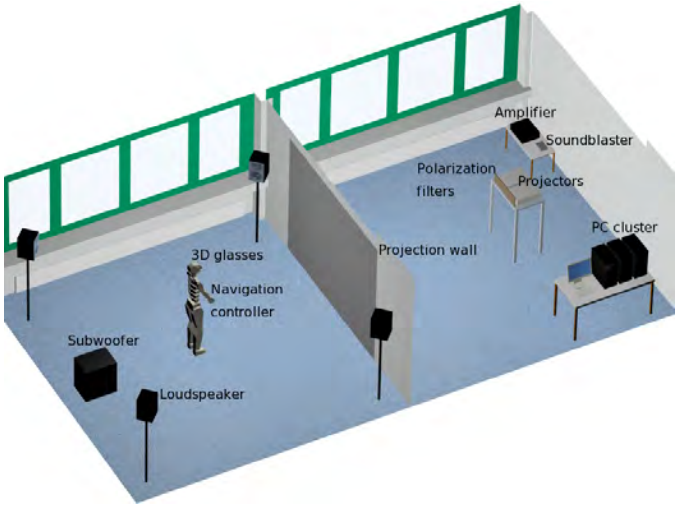


Figure 3 Visual and audio Virtual Reality display system.

3 Acoustic Virtual Reality System

In this section we present our visual and auditive virtual reality display system. A schematic overview is given in Fig. 2. As input the system requires the geometry model of the room, the absorption or reflection properties of the room surfaces, sound sources characteristics, and listener positions. The module ”Acoustic simulation” computes the modal state space model (low frequencies) by use of the Finite Element Method (FEM) described in Section 4.2, and the room impulse responses (RIR) and the phonon map (middle and high frequencies) by means of the phonon tracing algorithm introduced in Section 4.1. On the basis of this information the anechoic source signal is modified for sound synthesis in the ”Acoustic rendering” module. Thereafter, the soundfield synthesis is performed utilizing the acoustic hardware. The ”Acoustic Visualization” module provides the visualization of wave propagation from the sound source. The graphical rendering ("Visual rendering" module) is implemented utilizing a stereoscopic back projection system. By use of our VR system, a walkthrough of the listener positions is possible, such that a visual and auditive impression of the scene can be provided.

The hardware of the Virtual Reality system includes the 3D back projection system and the acoustic system (see Fig. 3). The 3D back projection system is composed of two high resolution digital D-ILA projectors for displaying mono and passive stereo signals, two circular mechanical shiftable polarization filters, and one projection wall, suitable for polarization filters. By means of

Table 1 Technical details of the VR System.

3D Powerwall	
2 projectors	1400x1050, 1500 ANSI Lumen
1 screen	plastic film suitable for polarization filters (2.88m x 2.30m)
2 filters	circular polarization filters
glasses	plastic glasses with circular polarization inserts
4.1 Acoustic system	
1 sound card	7.1 USB sound blaster
4 loudspeakers	two-way universal loudspeakers
1 subwoofer	active subwoofer with integrated amplifier
1 amplifier	four-channel amplifier with integrated programmable FIR Filters
PC cluster	
1 master	Dual Xeon 3.0 GHz, 2GB RAM, PCI-Express, NVidia Quadro FX1300
2 render nodes	Single Xeon 3.0 GHz, 2GB RAM, PCI-Express, NVidia Quadro FX5400

this system a stereoscopic rendering of virtual scenes is possible, observed by wearing polarizing eyeglasses. The acoustic system contains a 7.1 soundblaster and a professional surround sound equipment. This surround sound system on its part includes four two-way loudspeakers and one subwoofer facilitating the localization of the virtual sound source. The system is driven by the PC-cluster composed of three connected DELL computers. Two of them are the render nodes, and one acts as master controlling the navigation and audio-visual output. Technical specifications of the hardware of the Virtual Reality system are summarized in table 1. To reduce the intensity of reflections from the walls we attached acoustic absorber plates inside our virtual reality lab.

The software application for the visual stereoscopic and acoustic rendering of the simulation results on the hardware described above is implemented in C++ using QT[1] and OpenGL[2] APIs for GUI programming and rendering. Particularly, we used QT's client-server concept for controlling the graphical representation. The render nodes are are responsible for drawing the view for the left and right eye, respectively. To ensure that both rendered images are displayed simultaneously for a flicker-free representation of the scene, synchronization of the render nodes is required. Some APIs for parallel rendering already exist, for example Chromium[3], which is a system for interactive rendering on clusters.

To avoid restrictions imposed by existing synchronization software packages, a simple method based on sockets and full hand-shake synchronization is implemented. Nevertheless, since pure OpenGL is used for graphic rendering the application of Chromium or another VR API's (e.g. VR Juggler[4], DIVERSE[5]) in order to be flexible to use another VR hardware configuration is straightforward. The full hand-shake synchronization is realized as follows. The master sends command messages to the render nodes. When both renderer replied on a successful execution of the instruction, a new command can be send. In Fig. 4 the loading process of a room geometry from a file is conceptually shown. The master sends the command for loading the scene from file (*load*

[1] http://trolltech.com/
[2] http://www.opengl.org/
[3] http://chromium.sourceforge.net/
[4] http://www.vrjuggler.org
[5] http://diverse-vr.org/

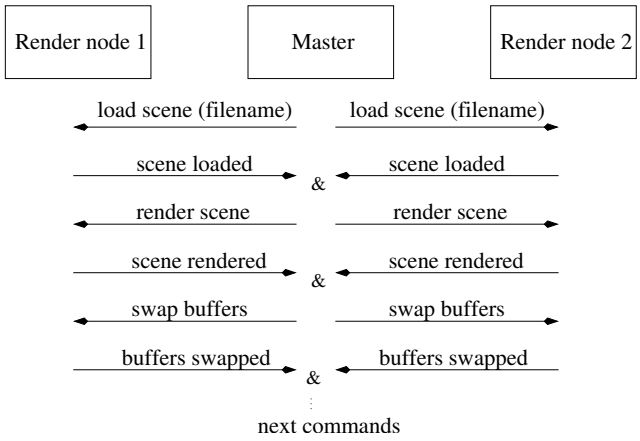


Figure 4 Full hand-shake synchronization for rendering

scene). After the affirmative reply from both render nodes, instruction for rendering follows (*render scene*). Is this successfully executed and confirmed, the buffers can be swapped now for the final representation of the left eye and right eye views on the display. Before the master can send new instructions to render nodes, it needs to wait for the response to the previous message, even in the case of single operations like swapping buffers. Otherwise, two consecutive messages sent by the master may collide causing a serious time delay.

For auralization purpose we used the synthesis method described in Section 5. In order to drive the loudspeaker of the audio system separately with the corresponding signals the PortAudio library[6] for acoustical output is utilized. The audio playback is administrated by the master computer and is started in a new process (*thread*).

4 Room Acoustic Simulation

4.1 Phonon Tracing Approach

Photon mapping [18, 19] is often used for rendering photo-realistic images, supplementing uni-directional raytracing by a variety of visual effects, like color bleeding and caustics. We adopt a similar approach to the simulation of sound, named phonon tracing [5], which is summarized in following.

Problem Specification

Our simulation algorithm requires the following input information:

- position of sound source s,
- emission distribution E of sound source,
- one ore more listener positions l_i,
- a triangulated scene with tagged material m_j,
- an absorption function $\alpha_j : \Omega \mapsto (0,1]$ for each material,
- an acoustic BRDF for each material (if applicable),
- an energy threshold ε for terminating the phonon paths.

[6] www.portaudio.com

The output of our approach is a FIR filter f_i for each listener's position l_i corresponding to the impulse response with respect to the sound source and the phonon-map containing for each phonon the energy spectrum e_p, the traversed distance d_p, the phonon's position p_p at the reflection point, its outgoing direction v_p, number of reflections r_p, and the material m_p at the current reflection.

Our simulation algorithm is executed in two stages, the *phonon tracing* step constructs the phonon map, and the *phonon collection and filtering* step collects the phonon's contribution to a FIR filter for every listener position.

Phonon Tracing

Every phonon p emitted from the sound source carries the following information:

- an energy spectrum $e_p : \Omega \mapsto \mathbb{R}^+$,
- the distance d_p traversed from the source,
- the phonon's current position p_p,
- the outgoing direction v_p.

Our absorption and energy functions α_j are represented by $n_e = 10$ coefficients associated with the frequencies $40, 80, 160, ..., 20480$ Hz. The basis function for the energy spectrum are wavelets adding up to a unit impulse. Every phonon is composed of different frequencies, which is more efficient than tracing a single phonon for each individual frequency band.

Phonons are emitted from the source s according to the emission probability distribution E and have at starting point a unit energy spectrum $e_{p,i} = 1$ $(i = 1, ..., n_e)$. At the intersection of the phonon ray with the scene, the phonon direction d_p is reflected with respect to the surface normal and the absorbed energy is subtracted according to the local material m_j, and the distance d_p is set to the traversed distance. The phonon is fixed at the intersection point, contributing to a global phonon map.

If the maximal energy of the phonon exceeds the energy threshold, i.e. $max\{e_{p,i}\}_{i=1}^{n_e} > \varepsilon$ and a minimal number of reflections is achieved, the next phonon re-uses the path and energy of the preceding one, saving computation time. It started at the current position with respect to the outgoing direction d_p and contributes to the phonon map at next surface intersection. If the threshold is not exceeded and a minimum number of reflections has been computed, the a new phonon is stated from the source. After a prescribed number n_p of phonons have contributed on the global phonon map, the tracing is terminated. The phonon map is used for further visualization purposes on our virtual reality system.

Phonon Collection and Filtering

The remaining task of the phonon tracing method is collecting the phonon's contribution to a FIR filter f for every listener's position l. This filter corresponds to the impulse response from the source, recorded at l, such that convolution with an anechoic signal, reproduces the perceived signal.

In the case of uniform absorption for all frequencies, the contribution of a phonon visible from the listener is simply a scaled, translated unit pulse (Dirac). The Dirac is shifted by the time elapsed between emission and reception of a phonon and scaled by the phonon's energy $e_{p,i}$ multiplied by a gaussian weighting the distance of the ray to the listener. In classical acoustic ray tracing [25, 26], a sphere is used to collect rays at listener position. Using a gaussian, however, provides much smoother filters, since more phonon rays contribute to the filter, weighted by their shortest distance.

In the more general case of frequency-dependent absorption, the unit impulse is subdivided into wavelets representing the individual frequency bands. The filter becomes the a sum of this wavelets scaled by $e_{p,i}$ and shifted by the elapsed time. In our implementation we use 10 frequency bands and absorption coefficients for the frequencies $\omega_i = 20 \cdot 2^i$ Hz $(i = 1, ..., 10)$. We construct band-pass filters in spectral domain by means of cosine functions in order to obtain quickly decaying wavelets.

The wavelets are computed by the inverse Fourier transform. Our band-pass filters have the following properties:

- compact support in the frequency domain Ω,
- symmetry and smoothness in both domains,
- their sum is one in Ω and a Dirac in the time domain.

The impulse response filters produced by our implementation are sampled at a rate of 48 kHz. For generating the filter bank, we used 2^{14} samples, employing the inverse FFT for computing the discretized wavelets. After the computation the response filter is normalized. A complete description of our phonon tracing algorithm, particularly addressing the filter design, can be read in more detail in [5].

4.2 Finite Element Method (FEM)

Phonon tracing or any other method based on geometric acoustics (ray tracing, mirror image) fail in the low frequency range for two reasons:

1. Wavelengths are of the order of typical dimensions of the room. Hence, diffraction can no longer be neglected.
2. Damping is typically low at low frequencies and reverberation times become too long to be represented by a convolution kernel of reasonable length.

Therefore, we have to fall back on wave acoustics to simulate the low frequency part of the sound field. For closed rooms this is preferably done by the finite element method (FEM), which approximates the wave equation by a large system of ordinary differential equations (ODEs) the unknowns of which are the pressures at grid points covering the room. In general, there are by far too many unknowns to solve these systems of ODEs in real time. Hence, we need to reduce the system to a concise state-space model with similar input-output behavior in the frequency range of interest.

There are many different approaches to model reduction [2]. The common observation is that system dynamics can often be represented quite well by a superposition of a few (generalized) eigen-modes. The coefficients of these modes are the unknowns of the new reduced system. Finally, assuming samplewise constant input (e.g. acceleration of the loudspeaker membrane), the continuous state-space model is transformed into a discrete one, which can be solved in real time. In practice, a low pass filter is used to split the input signal into a low frequency part and a remainder. The low frequency part (< 300 Hz) is handled in the way described here and the remainder by phonon tracing.

In the following we list the steps to get from the wave equation (1) to a reduced discrete state-space model (5) describing the transient response of a room to an excursions of a loudspeaker membrane. The wave equation and associated boundary conditions read:

$$\begin{aligned} \frac{\partial^2 p}{\partial t^2} - c_0^2 \Delta p &= 0 \quad \text{on} \quad G \\ c_0 \frac{\partial p}{\partial n} &= -\frac{1-R}{1+R} \frac{\partial p}{\partial t} \quad \text{on} \quad \Gamma_w \\ \frac{\partial p}{\partial n} &= -\rho_0 \frac{\partial^2 x_m}{\partial t^2} \quad \text{on} \quad \Gamma_m \,. \end{aligned} \tag{1}$$

$p = p(t,x)$ denotes pressure, c_0= 343 m/s the velocity of sound, and ρ_0 =1.2 kg/m the density of air at room temperature, G the interior of the room, and Γ_w and Γ_m the surfaces of walls and membrane, respectively. x_m is the excursion of the membrane and R is a reflection coefficient. It may depend on the particular wall, but is constant for all frequencies. This is a minor problem, as we use the model only for a small frequency band.

Approximating the pressure distribution by a superposition of, for instance, piecewise quadratic ansatzfunctions $p(t,x) = \sum_{i=0}^{N} p_i(t)\, \varphi_i(x)$ and integrating (1) with respect to the φ_i gives a FE model

of the form:

$$\begin{aligned} M\ddot{p} + D\dot{p} + Kp &= Fu \\ y &= Pp\,. \end{aligned} \tag{2}$$

The real $N \times N$ matrices M, D, K are called mass, damping, and stiffness matrix. $p = p(t)$ is a vector composed of the coefficients p_i. $u = u(t)$ is the input, e.g. the acceleration of the membrane. F transforms this input into a force. P is a projection matrix extracting certain interesting pressures y_i. Setting

$$\hat{x} = \begin{bmatrix} p \\ \dot{p} \end{bmatrix}, \quad \hat{E} = \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}, \quad \hat{A} = \begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} 0 \\ F \end{bmatrix}, \quad \textit{and} \quad \hat{C} = \begin{bmatrix} P & 0 \end{bmatrix}$$

the FE model may be rewritten as a state-space model:

$$\begin{aligned} \hat{E}\dot{\hat{x}} &= \hat{A}\hat{x} + \hat{B}u \\ y &= \hat{C}\hat{x}\,. \end{aligned} \tag{3}$$

Assuming that $\hat{x}$ is essentially composed of the columns of a matrix $U \in \mathbb{R}^{N \times n}$, $n \ll N$, and projecting the equations on the columns of $V \in \mathbb{R}^{N \times n}$ we end up with a reduced state-space system where

$$\tilde{x} = U\hat{x}, \quad \tilde{E} = V^t \hat{E} U\,, \quad \tilde{A} = V^t \hat{A} U\,, \quad \tilde{B} = V^t \hat{B}\,, \quad \tilde{C} = \hat{C}U\,. \tag{4}$$

The columns of U and V may be found by expanding the associated transfer function $H(s) = \hat{C}(s\hat{E} - \hat{A})^{-1}\hat{B}$ about some shifts $s_j = \mathrm{i}\omega_j$. Here, we used the *rational dual Arnoldi* algorithm described in [35]. Finally, dividing the first equation of the reduced version of (3) by $\tilde{E}$ and performing a balanced truncation [48] we end up with a state-space system of typically a few hundred unknowns rather than several 10,000 degrees of freedom. Integrating the reduced version of (3) over the length Δt of one sample for constant input u_n, i.e.

$$x_n = \mathrm{e}^{\tilde{A}\Delta t} x_{n-1} + \int_0^{\Delta t} \mathrm{e}^{\tilde{A}(\Delta t - \tau)}\, \mathrm{d}\tau\, \tilde{B} u_{n-1},$$

leads to a discrete state-space system

$$\begin{aligned} x_n &= Ax_{n-1} + Bu_{n-1} \\ y_n &= Cx_n\,. \end{aligned} \tag{5}$$

Switching to a suitable basis $x_n = T\xi_n$ it is always possible to turn the system into *companion canonical form* [23], where the first $n-1$ rows of A coincide with the last $n-1$ rows of the identity matrix of order n, the last row contains the negative coefficients of the characteristic polynomial of A, and B is the n-th unit vector. Hence, updating the state vector x_n and evaluating the pressure at a certain position requires $2n$ multiplications and $2n-1$ additions. Below, we will present an example where $n = 149$, i.e. 595 floating point operations are needed per new sample. On the other hand, if the sampling rate is 48,000 and we are using a convolution kernel instead which can represent at least a full wave of a 20 Hz signal, we need at least 4799 floating point operations.

