ISSN 1614-3213
ISBN 978-3-88579-438-7

This volume contains papers from the first annual workshop of the DFG's International Research Training Group "Visualization of Large and Unstructured Data Sets – Applications in Geospatial Planning, Modeling, and Engineering" held at Dagstuhl Castle in June 2006.

The topics covered in the papers range from fundamental visualization and interaction techniques to visualizations of application data from various domains, such as Geospatial Planning, Modeling, or Engineering.

Hans Hagen, Andreas Kerren, Peter Dannenmann (Eds.):
Visualization of Large and Unstructured Data Sets

S-4

GI-Edition

Lecture Notes
in Informatics

Hans Hagen, Andreas Kerren,
Peter Dannenmann (Eds.)

Visualization of Large and Unstructured Data Sets

First workshop of the DFG's International Research Training Group "Visualization of Large and Unstructured Data Sets – Applications in Geospatial Planning, Modeling, and Engineering"

June 14–16, 2006, Dagstuhl, Germany

Seminars

Hans Hagen, Andreas Kerren, Peter Dannenmann (Eds.)

Visualization of Large and Unstructured Data Sets

First workshop of the DFG's International Research Training Group
Visualization of Large and Unstructured Data Sets – Applications in Geospatial Planning, Modeling, and Engineering

June 14–16, 2006
Dagstuhl Castle, Germany

Gesellschaft für Informatik 2006

Lecture Notes in Informatics (LNI) – Seminars
Series of the Gesellschaft für Informatik (GI)

Volume S-4

ISBN 978-3-88579-438-7
ISSN 1614-3213

Volume Editors
Prof. Dr. Hans Hagen
Department of Computer Science
University of Kaiserslautern
P.O. Box 3049
D-67653 Kaiserslautern, Germany
Email: hagen@informatik.uni-kl.de
Dr. Andreas Kerren
Department of Computer Science
University of Kaiserslautern
P.O. Box 3049
D-67653 Kaiserslautern, Germany
Email: kerren@acm.org
Dr. Peter Dannenmann
Intelligent Visualization and Simulation
German Research Center for Artificial Intelligence (DFKI)
P.O. Box 2080
D-67608 Kaiserslautern, Germany
Email: Peter.Dannenmann@dfki.de

Preface

The International Research Training Group (IRTG) “Visualization of Large and Unstructured Data Sets – Applications in Geospatial Planning, Modeling, and Engineering” is a joint effort of the University of Kaiserslautern (Germany) and the U.S. partners University of California at its Davis and Irvine campuses, Arizona State University, and University of Utah. It is funded by the German Science Foundation (DFG) under grant DFG GK 1131.

The primary research goal of this graduate program is the enhancement of scientific and information visualization techniques applied to large and unstructured data sets. Every visualization task is based on application data. For providing these data, we integrate applications from the domain “Geospatial Planning, Modeling, and Engineering”, which produce these huge amounts of unstructured data that are of interest for the visualization tasks at hand. This integration is necessary to allow a deeper understanding of the provided data due to the sharing of knowledge through the projects.

Up to now, visualization of large and structured or small and unstructured data sets is the state of the art. Large and unstructured data sets are still not very well understood, especially with respect to visualization. In order to address these questions, we have defined a set of projects aiming at solving these problems. In detail, we are handling visualization problems, with respect to modeling, feature detection, and comparison tasks. For doing this, both the extension of existing techniques and the development of new ones are investigated.

In the application areas there is an increasing need to handle huge amounts of unstructured data that are produced either by data from field measurements like environmental observation stations, from experiments, and from simulation. For example, nowadays environmental monitoring systems are capable of measuring data at a very high resolution and in a large number of frequency bands. On the other hand, in scaled-down earthquake laboratory experiments within a centrifuge improved sensor technology permits the measurement of an increased number of parameters at higher sampling rates. Finally, earthquake simulations produce more and more data because of more elaborate simulation techniques. All these improvements in measurement technology lead to large, high-dimensional data sets. Visualizing these data is very useful to get new insights into the problems involved. The visualizations themselves are based on improved or newly developed visualization techniques like volume modeling, feature detection and visualization, etc.