Numerical Example

We consider a small room with two doors, a heating covered by absorbing material, walls made of bricks or concrete, an absorbing ceiling and a loudspeaker in one of the corners, cf. Fig.5. We assume a reflection coefficient $R = 0$ at the heating (total absorption), $R = 0.8$ at the ceiling, and $R = 1$ (total refection) at all the other surfaces. The original FE model contains 29272 unknowns and was created

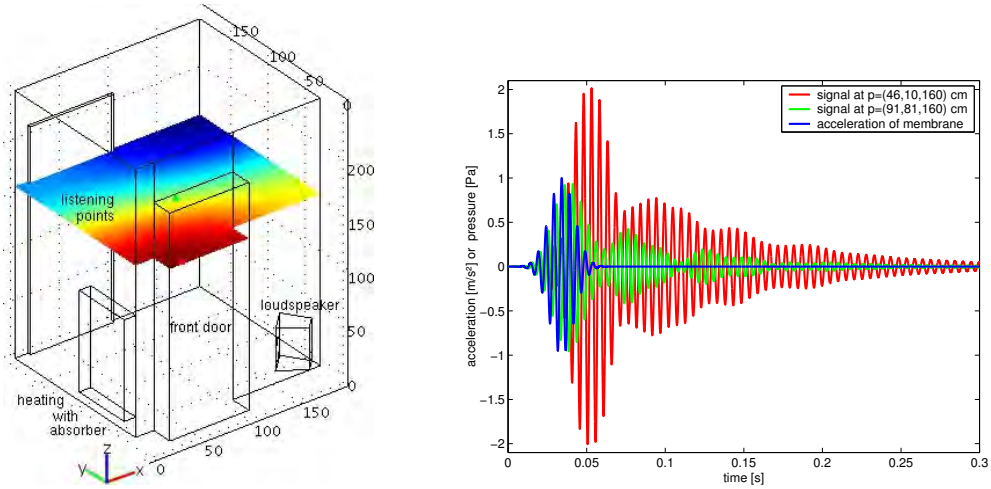


Figure 5 Response to a low frequency beep in a small weakly absorbing room .

with FEMLAB. Model reduction was done in MATLAB and yielded a model of order 145. Virtual measurements are taken in a plane 160 cm above the floor. The fourier transform of the input signal to the loudspeaker is a Gaussian centred about 200 Hz with standard deviation of 20 Hz. Responses are illustrated for a virtual microphone in the small indentation at the front door and in the middle of the room. Note that the indentation acts as a resonator. A standing wave is excited which oscillates between front door and opposite corner while taking low values in the center of the room. Close to the front door reflections accumulate in such a way that the maximum amplitude is reached when the loudspeaker has already stopped playing.

5 Acoustic Synthesis

Let $\tilde{p}_i \in \mathbb{R}^{2n_b}$ a section of the digital signal simulated in point $\tilde{x}_i \in \mathbb{R}^2$ of the horizontal measuring plane cutting the virtual room. n_b is the block length. We choose a power of 2 to speed up later Fourier transforms. The n_p points are distributed equally on a small circle of radius r about the virtual head position. The radius has to be of the same order as the wave length corresponding to the highest frequency to be reproduced:

$$r \approx \lambda_{\min} = \frac{c_0}{f_{\max}} \,. \tag{6}$$

For $f_{\max}$= 20,000 Hz and a sound velocity of c_0= 343 m/s the radius is only 12 mm. The exact radius depends on the number of loudspeakers n_l. To get an idea, imagine that an arbitrary sound field of frequency $f_{\max}$ has to be approximated by a polynomial with n_l coefficients which, for small n_l, will be valid only within a small radius. Choosing r this small we avoid artefacts in reproducing the high frequency part of the sound field while, at lower frequencies, the approximation will still be good also for larger radii.

The system for sound field synthesis is made to reproduce sound in n_p fixed positions on a ring of radius r about a point in the projection room. As the proband may change the line of vision in the virtual room these points cannot be directly associated with the n_p points from the fixed grid where sound is simulated. Hence, the simulated samples $\tilde{p}_{i,n}$ of time step n are interpolated and evaluated at the rotated real positions x_i using cubic splines in the angle.

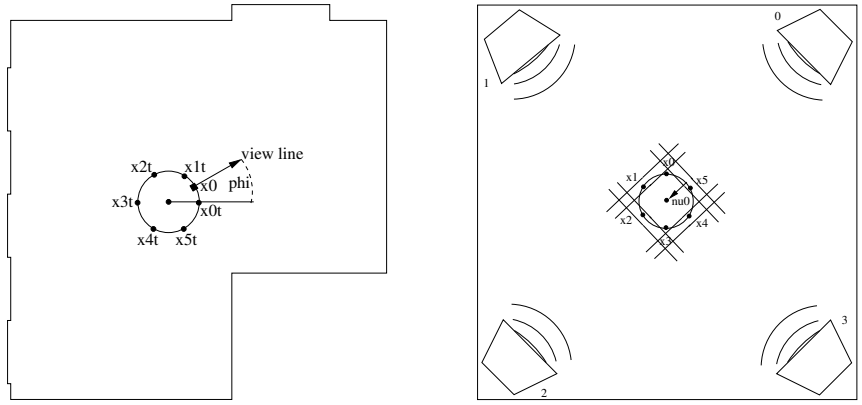


Figure 6 Virtual and projection room.

The resulting signals $p_i \in \mathbb{R}^{2n_\mathrm{b}}$ are first windowed by a lifted cosine and then subjected to a fast Fourier transform:

$$\hat{p}_i = \mathrm{fft}(p_i^w), \quad p_{i,n}^w = \tfrac{1}{2}\left[1 - \cos\frac{\pi(n+\frac{1}{2})}{n_\mathrm{b}}\right] p_{i,n} \,. \tag{7}$$

Synthesizing the sound field we assume that the loudspeakers induce plane waves in the central listening position and that time delay and amplification are the same for all loudspeakers. These conditions are met fairly well as we use an anechoic projection room and the speakers are furnished with FIR filters and placed on a circle. However, in a later version the plane-wave assumption will be replaced by using measured impulse responses. So far, the Fourier coefficients $\hat{q}_{j,k}$ of the signal of the j-th speaker are computed to satisfy:

$$\hat{p}_{i,k} = \sum_{j=1}^{n_\mathrm{l}} \hat{q}_{j,k} \exp\left(-\mathrm{i}\frac{\omega_k}{c_0}\langle \nu_j, x_i\rangle\right) \,. \tag{8}$$

$\omega_k = \frac{\pi f_\mathrm{s}}{n_\mathrm{b}} k$ is the k-th angular frequency, f_s the sampling rate, and ν_j the radiation direction of the j-th speaker. This leads to a linear system for each frequency:

$$A_k \hat{q}_{.k} = \hat{p}_{.k} \,. \tag{9}$$

For stability reasons the number n_p of reproduction points is chosen greater than the number n_l of speakers and (9) becomes overdetermined. Moreover, in order to keep speaker signals bounded we introduce a small regularization parameter α:

$$\hat{q}_{.k} = T_k \hat{p}_{.k}, \quad T_k = [A_k^* A_k + \alpha \mathrm{I}]^{-1} A_k^* \,. \tag{10}$$

The matrices T_k are computed once for each frequency when initializing the system. As the speaker signals will be real we have $\hat{q}_{.,k} = \hat{q}^*_{.,n_\mathrm{b}-k}$ and only half of the systems need to be solved. Next, the $\hat{q}_j$ are retransformed into time domain by the inverse Fourier transform to get q_j^w. The superscript w indicates that these speaker signals will reproduce only the windowed measurements. To get the final speaker signals, the first half of q_j^w is added to the last half of the q_j^w computed for the last block:

$$q_j = q_{j,1\ldots n_\mathrm{b}}^{w,\mathrm{new}} + q_{j,n_\mathrm{b}+1\ldots 2n_\mathrm{b}}^{w,\mathrm{old}} \,. \tag{11}$$

Finally, $q^{w,\text{new}}_{j,n_b+1\ldots 2n_b}$ is stored for adding in the next step. The following table summarizes the algorithm:

1. *Get next block of simulated measurements.*
2. *Per time step: interpolate simulated measurements at reproduction points of projection room.*
3. *Scale measurements in projection room by lifted cosine-function.*
4. *Fast Fourier transform of scaled measurements in projection room.*
5. *Per frequency: compute loudspeaker signals to optimally reproduce measurements in a regularized least squares sense.*
6. *Inverse fast Fourier transform of loudspeaker signals.*
7. *Add first half of loudspeaker signals to last half stored for last block.*
8. *Store last half of loudspeaker signals to add in the next step.*
9. *Goto 1.*

6 Visualization

Our first visualization method described in [5] focuses on the spatial propagation of a sound wave from the source. The corresponding wave front traverses the room and is reflected on surfaces, altering its intensity and energy spectrum. We visualized this sound waves by rendering small spheres representing the sound particles. These are color coded by means of their spectral energy. Therefore, we use the RGB components, such that blue corresponds to the average of the energy by 40, 80, 160, 320 Hz, green corresponds to the average of energy by 640, 1280, 2560 Hz, and red to the average by 5120, 10240, 20480 Hz. When sliding through time, the spheres follow the simulated phonon paths. We integrated following functions in our interactive visualization system:

- varying the percentage of phonons to be rendered,
- rendering only the phonons reflected from a selected material,
- exchanging selected materials,
- varying time / transversed distance.

The data structure supporting this visualization is an array of phonons carrying their energy spectrum e_p, the traversed distance d_p, the phonon's position p_p at a reflection point, and its outgoing direction v_p, according to section section 4.1. In addition, we record the number of reflections r_p and the material m_p at the current reflection. Since all phonons sharing the same path are listed consecutively in the array, it is simple, for example, to select all consecutive pairs p_i, p_{i+1} where the current time t satisfies $t_{pi} \leq t < t_{pi+1}$ and to draw a sphere on the line segment p_{pi}, p_{pi+1}, corresponding to a phonon's location at time t.

The exchange of a certain material requires only the phonons' energy to be re-evaluated, where the phonon paths remain fixed. To allow the exchange, it is necessary to enforce a minimum number of reflections for every path in advance, since otherwise materials with high absorption coefficients cannot be replaced. An application scenario of this kind is provided in the next section.

To improve the visualization of sound propagation to be able to better distinguish between different reflected wave fronts and still preserving the scalability we use surface elements (surfels) [38] for particle representation. In Fig. 7 a sketch of surfels is depicted. The surfel representation can be adjusted according to users needs on accuracy as well as graphics hardware capabilities. The level of detail can be changed by prescribing the number of points describing the circumcircle of the disk (from quad to circle in Fig. 7).

In order to depict the propagated sound wave front, a surfel is rendered on each particle position at given time. The size of the surfel is given by the radius r_i and the orientation by the normal n_i

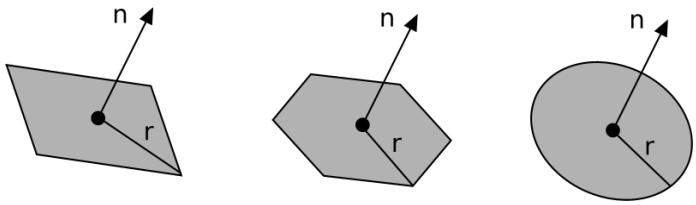


Figure 7 Sketch of surface elements (surfels) at different levels of detail.

which corresponds to the sound particle traveling direction. The radius r_i varies depending on the number n_{ph} of phonons to be rendered (given by the user) as well as on the traversed distance l of the sound wave (time step given by the user):

$$r_i = s\sqrt{\frac{l}{n_{ph}}}\,. \tag{12}$$

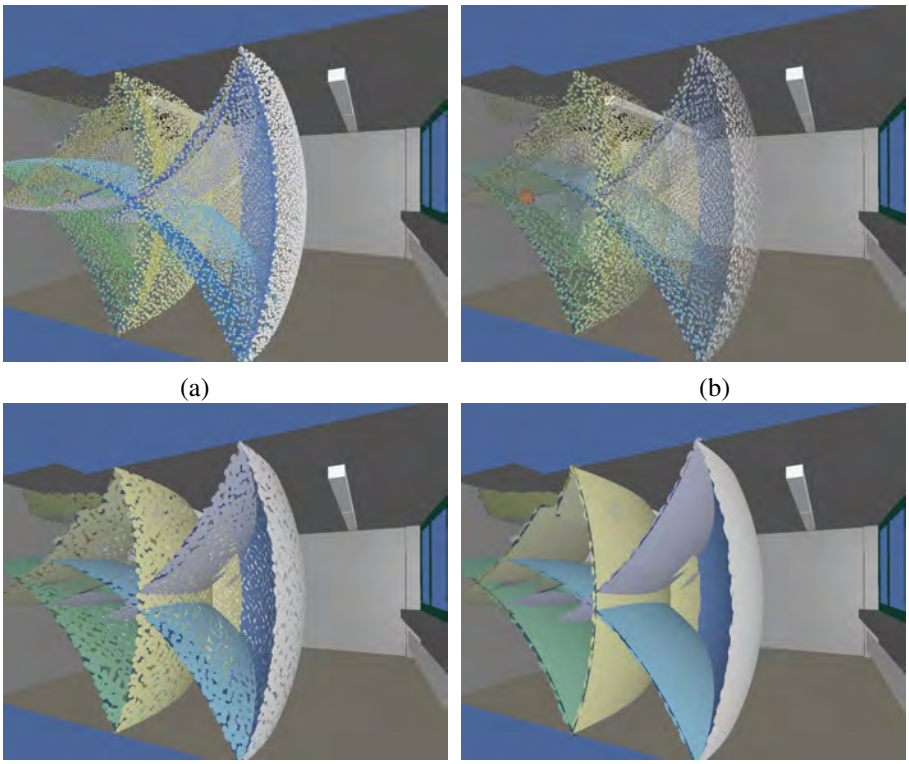
The radius r_i can also be expanded by the scaling factor s in order to obtain a surface rendering of the sound wave front. A small radius r_i results in a visualization on the wave front similar to that using spheres, whereas a large r_i leads to a surface-like representation. An example of sound wave propagation visualization is depicted in Fig. 8.