The current issue of GI’s Lecture Notes in Informatics presents the results of the first annual workshop of this IRTG held in Dagstuhl, June 14–16, 2006. Aim of this meeting was to bring together all project partners, advisors, and of course PhD students as well as to report on the different research projects. After three days of presentations and lively discussions at Dagstuhl Castle, slightly more than three months were spent on writing papers that cover the outcome of the first year of the graduate program and give surveys on related topics. These papers were cross-reviewed internally as well as by the project’s advisors. Note that the covered topics do not include all ongoing projects because not all IRTG members could attend.

We would like to thank all attendees for their contribution to this fruitful workshop. We are also grateful to Dagstuhl Castle for their support and hosting this event, and we thank the Gesellschaft für Informatik e.V. (GI) for publishing the workshop papers in the LNI series.

October 2006

Hans Hagen
Andreas Kerren
Peter Dannenmann

Workshop Organizers

Peter Dannenmann
German Research Center for Artificial Intelligence (DFKI), Germany

Hans Hagen
University of Kaiserslautern and
German Research Center for Artificial Intelligence (DFKI), Germany

Younis Hijazi
University of Kaiserslautern, Germany

Andreas Kerren
University of Kaiserslautern, Germany

Burkhard Lehner
University of Kaiserslautern, Germany

Ariane Middel
University of Kaiserslautern, Germany

Principal Investigators of the IRTG[1]

Prof. Dr. Hans Hagen (Speaker and Overall Principal Investigator)
Department of Computer Science
University of Kaiserslautern
P.O. Box 3049
D-67653 Kaiserslautern, Germany
Email: hagen@informatik.uni-kl.de

Prof. Dr. Bernd Hamann (Speaker and Co-Principal Investigator)
Institute for Data Analysis and Visualization
Department of Computer Science
University of California, Davis
One Shields Avenue
Davis, CA 95616-8562, USA
Email: bhamann@ucdavis.edu

Prof. Dr. Paul Steinmann (Co-Principal Investigator)
Department of Mechanical and Process Engineering
University of Kaiserslautern
P.O. Box 3049
D-67653 Kaiserslautern
Email: ps@rhrk.uni-kl.de

[1]`http://www.irtg.uni-kl.de/`

Sponsor

Deutsche
Forschungsgemeinschaft

DFG

Partner Universities

Technische Universität
Kaiserslautern

UCDAVIS
UNIVERSITY OF CALIFORNIA

UCIrvine
University of California, Irvine

U
THE UNIVERSITY OF UTAH

ASU
Arizona State
University

Contents

Survey of Texture-based Techniques in Flow Visualization

Guo-Shi Li, Xavier Tricoche, and Charles Hansen
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT, USA
{lig, tricoche, hansen}@sci.utah.edu

Abstract: Texture-based techniques constitute an effective choice to generate intuitive visualizations of vector fields. The resulting dense representation is a powerful way to convey essential patterns of the vector field while avoiding the tedious task of seeding individual streamlines to capture all the structures of interest. Although this approach has been considered computationally expensive in the past, recently we have seen significant advances in this field thanks to the rapid evolution of graphics hardware and algorithms leveraging the resulting computational power. In this paper, a survey of texture-based flow visualization techniques is presented.

1 Introduction

Vector fields play an important role in many scientific, engineering, and medical disciplines. A wide variety of phenomena, such as electromagnetic fields, fluid flows, or the diffusion process within the living cell, can be described and studied with the help of vector fields. Since the flow field of interest is oftentimes invisible, flow visualization techniques are introduced to depict certain flow properties directly for visual perception. Many of these experimental techniques were first pioneered by renowned fluid dynamics scientists such as Reynolds, Prandtl, and Mach, and they remain invaluable tools to this date.

Over the past few decades we have seen the rapid advancement and progress of Computational Fluid Dynamics (CFD), which has become the primary source of vector field datasets. Vector fields generated from large scale numeric simulations usually exhibit complex structures, therefore proper computational visualization techniques are necessary for researchers and engineers to perform an effective data analysis and comprehend the flow behavior. According to [LvWJH04], there are different approaches to flow visualization:

1. Direct flow visualization

   Techniques belonging to this category directly translate the given flow data into visual representation. Common methods include color coding velocity. The computational cost of this approach is very low, which is valuable since typical datasets are large. Such methods are able to generate straightfoward representation of the two-dimensional flow data. However, when visualizing 3D vector fields, the effectiveness of direct flow visualization is severely hampered by occlusion.