7 Visualization Application

The visualization approaches described in the previous section were used to examine the acoustic properties of a virtual reality (VR) laboratory that is also used for auralization purposes. Fig. 10 shows the wave front propagation based on 30000 paths traced from the sound source resulting in 910610 phonons in the phonon map, at a traversed distance of 1.5, 4.5, and 10 m. At small distances / traversal times, the individual wave fronts can be recognized, whereas large distances provide insight into the frequency decomposition of the various reflections. We observe a shift towards lower frequencies, since the particles color is dominated by the blue component after a number of reflections.

In order to identify the reason for the frequency shift, we look at sound particles reflected from a selected material in their earlier paths. Fig. 9 shows these particles for reflections from walls, floor, and canvas, respectively. We observe that particles reflected from walls carry mostly a yellowish color, despite of their potential reflection from additional materials. Hence, the energy of these particles is shifted towards the mid and high frequencies. The floor and the canvas reduce high frequencies. Reflections from the canvas affect mostly the right side of the room, whereas the impact of the carpet is much greater. While a potential frequency shift can already be seen in material absorption coefficients, their impact on room acoustics can be studied much better with the aid of our visualization approach.

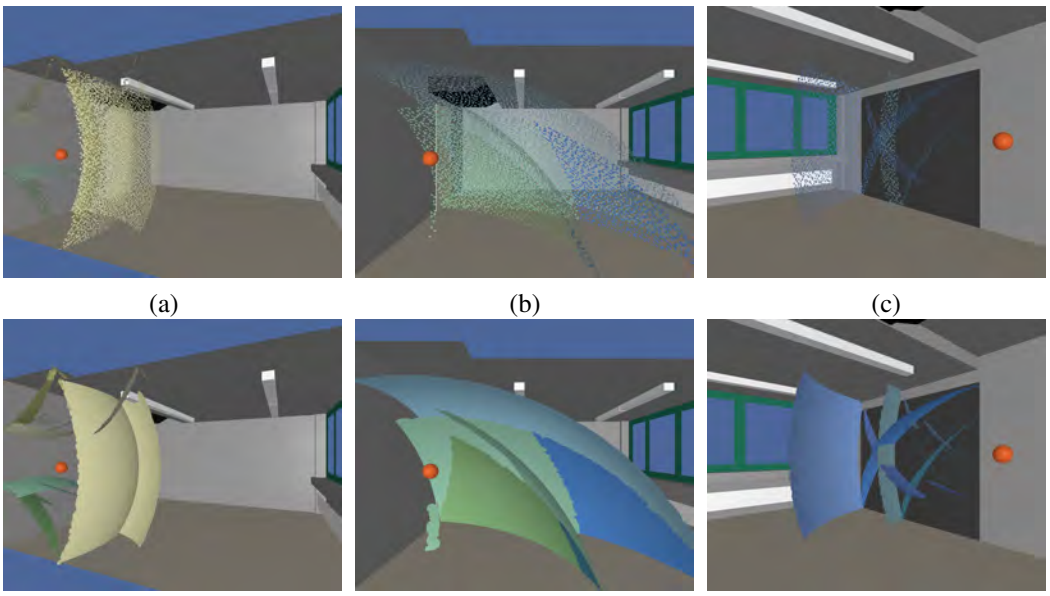
In Fig. 11 the carpet is replaced by a material with similar absorption as the walls. When comparing Fig. 11(a) and 11(b), it becomes evident that this change is sufficient for increasing the intensities of mid and high frequencies. This interactive visual observation of intensities changes because of material modifications adds significantly to the acoustic properties improvement during the design process. While the acoustics of the laboratory are not much of importance, it may have a greater impact on the design of larger classrooms. Optimizing the acoustics of such larger rooms may, for example, eliminate the need of using a microphone or improve the auditive quality of concerts.



(a) (b)

(c) (d)

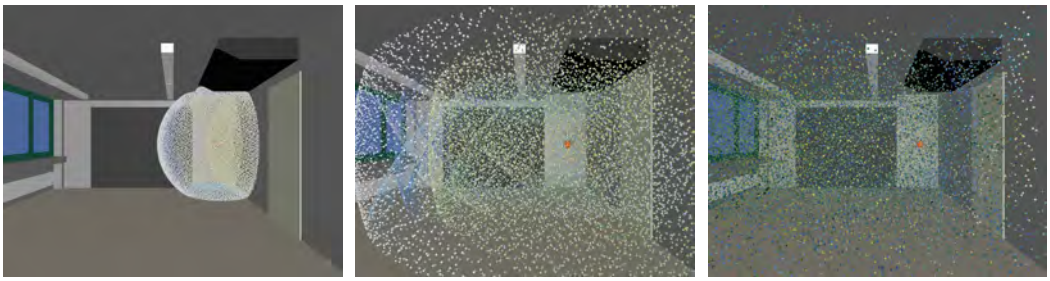
Figure 8 Wave front representation using surface elements (surfels) in contrast to spheres. Spheres used for rendering of phonons (a). Surfels with increased radius from (b) to (d).



(a) (b) (c)

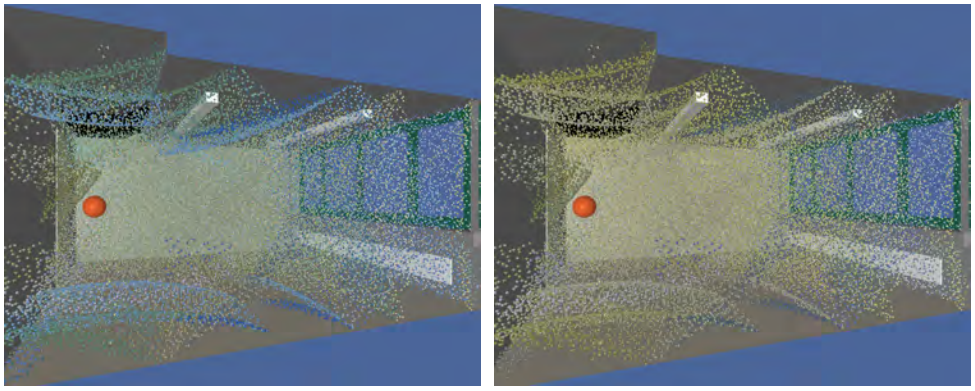
(d) (e) (f)

Figure 9 Wave fronts reflected from walls (a+d), floor (b+e), and projection wall (c+f) using smaller (a,b,c) and greater (d,e,f) radius.



(a) (b) (c)

Figure 10 Wave front propagation from a spherical sound source. The visualization shows phonons with RGB color-coded spectral energy due to reflection at different materials at 1.5m (a), at 4.5m (b), and at 10m (c).



(a) (b)

Figure 11 Changing material of the floor. (a) all phonons at d = 4.5m; (b) same as (a) with new material.

8 Conclusions

In this work we have presented an audio visual Virtual Reality system for room acoustics. For acoustic simulation purposes we have combined a geometric approach, the phonon tracing, and a FEM based solver. Sound rendering is performed utilizing a professional sound equipment using our synthesis approach. Furthermore we have introduced visualization techniques for sound wave propagation from the sound source. For visual rendering a stereoscopic 3D back projection display is used. The system enables the exploration of acoustical behavior inside a room visual as well as aural allowing the feasibility to improve the acoustics of the room during the design process. Future work includes continuative methods visualizing well defined room acoustic metrics as well as the visualization of the low frequency part simulated by means of FEM. Further work on integration of human machine interfaces like 3D mouse, game port or haptic devices would improve the interaction in the virtual environment.

Acknowledgments

This work was supported by the Stiftung Rheinland Pfalz für Innovation under contract no. 15202-386261/644.

References

1 J.B. Allen and A. Berkeley. Image method for efficiently simulating small-room acoustics. *J. Acoust. So. Amer.*, 65(4):943–950, Apr. 1979.

2 A.C. Antoulas and D.C. Sorensen. Approximation of large-scale dynamical systems: An overview. Technical report, Rice University, 2001.

3 I.and T. Lentz Assenmacher, T. Kuhlen, and M. Vorländer. Integrating real-time binaural acoustics into VR applications. In *Eurographics Symposium on Virtual Environments*, pages 129–136, Grenoble, June 8–9 2004.

4 A.J. Berkhout, D. de Vries, and P. Vogel. Acoustic control by wave field synthesis. *J. Acoust. Soc. Am.*, 93(5):2764–2778, 1993.

5 M. Bertram, E. Deines, J. Mohring, J. Jegorovs, and H. Hagen. Phonon tracing for auralization and visualization of sound. In *IEEE Visualization*, pages 20–30, Minneapolis, MN, October 2005.

6 J. Borish. Extension of the image model to arbitrary polyhedra. *J. Acoust. So. Amer.*, 75(6):1827–1836, 1984.

7 P. Bourke. 3d stereo rendering using opengl (and glut). *http://astronomy.swin.edu.au/~pbourke/opengl/stereogl/*, 1999.

8 L. Cremer and H.A. Müller. *Die wissenschaftlichen Grundlagen der Raumakustik, Band I*. S. Hirzel Verlag Stuttgart, 1978. 2. völlig neubearbeitete Auflage.

9 E. Deines, M. Bertram, J. Mohring, J. Jegorovs, F. Michel, H. Hagen, and G.M. Nielson. Comparative visualization for wave-based and geometric acoustics. In *IEEE Visualization 2006*, pages 1173–1179, Baltimore, Maryland, USA, October 29 – November 3 2006.

10 E. Deines, F. Michel, M. Bertram, H. Hagen, and G. Nielson. Visualizing the phonon map. In *Eurographics / IEEE-VGTC Symposium on Visualization*, Lisbon, Portugal, Mai 8-10 2006.

11 Y. Fukushima, H. Suzuki, and A. Omoto. Visualization of reflected sound in enclosed space by sound intensity measurement. *Acoust. Sci. & Tech.*, 27(3):187–189, 2006.

12 Thomas A. Funkhouser, Ingrid Carlbom, Gary Elko, Gopal Pingali Mohan Sondhi, and Jim West. A beam tracing approach to acoustic modeling for interactive virtual environments. In *Computer Graphics (SIGGRAPH 98)*, pages 21–32, Orlando, FL, July 1998.

13 Thomas A. Funkhouser, Patrick Min, and Ingrid Carlbom. Real-time acoustic modeling for distributed virtual environments. In *Computer Graphics (SIGGRAPH 99)*, pages 365–374, Los Angeles, August 1999.

14 W. Hackbusch. *Integralgleichungen, Theorie und Numerik*. Teubner Verlag, 1989.

15 L. Harrison, D. McAllister, and M. Dulberg. Stereo computer graphics for virtual reality. In *ACM SIGGRAPH '97*, pages 20–30, Minneapolis, MN, October 1997. Course Notes 6.

16 Larry F. Hodges. Tutorial: Time-multiplexes stereoscopic computer graphics. In *IEEE Computer Graphics & Applications*, pages 20–30, Orlando, FL, March 1992.

17 Marcin Jedrzejewski and Krzysztof Marasek. Computation of room acoustics using programable video hardware. In *International Conference on Computer Vision and Graphics ICCVG'2004*, Warsaw, Poland, September 22-24 2004.

18 Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques '96 (Proceedings of the 7th Eurographics Workshop on Rendering)*, pages 21–30, 1996.

19 Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scene with participating media using photon maps. In *Computer Graphics (SIGGRAPH 98)*, pages 311–320, July 1998.

20 B. Kapralos, M. Jenkin, and E. Millios. Sonel mapping: Acoustic modeling utilizing an acoustic version of photon mapping. In *IEEE International Workshop on Haptics Audio Visual Environments and their Applications (HAVE 2004)*, Ottawa, Canada, October 2-3 2004.

21 S. Khoury, A. Freed, and D. Wessel. Volumetric visualization of acoustic fields in cnmat's sound spatialization theatre. In *Visualization '98*, pages 439–442 & 562. IEEE, 1998.

22 B. Klehs and T. Sporer. Wave field synthesis in the real world part 1: In the living room. In *114th AES Convention*, Amsterdam, March 22 - 25 2003.

23 H.W. Knobloch and H. Kwakernaak. *Lineare Kontrolltheorie*. Springer-Verlag, 1985.

24 N. Korany, J. Blauer, and O. Abdel Alim. Acoustic simulation of rooms with boundaries of partially specular reflectivity. *Applied Acoustics*, 62:875–887, 2001.

25 U. Krockstadt. Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound and Vibrations*, 8(18):118–125, 1968.

26 U. Kulowski. Algorithmic representation of the ray tracing technique. *Applied Acoustics*, 18:449–469, 1984.

27 C. Lauterbach, A. Chandak, and D. Manocha. Interactive sound propagation in dynamic scenes using frustum tracing. In *In Proceeding of IEEE Visualization*, Sacramento, CA, USA, October 28 – November 1 2007.

28 T. Lokki and V. Nenonen. Imersive visualization of room acoustics. In *Joint Baltic-Nordic Acoustics Meeting*, Gothenburg, Sweden, November 8–10 2006.

29 Tapio Lokki. *Physically-based Auralization*. PhD thesis, Helsinki University of Technology, 2002.

30 Tapio Lokki, Peter Svensson, and Lauri Savioja. An efficient auralization of edge diffraction. In *Audio Engineering Society, 21th International Conference*, pages 166–172, St. Petersburg, Russia, June 2002.

31 J. Merimaa, T. Lokki, T. Peltonen, and M. Karjalainen. Measurements, analysis, and visualization of directional room responses. In *Proceedings of the 111th Audio Engineering Society (AES) Convention*, New York, NY, USA, September 21–24 2001.

32 M. Monks, B.M. Oh, and J. Dorsey. Acoustic simulation and visualization using a new unified beam tracing and image source approach. In *Convention of the Audio Engineering Society*. ACM, 1996.

33 M. Monks, B.M. Oh, and J. Dorsey. Audioptimization: Goal-based acoustic design. *IEEE Computer Graphics and Applications*, 20(3):76–91, 2000.

34 M. Naef, O. Staadt, and M. Gross. Spatialized audio rendering for immersive virtual environments. In *Symposium on Virtual Reality Software and Technology*, pages 65–72. ACM, November 2002.

35 K.H.A. Olsson. Model order reduction in FEMLAB by dual rational Arnoldi. Master's thesis, Department of Mathematics, Chalmers University of Technology and Göteborg University, 2002.

36 A. Omoto and H. Uchida. Evaluation method of artificial acoustical environment: Visualization of sound intensity. *Journal of Physiological Anthropology and Applied Human Science*, 23:249–253, 2004.

37 S. Petrausch and R. Rabenstein. Highly efficient simulation and visualization of acoustic wave fields with the functional transformation method. In *Simulation and Visualization*, pages 279–290, Otto von Guericke Universität, Magdeburg, March 2005.

38 H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, New Orleans, LA, USA, 2000.

39 V. Pulkki and T. Lokki. Visualization of edge diffraction. *Acoustics Research Letters Online (ARLO)*, 4(4):118–123, 2003.

40 Rudolf Rabenstein, Oliver Schips, and Alexander Stenger. Acoustic rendering of buildings. In *5th Int. IBPSA Conference Building Simulation*, volume 2, pages 181–188, Prag, Czech Republic, September 1997.

41 Lauri Savioja, Tapio Lokki, and Jyri Huopaniemi. Auralization applying the parametric room acoustic modeling technique - the diva auralization system. In *International Conference on Computer Auditory Display*, pages 219–224, Kyoto, Japan, July 2-5 2002.

42 A. Sontacchi and R. Hoeldrich. Enhanced 3D sound field synthesis and reproduction system by compensating interfering refelexions. In *COST G-6 Conference on Digital Audio Effects*, Verona, December 7–9 2000.

43 A. Stettner and D.P. Greenberg. Computer graphics visualization for acoustic simulation. In *International Conference on Computer Graphics and Interactive Techniques*, pages 195–206. ACM, 1989.

44 Y. Tokita and Y. Yamasaki. Visualization of 3-dimensional sound fields by numerical solutions of particle displacement. *Acoust. Sci. & Tech.*, 26(2):215–217, 2005.

45 R.R Torres, U.P. Svensson, and M. Kleiner. Edge diffraction in room acoustics computations. In *EAA Symposium on Architectural Acoustics*, Madrid, Spain, Oct. 16-20 2000.

46 M. Vorländer. Simulation of the transient and steady-state sound propagation in rooms using a new combined ray-tracing/image-source algorithm. *J. Acoust. So. Amer.*, 86(1):172–178, 1989.

47 T. Yokota, S. Sakamoto, and H. Tachibana. Visualization of sound propagation and scattering in rooms. *Acoust. Sci. & Tech.*, 23(1):40–46, 2002.

48 K. Zhou and J.C. Doyle. *Essentials of robust control*. Prentice-Hall inc., 1998.

Saliency Guided Summarization of Molecular Dynamics Simulations

Robert Patro[1], Cheuk Yiu Ip[1], and Amitabh Varshney[1]

1 **University of Maryland, College Park MD 20742, USA**
{rob,ipcy,varshney}@cs.umd.edu

Abstract

We present a novel method to measure saliency in molecular dynamics simulation data. This saliency measure is based on a multiscale center-surround mechanism, which is fast and efficient to compute. We explore the use of the saliency function to guide the selection of representative and anomalous timesteps for summarization of simulations. To this end, we also introduce a multiscale keyframe selection procedure which automatically provides keyframes representing the simulation at varying levels of coarseness. We compare our saliency guided keyframe approach against other methods, and show that it consistently selects superior keyframes as measured by their predictive power in reconstructing the simulation.

1998 ACM Subject Classification J.3 Life and Medical Sciences

Keywords and phrases Molecuar Dynamics, Saliency, Simulation

Digital Object Identifier 10.4230/DFU.SciViz.2010.321

1 Introduction

Molecular dynamics trajectories play a vital role in enhancing our understanding of the building blocks of life at the nanoscale. A number of recent advances in modeling and simulation of proteins and nucleic acids continue to provide us with novel insights into the relationship between the form and function of these dynamic biological nanomachines. In our efforts to simulate ever-more accurate models of physics and chemistry, such simulations out of necessity have to occur over very small time scales, typically femtoseconds. However, the major molecular conformational changes of interest typically occur over timescales ranging from a few microseconds to seconds. This difference in simulation timescales is being bridged by novel algorithmic approximations, advances in hardware, as well as by simply running longer time-scale simulations facilitated by larger storage capacities of modern computer systems [9].

However, just because we now have the ability to simulate exceedingly long timescale molecular dynamics simulations, does not necessarily mean that we are better equipped to gain visual insights using such simulation datasets. Since the capabilities of the human visual system remain unchanged and the bandwidth into the human cognitive machinery remains constant, we have now reached a stage where the current generation simulation datasets can easily overwhelm the limits of human comprehension. In real world, the human visual system deals with the glut of information coming at it from the world by focusing retinal hardware and attention on what is most important, or salient. The challenge of visual presentation and analysis of very large datasets compels us to re-examine not just how to present data, but what data to present. In this paper we discuss some of our recent research that deals with how to effectively summarize large molecular dynamics trajectories using ideas inspired by the visual saliency mechanism of the human visual system.

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 321–335

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

The main contributions of this paper are:

- We present a multiscale saliency operator for molecular dynamics trajectories that is successful at identifying the most salient time steps of a molecular dynamics simulation.
- We show how one can find the most representative frames of a molecular dynamics trajectory by simply inverting the multiscale saliency operator.
- We quantitatively show the benefits of using our multiscale saliency operator to summarize molecular dynamics trajectories compared with other methods such as a random scheme or the Douglas-Peucker scheme.
- We validate our methods using several real-world examples of long time scale molecular dynamics trajectories.

We believe that the research directions that we have identified in this paper are early-stage efforts that will hopefully spark a number of follow-on methods to automatically or semi-automatically identify and visualize salient features, events, and trends in very large-scale time-varying datasets. Our methods, in turn, build upon the seminal work of several other prominent researchers in the field. We give a summary of the related research in the next section.

2 Background and Related Work

The last few years have seen a growing interest in summarization of large-time varying datasets. Perhaps the greatest amount of research has been in abstraction [14] and summarization of videos [3]. However, much of the research on video summarization is not directly applicable to summarization of 3D datasets. In graphics, very interesting work has been done in summarization of articulated characters [2] as well as their compression [11, 1]. In volume visualization, Silver and Wang [12] have used the framework of template matching to identify key features such as reconnection events in large time-varying 3D volumes, such as those arising from computational fluid dynamics (CFD) simulations.

Almost all the previous work in characterization of time-varying datasets has been with either articulated skeletal human models or time-varying volume datasets. Although these methods provide helpful insights, they do not directly carry over for use in molecular dynamics simulations. This is because, unlike molecular dynamics simulations the movements in character animation are purposive, described by a set of continuously changing joint angles. Thus an event of interest in character animation can be detected by a change of such movements. Molecular dynamics simulations are characterized by a variety of motions at multiple scales. There are fine-scale Brownian motions and larger-scale conformational changes. To handle this challenge, we define a multiscale saliency operator that works with several different-sized sliding windows.

Ideally, we would like to identify the most important time steps in large time-varying molecular dynamics simulations for the purposes of summarization, fast-previewing, indexing, and further analysis. We build upon the ideas of image saliency by Itti *et al.* [8] and mesh saliency by Lee *et al.* [10]. They use a center-surround operator to identify the uniqueness of a pixel or a vertex with respect to its neighborhood. In our approach, we define the importance of a timestep by its difference from its neighbors, both forward and backward in time, over multiple scales. Specifically, we analyze a difference of Gaussian weighted average positions centered around each timestep for several different scales. Subsequently, we combine this information into a single multiscale saliency function $\mathcal{S}$, and present a multiscale keyframe selection procedure to obtain representative (or conversely, anomalous) frames based upon this function. Further details of our approach are in Sections 3 and 4.

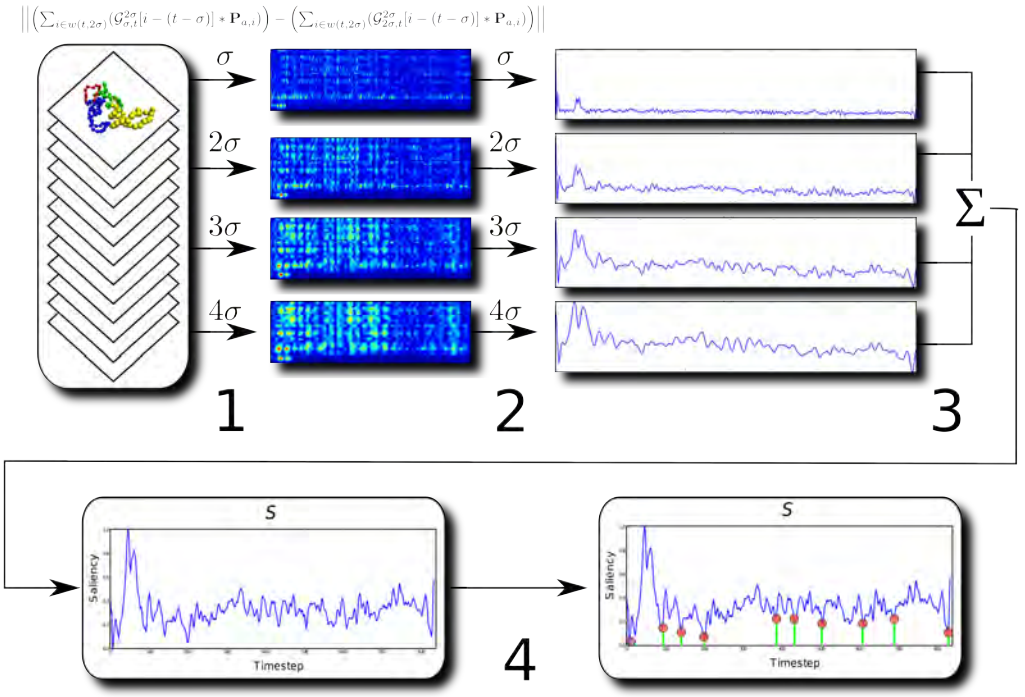


Figure 1 This figure gives an overview of our salient timestep selection procedure. In step 1, the per-atom saliency measure is taken at multiple scales. Step 2 then composites these per-atom saliency functions into a single per-timestep saliency function for each scale. In step 3, the separate single-scale saliency functions are combined into a single multiscale saliency function. Finally, in step 4, the multiscale keyframe selection (**MSKS**) procedure is used to select representative timesteps from the simulation using the multiscale saliency function.

3 Multiscale Saliency for Molecular Dynamics Trajectories

The notion of saliency is general. Indeed, even in the particular domain of molecular dynamics simulations, one might reasonably suggest many methods by which to compute saliency. In order to focus the scope of this work, we shall consider saliency on a per-timestep basis. In section 4, we consider the representation of simulation data through the selection of key timesteps. Thus, we are motivated to define our saliency measure on a per-timestep basis. Inspired by the saliency mechanism of the human visual system, we have decided to formulate our saliency operator for molecular dynamics trajectories to be multiscale and to make use of a center-surround mechanism [8].

Even among those measures adhering to these criteria, there are numerous possible definitions of per-timestep saliency. However, since the notion of saliency in the domain of molecular dynamics simulations has not been well explored, we shall introduce a simple but straightforward definition, which we have found to be very effective. An overview of our method is given in figure 1.

3.1 Saliency Definition

In order to define our notion of saliency, it will be necessary to introduce some notation. Our main analysis will be on an order-3 tensor $\mathbf{P}$, containing the position of each atom for every timestep of the simulation data. For a simulation with n timesteps and m atoms, $\mathbf{P}$

will be an $m \times n \times 3$ tensor, such that $\mathbf{P}_{a,t,-} = \vec{p_{a,t}}$ is the 3-vector containing the Cartesian coordinates of atom a at timestep t. For the sake of brevity, we shall write $\mathbf{P}_{a,t,-}$ as $\mathbf{P}_{a,t}$. It will also be useful to index ranges of $\mathbf{P}$. The expression $\mathbf{P}_{I,J}$ will be used to denote a block of values from $\mathbf{P}$ spanning atoms $I = [i_1, i_2, \dots]$, timesteps $J = [j_1, j_2, \dots]$, and all spatial coordinates.

Further, it will be useful to define the notion of a selection window. Let $w(i, \sigma) = [i - \frac{\sigma}{2}, \dots, i + \frac{\sigma}{2}]$ denote the selection window of size $\sigma + 1$ centered about i, where σ is assumed to be an even integer. As these selection windows will be used to address our position tensor $\mathbf{P}$, we must be careful to assure they have a valid definition for each timestep i. This is achieved simply by "reflecting" the simulation data about the first and last frame to handle the respective border cases.

Finally, we shall make use of the notion of a discretely sampled Normal distribution. We denote by $N(i, \sigma)$, the normal distribution with mean i and variance σ, and by $N(i, \sigma)(j)$, the evaluation of $N(i, \sigma)$ at j. Then, we define $\mathcal{G}^k_{\sigma,i} = [N(i, \sigma)(i - \frac{k}{2}), \dots, N(i, \sigma)(i), \dots, N(i, \sigma)(i + \frac{k}{2})]$ as the set containing k values, each value obtained via the evaluation of the appropriately parametrized normal distribution at a given location. $\mathcal{G}^k_{\sigma,i}$ is symmetric about its center, which is at the $\frac{k}{2}^{\text{th}}$ entry, $\mathcal{G}^k_{\sigma,i}[\frac{k}{2}]$.

3.1.1 Per-atom Saliency

In practice, we define our saliency field for each atom and for each timestep over a number of scales. To simplify the exposition, we will first consider only a single scale σ. Let us consider computing the saliency of atom a at timestep t; it is given by the following equation:

$$\mathcal{S}_\sigma[a, t] = \left|\left| \left(\sum_{i \in w(t, 2\sigma)} (\mathcal{G}^{2\sigma}_{\sigma,t}[j] * \mathbf{P}_{a,i}) \right) - \left(\sum_{i \in w(t, 2\sigma)} (\mathcal{G}^{2\sigma}_{2\sigma,t}[j] * \mathbf{P}_{a,i}) \right) \right|\right| \tag{1}$$

Where $j = i - (t - \sigma)$ is used as the local index into the discretely sampled Normal distributions. Essentially, we consider the Gaussian weighted average of the atom's position at scales σ and 2σ, and define the saliency as the norm of the difference. In this formulation, we consider σ as the "center" scale and 2σ as the "surround" scale. The greater the difference between the center and the surround, the higher the saliency at the point of evaluation.

3.1.2 Per-timestep Saliency

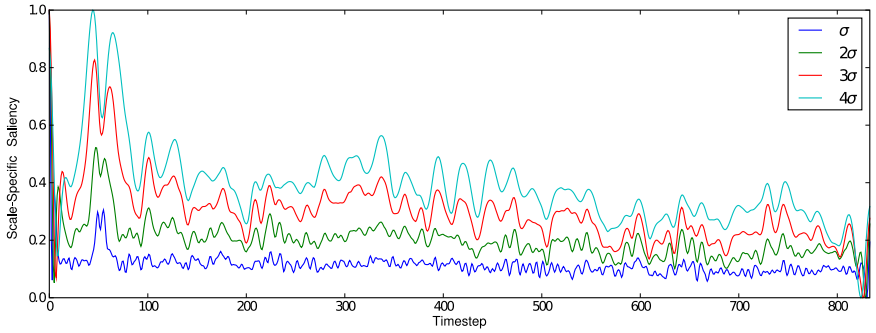
Once we have computed the per-atom saliency for each atom and timestep, we can aggregate information to obtain a per-timestep measure of saliency. We define the saliency of timestep t at scale σ as follows:

$$\mathcal{S}_\sigma[t] = \frac{1}{m} \sum_{a=1}^{m} \mathcal{S}_\sigma[a, t] \tag{2}$$

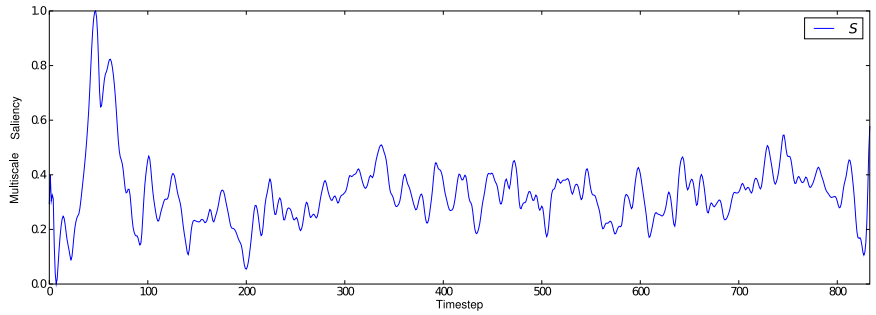
The saliency for timestep t is given simply as the mean saliency of all m constituent atoms at timestep t.