2. Flow integral geometry flow visualization

   To better understand long-term behavior of the vector fields, geometric primitives can be used to further enhance the visual perception. The simplest form of this approach is particle tracing. Assuming a weightless particle is introduced into the vector field, its trajectory $p(t)$ induced by the vector field over time can be obtained by numerically solving the following differential equation:

$$p(t) = p_0 + \int_{s=0}^{t} v(p(s), s)ds$$

   The movement of a particle can be depicted as a moving vertex. Streamlines / pathlines further enhance the visual coherence by connecting successive particle positions along the flow. This idea can be further extended by advecting particles from a rack or a plane to construct stream surface [Hul92] and stream volume [MBC93]. The polygonal geometries constructed by these methods can be then rendered using standard graphics hardware.

   The line, surface, or volume geometries created from flow integral oftentimes can only cover a fraction f the entire domain due to the contracting or diverting behavior of the flow. In order to construct a set of geometries providing a representation of the overall flow behavior, the initial position of the particles, or *seeds*, need to be chosen carefully. There have been a large number of techniques proposed to achieve a better seeding strategies [TB96], [VKP00], [JL00] [MAD05]. This is not a trivial task since knowing the proper location for seeding implies a priori knowledge of the overall flow behavior, which is to be visualized. Even if the seeding locations are chosen carefully so that evenly distributed geometries can be constructed, the flow information in the gap between primitives is still missing.

3. Texture-based visualization

   The dense texture techniques generate a texture covering the entire domain. Every pixel in the visualization is shaded in a way that, the texture as a whole, convey some spatial/temporal patterns induced by the flow. Since the texture is synthesized pixel by pixel, this approach is has been long considered computationally expensive. Since the main topic of this paper is about techniques belonging to this category, we defer more detailed discussions in following sections.

4. Feature-based visualization

   Another approach is to visualize an abstraction of the flow field. The most common examples are topological features, such as sinks, sources, and saddle points.Other feature types exist which are not mathematically well-defined, such as vortices and flow seperation. Corresponding techniques often require expensive flow analysis in order to extract the features of interest. Once they are identified, flow features allow efficient visualization and can be very expressive. [PVH$^+$03] provides a thorough overview of this field.

The remainder of the paper is organized as follows. Section 2 describes texture-based techniques introduced before the emergence of GPU era. In section 3 we put more emphasis on recent research efforts leveraging the SIMD processing power provided by programmable graphics processors. In section 4, we conclude the paper by a discussion and pointing out possible venues of future work.

2 Classical Techniques

As mentioned previously the dense texture flow-based flow visualization is traditionally considered computationally expensive. In the geometry-based approach, visualizing the resulting mesh can be done efficiently using the graphics hardware at interactive frame rates. On the other hand, the texture-based approach needs to shade every pixel in the resulting visualization to properly depict the spatial and temporal behavior of the flow data. In a practical visualization setting, the number of pixels can easily be in the range of several hundred thousands. This is about two to three order of magnitude more than the number of seeds used in a geometry-based approach. Nevertheless, the extra amount of computation yeidls a the visualization that covers the entire domain, exempting the user from having to guess the flow behavior in the unfilled void. In this section, we discuss some of the seminal works considered “classical”, in a sense that they help to coin the concept of *dense texture flow visualization*, as well as being the precursor of more recent algorithms using GPUs.

2.1 Spot Noise

Spot Noise [vW91] is the technique first pioneering the field of dense flow texture visualization. The idea is to use the stochastic texture which local variations to represent the behavior of the underlying vector field. The local distortion of each *spot*, is dictated by the flow direction at that position in the domain. One can think of Spot Noise as the convolution of white noise with *spot* kernels with various sizes, shapes, and patterns. Figure 1 gives a visual impression of this method.

de Leeuw et. al. [dLP$^+$95] improve the spot parameter to simulate the visual effect of an experimental technique where a thin oil film is smeared on the surface of an embeded body. de Leeuw and van Wijk [dLvW95] extend it to work with high speed flows. In [dLvL97] de Leeuw proposes another extension for unsteady flow and provide a parallel implementation for better performance.