3.1.3 Multiscale Saliency

The definition of saliency we have so far provided exists only at a single scale, σ. Formulating multiscale saliency is as simple as considering multiple values of σ and composing the results.



(a) Saliency Function at Different Scales



(b) Multiscale Saliency Function

Figure 2 Each scale considered produces a different saliency function, as seen in (a). These saliency functions are combined into a single multiscale saliency function $\mathcal{S}$, as shown in (b).

The multiscale saliency measure for a given timestep t is defined over a set of scales as follows:

$$\mathcal{S}[t] = \frac{1}{|\Lambda|} \sum_{\sigma \in \Lambda} \bar{\mathcal{S}}_\sigma[t] \tag{3}$$

Where $\bar{\mathcal{S}}_\sigma$ is the normalized saliency function at scale σ and $\Lambda = \{\sigma_1, \sigma_2, \ldots\}$ is the set of scales from which the multiscale saliency is composed. Throughout this work, we will simply consider $\Lambda = \{\sigma, 2\sigma, 3\sigma, 4\sigma\}$, though many different choices are possible. The composition of the different single scale saliency functions into the single multiscale saliency function in illustrated in figure 2.

4 Using Saliency

4.1 Interpreting The Saliency Function

Having obtained the multiscale saliency function $\mathcal{S}$, we may now inquire about its intuitive relationship to the original simulation data. What, precisely, does $\mathcal{S}$ measure? We shall answer this question by again appealing to the criteria by which we defined our saliency function. The computation of $\mathcal{S}$ relies on a center-surround mechanism; effectively determining the uniqueness of a particular timestep with respect to its temporal neighborhood. Furthermore, the computation of $\mathcal{S}$ is multiscale; suggesting that it encodes this uniqueness over temporal neighborhoods of various size.

Thus, we suggest that $\mathcal{S}[t]$ provides a measure of the uniqueness of timestep t with respect to neighboring timesteps over multiple scales. Local extrema of $\mathcal{S}$ may now be interpreted to tell us something about the timesteps to which they correspond. In particular, local maxima of $\mathcal{S}$ correspond to timesteps which are very different, or *anomalous*, with respect to their surroundings. Similarly, local minima of $\mathcal{S}$ correspond to timesteps which are very similar to, or *representative* of, their surroundings. Equipped with this intuitive interpretation of $\mathcal{S}$, we can now suggest how the saliency function can be used, and how this intuition can be verified.

4.2 Representative Keyframe Selection

As mentioned in section 2, the changes in atomic position between consecutive timesteps of a simulation are dominated by Brownian motion. Only over larger timescales do interesting and purposeful molecular conformational changes occur. This suggest that one might be able to summarize the content of the simulation very precisely by choosing only a few timesteps (keyframes) to represent the entire simulation. Such keyframes can be useful for summarization, indexing, and numerous other tasks.

When finding such keyframes, there is a particular constraint that we wish to respect. Namely, *keyframes should be drawn from the actual simulation data.* Though it might be tempting to produce representative molecular conformations by aggregating information from multiple timesteps of simulation data, there are compelling reasons to avoid this approach. Molecular dynamics simulations are highly computationally intensive and modeled upon equations which respect numerous physical constraints (e.g. atom positions may not superpose, and a molecular configuration should be consistent with the potential field induced by the molecule's constituent atoms). Data representations which aggregate data and synthesize a representation may easily violate such constraints which are painstakingly considered during the course of the simulation. In order to avoid such problems, we shall not consider synthesizing representative keyframes. Rather, the representative keyframes we select will be taken directly from the original simulation data; thus ensuring that they are valid and consistent with the underlying physical model of the simulation.

Respecting this constraint, and understanding its motivation, we may now consider some different approaches to keyframe selection. In particular, we will suggest three methods for keyframe selection.

4.2.1 Random Keyframe Selection

Perhaps the simplest approach to obtaining keyframes is to sample uniformly at random from the full simulation. We shall label this approach as **RS**. Since differences in molecular configuration between nearby timesteps are dominated by Brownian motion, the selection of timesteps which are well distributed and reasonably temporally separated will likely provide a meaningful summary of the simulation. This approach has no dependence on the underlying data, and may yield an arbitrarily poor set of representative keyframes. However, its implementation is completely trivial and exceedingly fast.

4.2.2 Douglas-Peucker Keyframe Selection

The second method we shall consider for keyframe selection is an extension of the classical Douglas-Peucker (**DP**) algorithm [4]. Classically, this algorithm has been used to approximate planar curves by polylines. The algorithm itself is fairly simple and has an elegant recursive

definition. Consider two points, p_1 and p_2 residing on the planar curve $\mathcal{C}$. Consider the line segment $\overline{p_1 p_2}$ as the linear approximation of $\mathcal{C}$ between these two points. The Douglas-Peucker algorithm considers the orthogonal projection of $\mathcal{C}$ onto $\overline{p_1 p_2}$ at n uniform discrete sample positions between p_1 and p_2. Let $c^* \in \mathcal{C}$ denote the point for which this orthogonal projection is farthest from the true curve. The **DP** algorithm will add c^* to the set of approximation samples and then recursively descend onto the line segments $\overline{p_1 c^*}$ and $\overline{c^* p_2}$. Various criteria may be established for the termination of the algorithm; for example, one may terminate the algorithm when a desired number of approximation samples has been chosen, or when a maximum acceptable per-sample error threshold has been achieved.

The extension of the Douglas-Peucker algorithm to molecular dynamics simulation data is fairly straightforward. Instead of a planar curve, one attempts to approximate atom positions. Linear interpolation of atom positions is considered between consecutive keyframes, where the temporal distance between these keyframes is used as the linear interpolant. Thus, one may obtain a linear prediction $\vec{\tilde{p}}_{a,t}$ for each atom (a) at each timestep (t) of the true simulation data $\vec{p}_{a,t}$. The difference between the linear prediction of a timestep t and the true simulation data at t can simply be computed as the sum of differences between predicted and actual atom positions as in equation 4:

$$\epsilon_t = \sum_{a=1}^{m} \left|\left| \vec{\tilde{p}}_{a,t} - \vec{p}_{a,t} \right|\right| \tag{4}$$

In this adaptation of the **DP** algorithm, ϵ_t simply replaces the orthogonal distance between the curve and the approximating line segment; leaving the remainder of the algorithm largely the same.

4.2.3 Saliency Guided Keyframe Selection

Finally, we may consider using the multiscale saliency function $\mathcal{S}$ to guide the selection of keyframes. We shall refer to this approach as saliency guided (**SG**) keyframe selection. Though the intuition behind this method is simple, some care must be taken when actually using $\mathcal{S}$ for keyframe selection. First we must determine precisely how $\mathcal{S}$ should guide the selection of keyframes. In section 4.1, we suggested that local minima of $\mathcal{S}$ correspond to representative timesteps from the simulation. Therefore, it makes sense to choose the timesteps corresponding to the local minima of $\mathcal{S}$ as keyframes. First, however, we must more carefully define what we mean by *local* minima. Since $\mathcal{S}$ is obtained from the composition of saliency functions spanning multiple different scales, it is possible that the fine scale saliency functions will lead to small fluctuations between consecutive values of $\mathcal{S}$. However, unless the surrounding values of $\mathcal{S}$ are relatively smooth, such small fluctuations should not trigger keyframe selection.

To address this issue, we make use of non-maximal suppression. Given a window size, w, non-maximal suppression will suppress the entire signal with the exception of the w-local maxima. For example, when we consider $\mathcal{S}[t]$ for some timestep t, non-maximal suppression will suppress $\mathcal{S}[t]$ to 0 unless $\mathcal{S}[t] > \mathcal{S}[t+i], i \in [-\frac{w}{2}, \frac{w}{2}]$. A reasonable value of w will ensure that not too many keyframes are selected, and that small and non-meaningful fluctuations in $\mathcal{S}$ do not trigger keyframe selection. Here it is important to note that we consider non-maximal suppression of the function $1 - \mathcal{S}$, which is equivalent to a non-minimal suppression of $\mathcal{S}$ itself.

However, the use of non-maximal suppression introduces a new consideration. How should one choose a *reasonable* value of the window size w? Recall that w should be related to the

scale in $\mathcal{S}$ at which we wish to detect keyframes. Rather than choosing a fixed value of w, we introduce a procedure for multiscale keyframe selection (**MSKS**). The user provides the **MSKS** procedure with a target number of keyframes, k, and the procedure will return the smallest number $k' \geq k$ of keyframes obtained using a coarse to fine selection procedure. The **MSKS** is possible in part due to the *subset containment* property of non-maximal suppression. Consider performing non-maximal suppression on $1 - \mathcal{S}$ using a window of size w. We shall denote by $NMS(f, w)$ the non-maximal suppression of the function f using a window of size w. Further, let $NZ(f)$ denote the indices for which the discrete function f takes non-zero values. Then we may denote the set of desired keyframes (w-local maxima) as $F_w = NZ(NMS(1-\mathcal{S}, w))$. Now, consider a window of size $w' < w$. The subset containment property ensures that $F_w \subseteq F_{w'}$. This means that if $1 - \mathcal{S}[t]$ is a w-local maximum, then it is also a w'-local maximum $\forall w' < w$. Bearing this property in mind, we will now describe the **MSKS** procedure.

Algorithm 1: MSKS.

```
Input: S, k
Output: F
F = ∅;
w = |S|;
f = 1 − S;
while |F| < k do
    w = w/2;
    f' = NMS(f, w);
    for i = 0 ; i < |f'| ; i = i + 1 do
        if f'[i] > 0 then
            F = F ∪ i;
return F;
```

The **MSKS** procedure iteratively builds up a set of keyframes by running non-maximal suppression on $1 - \mathcal{S}$ with windows of decreasing width. The pseudo-code for this algorithm is given in algorithm 1. One important aspect of this algorithm is that when the user requests k keyframes, the algorithm will actually return $k' \geq k$ keyframes. This is due to a natural notion of scale, both in the saliency function $\mathcal{S}$ and in the multiscale keyframe selection procedure. For each window size used during the non-maximal suppression procedure, there are a number of natural keyframes which will be selected at the corresponding scale. This feature is important, because it allows the algorithm to select representations of the underlying simulation at varying levels of coarseness.

Figure 3 shows a plot of the multiscale saliency function $\mathcal{S}$, where 12 keyframes have been chosen using the **MSKS** procedure. For each keyframe that has been selected, a circle has been plotted at the corresponding local minima of $\mathcal{S}$. The color of each circle denotes the scale (window size = w), at which the timestep corresponding to that circle was first selected as a keyframe.

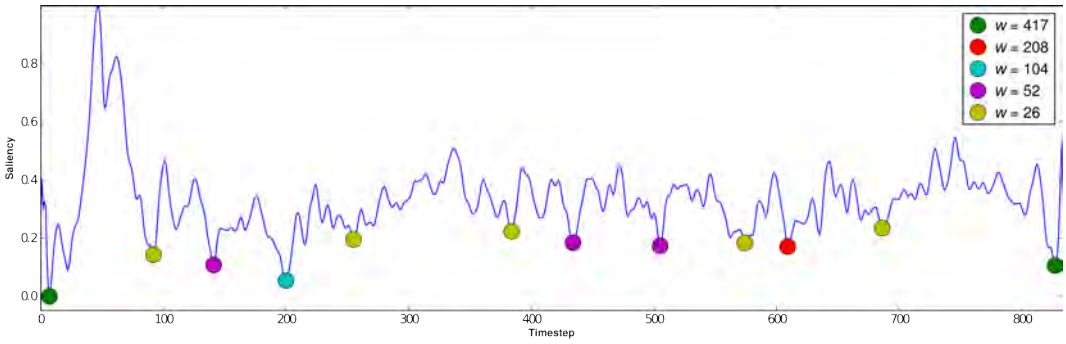


Figure 3 This figure illustrates the selection of keyframes from the multiscale saliency function, $\mathcal{S}$, using the **MSKS** procedure. For each timestep chosen as a keyframe, the corresponding local minima is marked with a circle. The color of each circle denotes the non-maximal suppression window size, w, at which the corresponding timestep was first chosen as a keyframe.

5 Results

5.1 Datasets

GroEL Transitions

GroEL is a molecular nanomachine that changes its conformation empowered by ATP chemistry. Spectacular conformational changes between the conformational states of GroEL , such as $T \rightarrow R$ and $R \rightarrow R''$ transitions, are induced upon ATP binding and hydrolysis at the catalytic site of equatorial domain, respectively. Normal operation of the GroEL -GroES chaperonin system [13] is of utmost importance for the cell function as this increases the yield of substrate proteins that are prone to misfold. The misfolding of proteins and subsequent aggregations often lead to the fatal neurodegenerative disorders such as Alzheimers and prion diseases. To better understand the allosteric transitions of GroEL molecule at the microscopic level, multiple sets of Brownian dynamics simulation were performed using a self-organized polymer (SOP) model [7, 5]. The SOP model adopts a strategy of using the minimal representation of proteins and RNA that retains the topological information. The simulations [6] show that $T \rightarrow R$ transition of GroEL , in which the apical domains undergo counterclockwise motion, is mediated by a multiple salt-bridge switch mechanism at the interfaces of seven subunits. The initial event in the $R \rightarrow R''$ transition, during which GroEL rotates clockwise, involves a dramatic outside-in concerted movement of helices K and L, exerting a substantial strain on he GroEL structure, induces the 90 degree clockwise rotation and 40 degree upward movement of apical domain. This simulation consists of 3668 atoms and 834 timesteps.

Folding of Tetrahymena Ribozyme

Large RNAs fold into complex structures which determine their biological activities. The RNA folding problem studies how RNA folds into a unique structure without searching through all possible conformation. How macromolecules from thermophilic organisms achieve thermostability has been a fascinating question for structural biologist and the biotechnology industry. There are many pathways in the folding procedure. Some pathways lead directly to the native state, while others result in “kinetically trapped” conformations that contain some native, as well as non-native interactions. This dataset presents the folding simulation of the

Tetrahymena Ribozyme. Force-quench refolding of the P4-P6 subdomain of the Tetrahymena ribozyme occurs through a compact intermediate. Subsequent formation of tertiary contacts between helices P5b-P6a and P5a/P5c-P4 leads to the native state. This simulation consists of 158 atoms and 1540 timesteps.

5.2 Verification Procedure

It is difficult to visually verify the representative power of the chosen keyframes. Thus, we will rely on a purely quantitative verification procedure. Representative keyframes should be able to predict their temporal neighborhood well. To measure this predictive power, we shall make use of the keyframes chosen by the various selection algorithms (i.e. **RS**,**DP**, and **SG**) and perform an interpolation procedure over them. The total error between the simulation data predicted solely by the keyframes and the actual simulation data will be used as a measure of the predictive/representative power of the selected keyframes. More precisely, for every timestep t which is not, itself, a keyframe, the interpolation procedure will yield a prediction with some resultant error ϵ_t. The representative power of the selected keyframes will be measured by $\epsilon = \sum_{t=1}^{n} \epsilon_t$, where a smaller ϵ is indicative of more representative keyframes. $\epsilon_{\mathbf{RS}}$, $\epsilon_{\mathbf{DP}}$, and $\epsilon_{\mathbf{SG}}$ are used to indicate the errors using the three methods.

5.3 Experimental Results

The saliency guided keyframe selection approach (**SG**), in paper is compared against the Douglas-Peucker(**DP**) and random selection methods (**RS**); all of which are described in section 4.2. The experiments show the keyframes selected by the saliency approach better approximates the simulations than the other two methods. For each simulation, we consider representing the simulation using three different numbers of key frames, selected using the **SG**, **DP**, and **RS** methods. The random sampling method illustrates the difficulty of this problem and establishes a baseline performance for comparison, and the **DP** algorithm is considered as the current state of the art. Random selection of keyframes are sampled 1000 times for each experiment. The reported performance of the random sampling method is an average across all the samples. The statistics intend to show the error of approximations produced by the two keyframes selection methods (**DP** and **SG**) are lower than random selection, and they are statically significant. The performance difference in between the average random selection and the other two method shows it is unlikely to randomly pick good keyframes to approximate the simulations. This shows that not all timesteps are equally representative, and that it is important to select the correct timesteps when summarizing these complex molecular dynamics simulations.

Table 1 and Table 2 show statistics of the relative improvement between **SG** and **RS**. The relative improvement is computed by:

$$\frac{\epsilon_{\mathbf{RS}} - \epsilon_{\mathbf{SG}}}{\epsilon_{\mathbf{RS}}}$$

Most of the results show the **SG** method performance is 10% - 20% better than the random selection performance.

The null hypothesis here is that the distribution of error due to **SG** is drawn from the same distribution of **RS**. The t-values shown in Table 1 and Table 2 shows this hypothesis can be safely rejected (95% confidence interval).

The relative cumulative approximation error of the experiments with respect to the **RS** method are shown in Figure 4 and Figure 5. The horizontal axis represents the timesteps and

Table 1 Relative improvement statistics of saliency guided (**SG**) frame selection over random selection (**RS**) for the GroEL simulation.

# keyframes	Relative improvement	stdev	t value
2	+1.49%	19.46%	2.58
5	+13.63%	15.63%	28
10	+18.0%	8.97%	63

Table 2 Relative improvement statistics of saliency guided (**SG**) frame selection over random selection (**RS**) for the Tetrahymena Ribozyme simulation.

# keyframes	Relative improvement	stdev	t value
15	+22.58%	9.51%	75
24	+22.96%	6.97%	94
45	+15.44%	4.2%	113

Table 3 Relative improvement statistics of saliency guided selection (**SG**) over Douglas-Peucker selection (**DP**).

(a) GroEL

# keyframes	Relative improvement
2	+22.81%
5	+9.25%
10	+6.61%

(b) Tetrahymena Ribozyme

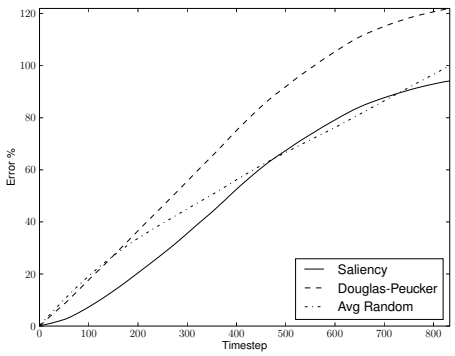
# keyframes	Relative improvement
15	+12.13%
24	+12.12%
42	+8.99%

the vertical axis represents the relative cumulative error. The three plotted lines represent the errors obtained by the **SG**, **DP**, and **RS** methods. Each line shows the cumulative error relative to the **RS** method at a certain timestep throughout the simulations.

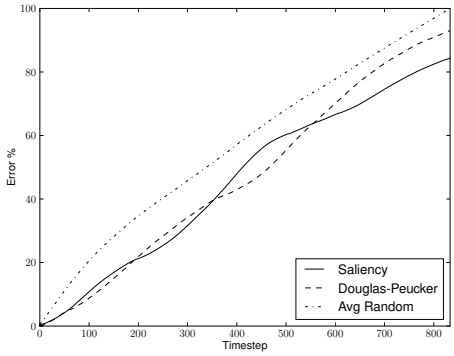
In the GroEL simulation, the **SG** method consistently results in the lowest overall error; whereas the 5 or 10 keyframes selected by the **DP** algorithm approximate the simulation better than random selection. The approximations produced by the **SG** method are $6\% - 22\%$ better than those produced by the **DP** algorithm. In the 2 keyframe experiment, The **DP** algorithm selects only the beginning and the ending timesteps, this results in significant loss of details during the simulation and hence it was even outperformed by average random selection.

In the Tetrahymena Ribozyme simulation, the **SG** method consistently results in the lowest overall error, and the **DP** algorithm also consistently approximates the simulation better than random selection. The approximations produced by the **SG** method are 12% - 15% better than those produced by the **DP** algorithm. The trend of the error plot shows that the **SG** method outperformed the **DP** algorithm and **RS** approach at every frame across the simulation.

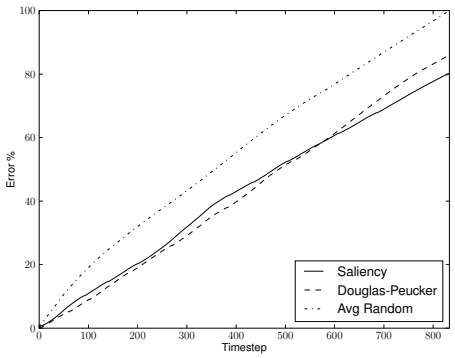
Table 3 shows a summary of relative improvement obtained by employing the **SG** frame selection method rather than the **DP** frame selection method.



(a) 2 keyframes



(b) 5 keyframes



(c) 10 keyframes

Figure 4 Relative cumulative error (with respect to **RS**) of approximating the GroEL simulation by the selected keyframes.

5.3.1 Selected Keyframes

The selected keyframes from the two simulation are presented in this section.

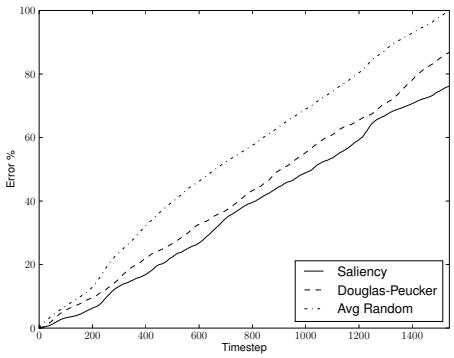
The frames from the GroEL simulation show the conformational changes of the functional subunits during the course of the $R \rightarrow R''$ transition. Figure 6 shows the 10 keyframes selected using the **SG** method for summarizing the GroEL simulation.

The frames from the Tetrahymena Ribozyme simulation show how the Ribozyme molecule folds from a straight chain in to it native state. Figure 7 shows the 10 keyframes selected via the **SG** method for summarizing the Tetrahymena Ribozyme simulation.

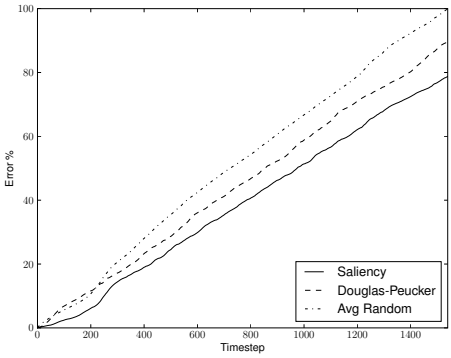
5.3.2 The Most Salient Frames

Recall that when using the **SG** method to obtain keyframes, we are interested in obtaining the most representative timesteps from the simulation data. These timesteps are chosen by using the **MSKS** procedure to find the local maxima of $1 - \mathcal{S}$ over various scales.

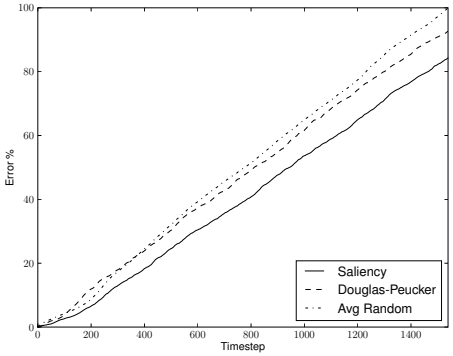
This procedure naturally leads one to wonder about the effect of using the **MSKS** procedure on $\mathcal{S}$ rather than $1 - \mathcal{S}$. While the local maxima of $1 - \mathcal{S}$ signify the timesteps which are the most similar with respect to their temporal neighborhood over various scales,



(a) 15 keyframes



(b) 24 keyframes



(c) 45 keyframes

Figure 5 Relative cumulative error (with respect to **RS**) of approximating the Tetrahymena Ribozyme simulation by the selected keyframes.

the local maxima of $\mathcal{S}$ correspond to those which are most different. Such anomalous timesteps are useful and informative in their own right. By simply inverting our multiscale saliency function, we are able to select *both the most representative and the most anomalous* timesteps of a simulation. Figure 8 and Figure 9 show the 5 most salient frames from the GroEL and Tetrahymena Ribozyme simulations.

6 Conclusion

In this work, we have introduced a notion of temporal saliency for molecular dynamics simulation. Such a notion is useful for summarizing, abstracting, indexing, previewing, and analyzing these large time-varying datasets. We have shown how our multiscale saliency function $\mathcal{S}$ can be used in conjunction with a multiscale keyframe selection procedure to choose representative frames from among the locally chaotic motion of molecular dynamics simulations. By employing interpolation over such keyframes as a measure of their predictive power, we have shown that saliency guided keyframe selection consistently chooses more representative frames than other methods.

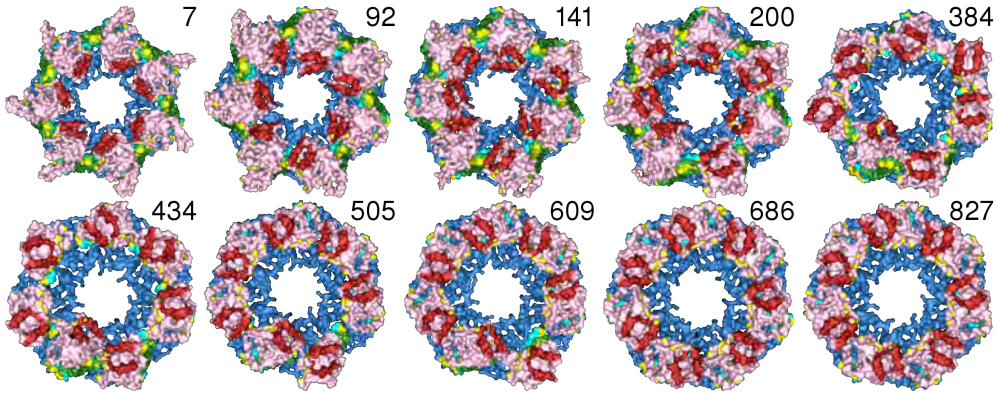


Figure 6 The 10 most representative timesteps from the simulation of GroEL nanomachine consisting of 3668 atoms and 834 timesteps. The corresponding saliency function is shown in Figure 2.

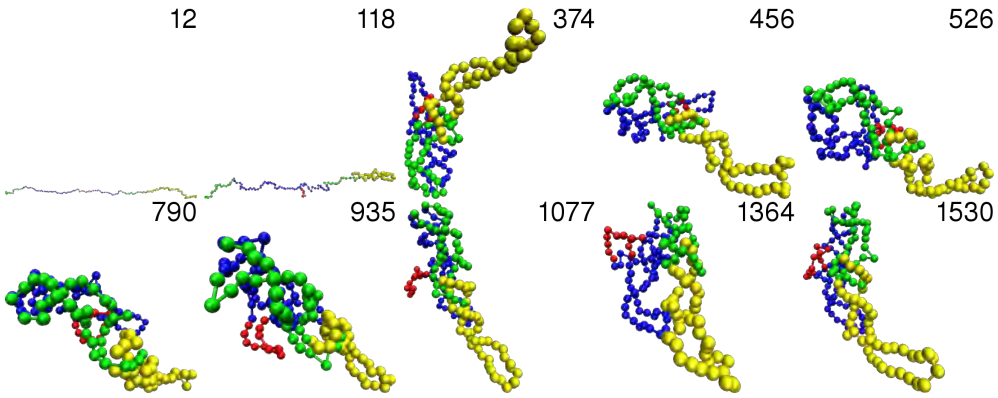


Figure 7 The 10 most representative timesteps from Tetrahymena Ribozyme simulation consisting of 158 atoms and 1540 timesteps.

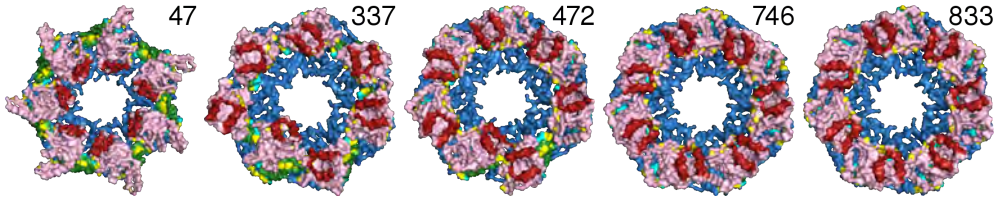


Figure 8 The 5 most salient timesteps from the 834 timesteps of the GroEL simulation with 3668 atoms.

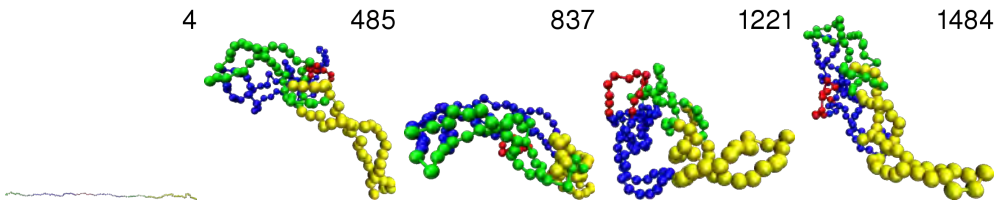


Figure 9 The 5 most salient timesteps from the 1540 timesteps of the Tetrahymena Ribozyme simulation with 158 atoms.

Acknowledgments

We gratefully acknowledge Andriy Anishkin, Sam Cho, Changbong Hyeon, Sergei Sukharev and D. Thirumalai for providing several datasets as well as expert insight into the corresponding simulations. We also want to thank Youngmin Kim and Dianne P. O'Leary for valuable discussions.

This work has been supported in part by the NSF grants: IIS 04-14699, CCF 04-29753, CNS 04-03313, CCF 05-41120, CMMI 08-35572 and the NVIDIA CUDA Center of Excellence. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the research sponsors.

References

1 Okan Arikan. Compression of motion capture databases. *ACM Transactions on Graphics*, 25(3):890 – 897, July 2006.

2 J. Assa, Y. Caspi, and D. Cohen-Or. Action synopsis: pose selection and illustration. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):667–676, 2005.

3 Daniel DeMenthon, Vikrant Kobla, and David Doermann. Video summarization by curve simplification. In *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, pages 211–218, New York, NY, USA, 1998. ACM.

4 David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

5 C. Hyeon, R. I Dima, and D. Thirumalai. Pathways and kinetic barriers in mechanical unfolding and refolding of RNA and proteins. *eprint arXiv:q-bio/0611050*, November 2006.

6 Changbong Hyeon, George H. Lorimer, and D. Thirumalai. Dynamics of allosteric transitions in groel. *Proc. Natl. Acad. Sci.*, 103:18939, 2006.

7 Changbong Hyeon and D. Thirumalai. Mechanical unfolding of rna: From hairpins to structures with internal multiloops. *Biophysical Journal*, 92:731, 2007.

8 L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. on Pattern Analysis and Machine intelligence*, 20(11):1254–1259, 1998.

9 John L. Klepeis, Kresten Lindorff-Larsen, Ron O. Dror, and David E. Shaw. Long-timescale molecular dynamics simulations of protein structure and function. *Current Opinion in Structural Biology*, 19(2):120 – 127, April 2009.

10 C. H. Lee, A. Varshney, and D. Jacobs. Mesh saliency. *ACM Transations on Graphics (Proceedings of ACM SIGGRAPH)*, 24(3):659 – 666, 2005.

11 Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Simple and efficient compression of animation sequences. In *Symposium on Computer Animation*, pages 209 – 217, 2005.

12 Deborah Silver and X. Wang. Volume tracking. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 157–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

13 Lorimer GH Thirumalai D. "chaperonin-mediated protein folding". *Annual Review of Biophysics and Biomolecular Structure*, 30:245–269, 2001.

14 Ba Tu Truong and Svetha Venkatesh. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(1):3, 2007.

Streaming Aerial Video Textures

Christopher S. Co[1], Mark A. Duchaineau[2], and Kenneth I. Joy[3]

1 **Google, Inc.**
2 **Lawrence Livermore National Laboratory, Livermore, CA, USA**
`duchaineau1@llnl.gov`
3 **Institute for Data Analysis and Visualization, University of California, Davis, USA**
`kijoy@ucdavis.edu`

Abstract

We present a streaming compression algorithm for huge time-varying aerial imagery. New airborne optical sensors are capable of collecting billion-pixel images at multiple frames per second. These images must be transmitted through a low-bandwidth pipe requiring aggressive compression techniques. We achieve such compression by treating foreground portions of the imagery separately from background portions. Foreground information consists of moving objects, which form a tiny fraction of the total pixels. Background areas are compressed effectively over time using streaming wavelet analysis to compute a compact video texture map that represents several frames of raw input images. This map can be rendered efficiently using an algorithm amenable to GPU implementation. The core algorithmic contributions of this work are methods for fast, low-memory streaming wavelet compression and efficient display of wavelet video textures resulting from such compression.

1998 ACM Subject Classification J.3 Life and Medical Sciences

Keywords and phrases Molecuar Dynamics, Saliency, Simulation

Digital Object Identifier 10.4230/DFU.SciViz.2010.336

1 Introduction

Aerial systems are being devised that deploy billion-pixel cameras, providing high-resolution wide-area video at several frames per second. These new cameras produce data streams that are a factor of one hundred larger than previously deployed in aerial imaging systems. In this setting, the main challenges are twofold:

- to transmit this huge pixel stream to the ground over available wireless bandwidths, at best about 20 megabits per second; and
- to display this huge image stream visualized over 3D terrain models, by extending the best known view-dependent display optimization techniques to handle data that is not only large spatially, but large temporally.

Current state-of-the art static image and video compression methods are at best capable of a factor of 100 compression while keeping high image quality to perform automated mover detection and analysis. An additional factor of 10 to 100 compression is needed. At the same time, the process of compression must occur before transmission to ground due to restricted bandwidth, payload storage, and power available on the aircraft.

To achieve the thousand to ten thousand times compression needed, alternatives to conventional image and video compression strategies are needed. We make two observations about wide-area aerial video that enable some specialization in compression techniques:

1. images are taken repeatedly over the same area, and
2. the primary interest is for moving objects.

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 336–345

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

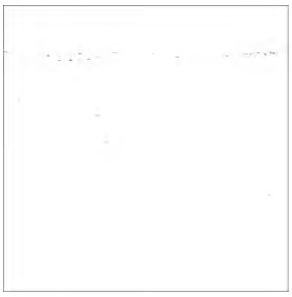
(a)  (b)  (c) 

Figure 1 For aerial images, such as the one shown in (a), we achieve high levels of compression by considering two portions of the images. The foreground portion shown in (b) is comprised of moving objects in the image and makes up only a tiny fraction of the image. The background portion shown in (c) varies slowly and smoothly over time, making it ideal for aggressive compression.

With two assumptions, the strategy pursued here is to compress foreground and background imagery separately, see Figure 1. Moving objects comprising the foreground make up a tiny fraction of the pixels in the image and can be transmitted directly. Background imagery varies slowly and smoothly, making it ideal for compression.

The background imagery is compressed in a streaming fashion with the aid of 1D wavelet analysis performed on a per-pixel basis. Using temporal wavelet analysis, the resulting *wavelet images* effectively represent several frames of raw input images. Since movers only occupy a tiny fraction of the overall imagery, and background imagery is reduced significantly to low temporal rates, and the overall compressed size can easily be a factor of 10 to 100 smaller than using existing state-of-the-art still-image and video compression on the image stream, while meeting the special accuracy requirements for analysis.

To minimize memory usage when compressing a large time sequence of huge images, streaming variants on wavelet analysis are used. The key idea is to view wavelet lifting diagrams as dependency graphs, and to effectively "parse" the diagrams as soon as images are taken. Overall this reduces the total memory footprint.

The system consists of several components—image segmentation, temporal compression, and rendering. In this paper, we focus on the description of an image segmentation approach where methods are derived from similar techniques used in the visualization and computer graphics community. Section 4 describes the image segmentation methods. The wavelet-based streaming compression algorithm is discussed in Sections 5, 6 and 7. Section 8 addresses the replacement of foreground movers by static elements. Section 9 illustrates the efficient rendering of the compressed information using a GPU-based algorithm. Sections 10 and 11 illustrates results obtained with this system.

2 Related Work

There is an extensive volume of work on segmenting foreground from background in computer vision [2]. Algorithms for object recognition and video tracking are most related to our work. In particular, the goal of video tracking is to locate moving objects in time. While the literature has primarily focused on improving the speed and accuracy of locating a few moving objects, relatively little research has been devoted to locating a large number of moving objects efficiently. Consider that aerial image sequences may contain hundreds of moving vehicles.

Wavelets have been used in computer graphics and visualization for a number of applications, including multiresolution analysis, variational modeling, and compression [7]. B-spline wavelets

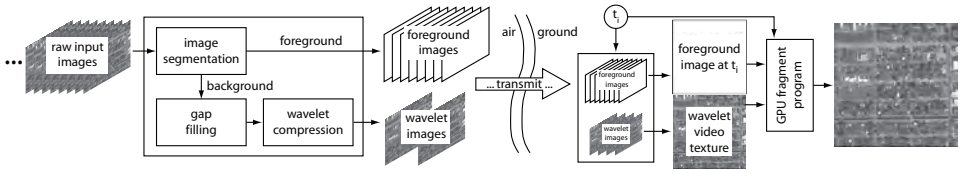


Figure 2 Overview of the complete compression system. A raw input stream of images are segmented into foreground and background images. Moving objects comprising the foreground make up a tiny fraction of the pixels in the image and are transmitted directly. Background images are compressed aggressively using wavelet-based techniques, greatly reducing the number of frames that need to be transmitted. After transmission, wavelet images are "decompressed" efficiently in the renderer for a user-specified time point t_i, and foreground pixels are overlayed on top.

have been combined with subdivision surface techniques to represent geometric models at multiple levels-of-detail [1, 7]. They compute wavelet transforms using a lifting approach [8] that divide wavelet analysis into smaller, more efficient lifting operations.

Streaming techniques treat the data as a continuous stream of information. These algorithms operate on restricted contiguous portions of data which are currently "in focus." Because streaming methods operate on a fixed amount of data at a time, they are often faster, more efficient, and more scalable than "global" techniques that require an entire data set to reside in main memory [5]. Recent research has focused on streaming algorithms for simplifying and compressing 3D geometric models [5]. The availability of online media has increased the need for streaming video encoding and decoding methods [6]. Several codecs for streaming video exist, however many of these methods focus on only streaming content delivery. Compression and encoding of the content is often considered a preprocess.

In our work, we combine image segmentation and wavelet methods to compress long sequences of huge aerial imagery in a streaming fashion. We analyze pixel intensity values over time to separate pixels that represent moving foreground objects from stationary background pixels. Background pixels form a slowly varying image sequence that we compress with the aid of wavelet analysis. We use streaming evaluation of the wavelet transformation to allow both compression and delivery to occur as soon as possible in a streaming fashion while keeping the memory footprint low. Wavelet encoded images compactly represent large numbers of images. They can be rendered efficiently making them suitable for incorporation into state-of-the-art large terrain, large texture viewers, such as the one developed by Hwa et al. [3, 4].

3 System Overview

Given a raw input stream of huge images, our system outputs foreground images and wavelet-compressed background images. In the first stage of processing, raw input images are segmented into foreground and background images by analyzing changes in pixel intensity over time. Holes in the background images are filled in preparation for wavelet compression. The background images are passed to a wavelet compression engine comprised of several levels of wavelet transform. We refer to these output background images as *wavelet images*. The number of wavelet images output is a constant factor smaller than the number of images input to the system. Wavelet images and foreground images constitute a compressed set of information that is suitable for transmission. After transmission, consecutive sets of wavelet images are used to render larger video sequences using a texture-mapping approach. Foreground images are overlayed on top of the reconstituted backgrounds. We refer to the sets of wavelet images as *wavelet video textures*. Figure 2 illustrates our complete system.

4 Image Segmentation

Segmentation of moving objects is performed in two phases, detection and completion. In the detection phase, the goal is to have at least a single pixel of positive detection per moving object, not including paralax motion of buildings, trees, or other tall structures. In the completion phase, pixels for the entire object are determined. Extra "guard pixels" surrounding the object are also labeled as part of the mover to obtain a conservative segmentation. Both phases of this algorithm could be accomplished with well known but expensive search and correlation strategies. For real-time processing on small sensor platforms, fast, local computations are desired.

For detection, two approaches were tested. The first approach uses two buffered frames to perform detection. Pixels are declared likely movers in frame f if (1) their value $v_f(x,y)$ is different from all the values within one pixel in frame $f+1$, (2) this frame-to-frame difference has a high gradient magnitude, (3) the gradient magnitude at frame f is high, and (4) the gradient directions in the two cases above are parallel to one another (either the same or exact opposite directions). This is formulated by defining a unit gradient vector

$$\vec{g} = \frac{\nabla v_f(x,y)}{||\nabla v_f(x,y)||}$$

and defining

$$d_f(x,y) = \min_{dx,dy \in [-1,1]} |v_f(x,y) - v_{f+1}(x+dx, y+dy)|$$

along with

$$\vec{h} = \frac{\nabla d_f(x,y)}{||\nabla d_f(x,y)||}$$

and

$$q = (\vec{h'} \cdot \vec{g'})(||\vec{g}|| - g_{m\min})||\vec{h}|| \; .$$

This consistently finds some pixels per mover, but is somewhat sensitive to noise, and does not completely eliminate paralax-induced detections of tall building edges.

The second detection approach uses five frames. The median value is obtained for a pixel over these frames. The detection is then the difference between the current frame's value and the least different value from a small neighborhood of the next frame. A small amount of "soft erosion" is performed on the results. This was found to be very reliable at detecting some pixels per mover, and was not as sensitive to noise or to building edge paralax.

After detecting a set of pixels from each mover, a conservative region of pixels is determined around each complete mover. Using the results of the five-frame detection, there are good starting points to seed a search for all mover pixels. We start with the core pixel of each connected component, defined as the last pixel to be deleted for the component by repeated erosion operations that are restricted to not break components into two pieces. From this starting point, we flood fill to all pixels that are near pixels with frame-to-frame differences with magnitudes above a specified threshold.

5 Wavelet Analysis Reviewed

The lifting approach [8] for discrete wavelet analysis uses a 1D input signal F represented by a set of samples f_i uniformly spaced in time at points t_i. F is decomposed into *scaling-function coefficients*, or *s-values*, and *wavelet coefficients*, or *w-values*. In the lifting approach, this decomposition is computed by relabeling signal values as

$$s_i = f_{2i} \qquad \text{and} \qquad w_i = f_{2i+1}$$

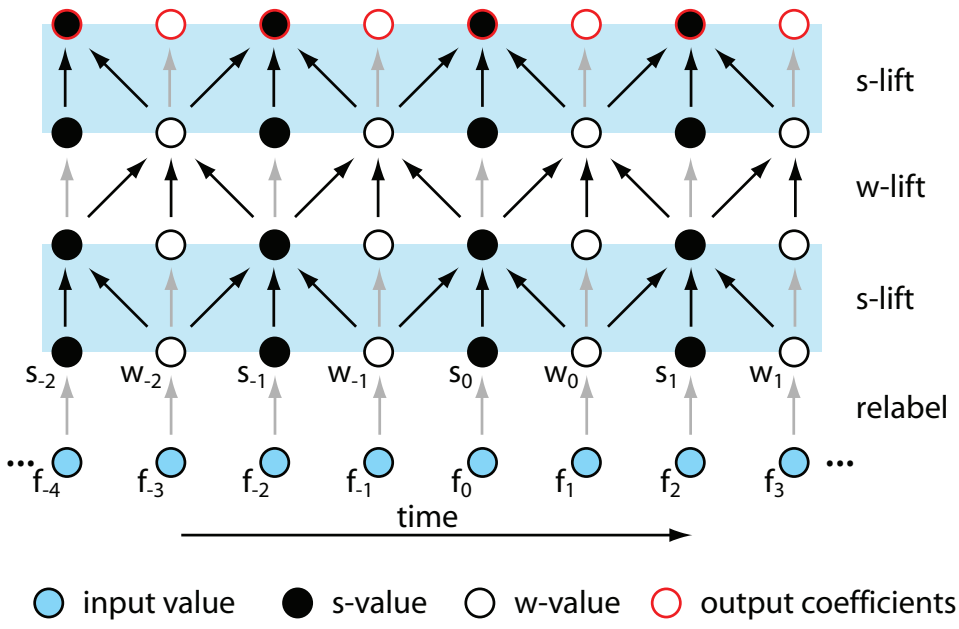


Figure 3 Illustration of cubic wavelet analysis. Input values shown as blue circles are relabeled as s-values and w-values. Three alternating s- and w-lift operations compute the final cubic wavelet coefficients outlined in red. Black arrows show which values contribute to intermediate values between lifting operations. Gray arrows illustrate which values remain the same between operations.

followed by a sequence of alternating *s-lift* and *w-lift* operations. The s-lift and w-lift operations are defined as
s-lift(a, b):

$$s_j = aw_{j-1} + bs_j + aw_j \qquad \forall j \qquad \text{and} \tag{1}$$

w-lift(a, b):

$$w_j = as_j + bw_j + as_{j+1} \qquad \forall j. \tag{2}$$

Therefore, *s*-lifts compute new s-values by computing a weighted sum of a given s-value with its neighboring w-values. Likewise, *w*-lifts compute new w-values by computing a weighted sum of a given w-value with its neighboring s-values. The order and number of lifting operations and the values of the weights a and b determine the type of wavelet used. In our work, we use uniform cubic B-spline wavelets [1, 7]. This transform is defined by the sequence of three lifting operations *s*-lift(-1/4, 1), followed by *w*-lift(-1, 1), and then *s*-lift(3/8, 2). Figure 3 illustrates the lifting approach to wavelet analysis.

6 Streaming Wavelet Analysis

Values resulting from wavelet analysis can be computed in a streaming fashion. We view the stages of the lifting operations as a dependency graph, where nodes represent computed values and directed edges represent how terms are combined to produce a computed value. From this perspective, the relationship between input signal values and output values reveals that the first s- and w-values can be computed after the first four input signal values have been received, see Figure 4a. Further

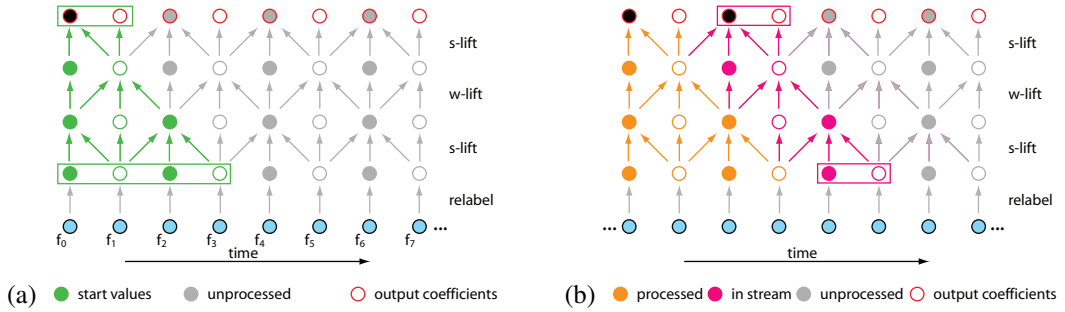


Figure 4 Illustration of streaming cubic wavelet analysis. The beginning of the computation shown in (a) requires four initial values shown here in green. These values produce one s-value and one w-value. Post-initialization computations are shown in (b). Here, pairs of input values produce pairs of output values. The pink circles represent stream values currently being computed, and orange circles represent those computed from the last stream update. Note that the new output values depend on three values from the previous computation, and so three values must be maintained in memory across stream updates. This dependence is visualized here as pink arrows emanating from orange circles. Gray circles represent values that have not yet been processed.

investigation reveals that subsequent pairs of s- and w-values can be computed as soon as pairs of input signal values are received, see Figure 4b. Only three internal results from previous computations need to be maintained between updates to compute two new output values. Thus, values from wavelet analysis can be produced in a streaming fashion, using a small memory footprint to track the computation state. For the purposes of this discussion, we call the computational engine that performs streaming wavelet analysis a *stream analyzer*.

7 Wavelet Compression

Compression is achieved by discarding w-values and performing several levels of wavelet analysis. Discarding w-values achieves a factor of two reduction in data size. The s-values from one level of analysis are used as input to the next level. Since each level of wavelet analysis achieves a factor of two reduction in data size, k levels of such nested wavelet analysis achieve a factor of 2^k reduction in data size. We achieve streaming wavelet compression by cascading k stream analyzers together such that s-value output from one stream analyzer is used as input to the next stream analyzer. Figure 5 illustrates a sample wavelet compression system constructed from three stream analyzers.

8 Filling Gaps

This wavelet analysis requires a uniform spacing of samples over time. Image segmentation removes foreground pixels, creating gaps in the stream passed to the wavelet compressor, and we must fill these temporal gaps in order to perform wavelet compression.

We adopt a simple strategy to fill in the gaps created by foreground pixels. The buffer keeps track of the last value received as well as the point in time it was received. Let f_{last} be the last pixel value received and t_{last} be the point in time it was received. Let f_{curr} be the newly arrived pixel value and t_{curr} be the current point in time. A gap exists if $t_{curr} - t_{last} > 1$. Missing values corresponding to time values $t_{last+1}, \ldots, t_{curr-1}$ are computed using linear interpolation between f_{last} and f_{curr}. We note that the goal here is to provide noise-free input to the wavelet compressor, and so linear interpolation suffices to maintain a stable input signal and is more desirable than filling gaps with zeroes. Figure 6 illustrates the gap filling process.

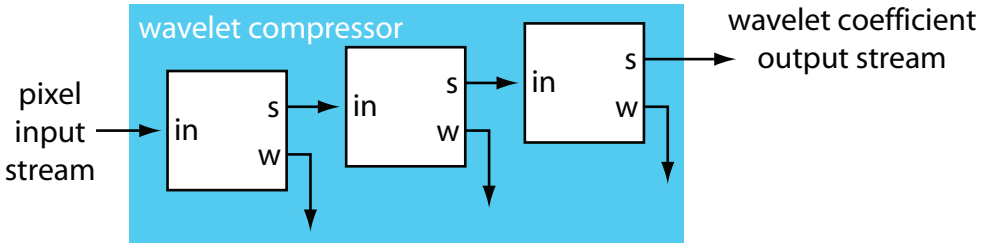


Figure 5 Illustration of a streaming wavelet compression system. A pixel value stream is input to the system, which computes a smaller output stream of wavelet coefficients. The system is constructed from three stream analyzers. Here, each box represents a stream analyzer that performs wavelet analysis, producing s- and w-values. The s-values are used as input to subsequent stream analyzers, and w-values are disregarded. Each level of wavelet analysis reduces the stream by a factor of two. Thus, this example system prduces an output stream that is a factor of $2^3 = 8$ smaller than the input stream.

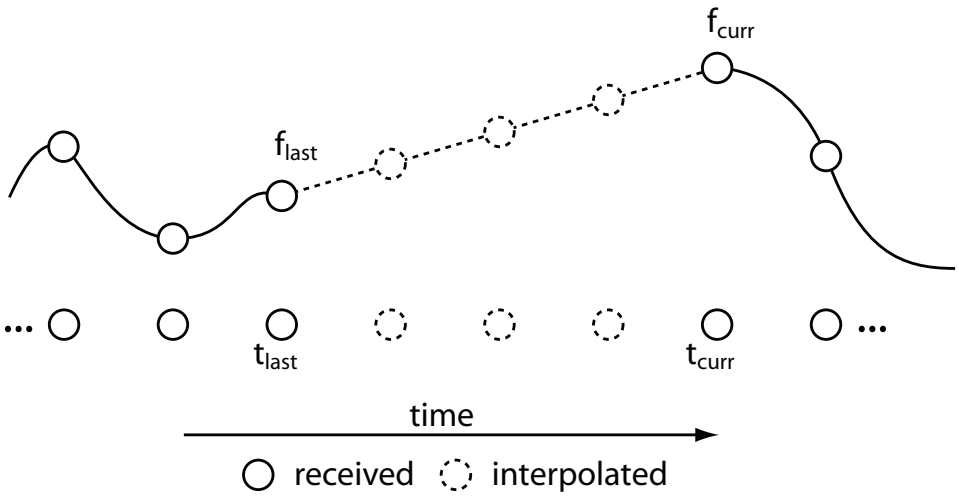


Figure 6 Illustration of gap filling. Gaps are detected easily by comparing the arrival time of the current pixel value against the arrival time of the last pixel value. Missing values are linearly interpolated using the current and last pixel values. Solid circles represent received values and dashed circles represent interpolated values.

9 Rendering Compressed Video Images

Our system performs gap filling and wavelet compression for each pixel of each background image, producing output images whose pixel values correspond to wavelet coefficients. We refer to the output images containing wavelet coefficients as *wavelet images*.

Each set of four consecutive wavelet images defines a single four-channel texture that compactly represents several frames of the raw input video. The four intensity values of each pixel of this image define a uniform cubic B-spline curve. Evaluating these per pixel B-spline curves at time t_i provides the pixel intensities comprising an image that approximates frame t_i of the raw input video. We refer to the four-channel textures as *wavelet video textures* and note that they can be rendered efficiently using a GPU fragment program.

Let I_u be the set of wavelet images. Each I_u represents 2^k frames of the original input video. At time t_i, the four wavelet images necessary to construct a wavelet video texture are I_{u-1}, I_u, I_{u+1}, and I_{u+2} where $u = i/2^k$. Given the wavelet video texture and a specific point in time, a GPU fragment program renders the appropriate background image by evaluating a uniform cubic B-spline. Once the compressed background image has been rendered, foreground pixels are overlayed onto the result.

10 Discussion

Performing streaming wavelet compression is advantageous for several reasons. It keeps the memory footprint small, which is critical, as the process is performed on a per-pixel basis. Since three internal values per pixel need to be kept between stream updates, 3×2^k values are needed per pixel to perform cubic wavelet compression. Streaming allows values to be returned as soon as possible, enabling results to be displayed sooner despite the extra computation. Our compression saves bandwidth in that a single wavelet image need only be sent once for every 2^k raw input images.

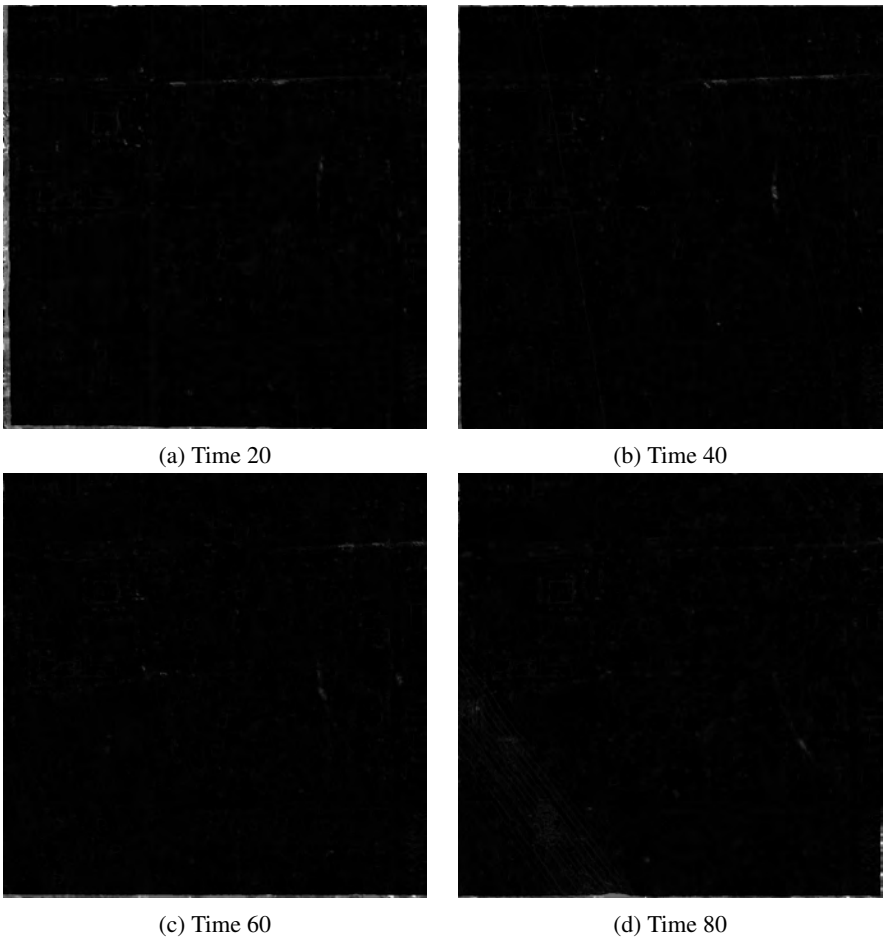
Our wavelet encoding provides a few distinct advantages. The smooth nature of our compression naturally removes undesirable noise found in the original image sequence. The encoding is amenable to hardware-accelerated rendering through the use of texture mapping and programmable GPUs, enabling rendering efficiency. The encoding results in a stack of wavelet image textures, which can be incorporated into state-of-the-art terrain visualization systems capable of managing and rendering large textures [3, 4].

Naturally, streaming compression impacts the latency between when images are captured and when they are displayed. This latency is most significant at the initialization of the compression system, where $3 \times 2^k - 2$ frames are necessary before the first wavelet image is produced. Subsequent wavelet images are produced for every 2^k raw input images, and three additional images are necessary before the first wavelet video texture can be rendered. Thus, the initial cold-start latency is $(3 \times 2^k - 2) + (3 \times 2^k) = 6 \times 2^k - 2$ frames.

Our approach to compression has some limitations. As with many other vision algorithms, our image segmentation procedure is sensitive to reflections off of shiny objects such as bodies of water or metal roof tops. Tall structures such as skyscrapers appear to move as the vantage point of the sensors shifts. Both of these issues cause fluctuations in pixel intensity over time making the problem of identifying foreground movers even more difficult and will be addressed in future work.

11 Results

We have tested our method on a 100 frame sequence consisting of 1 megapixel images. The total raw input size was 100 MB. Our test machine was a 2.8 GHz P4 PC with 2 GB of memory to perform



(a) Time 20 (b) Time 40

(c) Time 60 (d) Time 80

Figure 7 Difference images for select time steps of a 100 frame data set show two things: (1) the error introduced by cubic wavelet approximation is low and (2) some pixels corresponding to moving objects are not classified as foreground.

streaming wavelet compression on the sequence. In our implementation, we use $k = 4$ levels of wavelet analysis for compression, achieving a 16:1 compression ratio.

Figure 7 shows difference images between our compressed result and the original images. They demonstrate that our compression scheme works relatively well, but is sensitive to foreground pixels mislabeled as background pixels. The difference images also illustrate that undesirable noise in the original image sequence do not exist in the compressed result. We note that some images show portions of the original images toward the borders of the images. This is due to image registration, since some frames simply do not contain pixel information contained in other frames.

12 Conclusions and Future Work

We have described a streaming compression algorithm designed for huge time-varying aerial imagery. By treating foreground and background imagery separately, our system is able to achieve high levels of compression. We have developed a streaming formulation for wavelet analysis that satisfies several requirements of high-resolution aerial photography: computational efficiency, low-memory footprint,

compression suitable for transmission in low-bandwidth environments, streaming update of recently captured images, and high-quality approximation to slowly-varying background images. The wavelet images resulting from this compression are combined creating wavelet video textures that compactly represent large numbers of raw input frames and that are inexpensive to render. As our compression method is a streaming technique, it scales well to larger input image sizes and is insensitive to the duration of the image sequence.

As future work, we will investigate ways to achieve further compression. After all pixels are processed in the 1D temporal streaming transform, the wavelet images can be compressed spatially using a number of image compression techniques, including those based on biorthogonal wavelets. We hope to develop compression techniques tailored for fast-varying foreground imagery. Although the compression we have presented is highly effective, it depends on a good segmentation of background from foreground. This prerequisite motivates further development of fast and effective segmentation techniques.

Acknowledgments

This work was supported by Lawrence Livermore National Laboratory under contract B556671. We thank members of the Visualization and Graphics Group of the Institute for Data Analysis and Visualization (IDAV) at UC Davis for their support in this project.

References

1 M. Bertram, M. A. Duchaineau, B. Hamann, and K. I. Joy. Generalized B-spline subdivision-surface wavelets for geometry compression. *IEEE Transactions on Visualization and Computer Graphics*, 2004.

2 D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

3 L. M. Hwa, M. A. Duchaineau, and K. I. Joy. Adaptive 4-8 texture hierarchies. In *Proceedings of IEEE Visualization 2004*, pages 219–226, Los Alamitos, CA, October 2004. IEEE, Computer Society Press.

4 L. M. Hwa, M. A. Duchaineau, and K. I Joy. Real-time optimal adaptation for planetary geometry and texture: 4-8 tile hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):355–368, July-August 2005.

5 M. Isenburg. *Compression and Streaming of Polygon Meshes*. PhD thesis, University of North Carolina, Chapel Hill, 2004.

6 I. Richardson. *Video Codec Design: Developing Image and Video Compression Systems*. John Wiley & Sons, Ltd., 2002.

7 E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA, 1996.

8 W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A. F. Laine, M. A. Unser, and M. V. Wickerhauser, editors, *Wavelet applications in signal and image processing III*, volume 2569 of *Proceedings of SPIE*, pages 68–79, 1995.