2.2 LIC and related works

The expressive power of Spot Noise comes from the collective representation of *local* behaviors. The Line Integral Convolution (LIC) [CL93] provides better illustration of spatial relationship in the flow field by depicting *globally* coherent patterns. Local streamlines are computed at every pixel locations of a white noise, serving as the convolution path of 1D kernel of finite length. As a result, pixels belonging to the same streamline exhibit highly coherent values while those orthogonal to the flow direction are assigned to uncorrelated values due to the stochastic nature of the underlying white noise. A cyclic convolution kernel can be used to create the illusion of flow direction.

Since its introduction LIC has inspired a large number of research works. The list here is not intended to be complete. For a more comprehensive survey of LIC-related methods, please refer to [LHD$^+$04]. Forssell and Cohen [FC95] extend LIC to visualize unsteady flows on surface represented as curvilinear grids. Their approach consisting in applying convolution on pathlines rather than streamlines. Known as *fast LIC*, Stalling and Hege [SH95] propose to accelerate LIC computation by avoiding the redundant computation of streamline construction and convolution. The same authors also experiment using high order kernels to enhance the image quality in [HS98]. Zockler et al [ZSH96] propose to use a massive parallel machine to compute animation of LIC images. Battke et al. [BSH97] extend fast LIC to arbitrary surfaces in 3D represented as triangular meshes by computing LIC texture for each triangle individually in the packed texture domain (Figure 2). Interrante [IG97] applies LIC to visualize volumetric flows. In order to alleviate the occlusion problem, she applies LIC only to a fraction of the volume occupied by a certain number of streamlines while leaving the rest of the volume transparent. Halos are also introduced to surround streamline to enhance the depth perception. Chameleon [LBS03] is inspired by this work and is able to create LIC-like volumetric patterns at interactive frame rates by storing streamline parameterization in the volume which is then used as texture coordinates for texture mapping in the rendering stage. The authors also extend it to work with unsteady flow in [SLB04]. Please see Figure 3.

The methods discussed above are most effective when visualizing steady flows. Although extensions for unsteady flow exists, the visual effectiveness is usually not on par with its steady counterpart due to insufficient visual insights into the spatial and temporal evolvement of the flow. Shen and Kao [SK97] propose Unsteady Flow LIC, or UFLIC, to visualize unsteady flow with highly correlated spatial patterns smoothly evolving over time. Although the name suggests it as an extension of LIC to unsteady flow, it is not working under the same principle as LIC which relies on 1D *convolution* to create the perception of coherent line patterns. Instead, the algorithm utilizes a scheme called *forward scattering* to obtain both temporal and spatial coherency. In *forward scattering*, particles are released at every pixel locations in the domain to sample the current texture (initially a white noise) at each time step. For a prescribed life span, every particles deposit the value it carries to pixels on its path on a canvas where each pixel maintains a specialized data structure to receive values. Along with the value the particle also stores a timestamp indicating when the pixel is visited. To obtain the flow texture for the nth time step, for each pixel the values with timestamp corresponding to the interval between the $n-1$th and the nth time step are used to perform weighted average. This process is repeated for every time step. The spatial pattern perceivable in each image of the UFLIC animation is the combination of the behavior of the current and the past flow behavior. Therefore, the transition of patterns over time is smooth and intuitive.

UFLIC is computationally very demanding. The authors provide a parallel implementation in [SK98]. Liu and Moorehead's AUFLIC (Accelerated UFLIC) [LM02] [LM05] enhances the performance by save, re-use, and update pathlines using a seeding strategy similar to Fast LIC [SH95] on a multi-processor, shared memory machine. However, the performance reported (about 1 frame/sec) is still far from interactivity.

3 GPU techniques

In less than a decade, we have seen significant advancement of Graphics Processing Unit (GPU) technologies. Driven by the demand of the entertainment industry and with the support of the semiconductor manufacturing infrastructure, hardware companies such as nVidia and ATI have made tremendous amount of processing power available for a price of a consumer product. One issue distinguishes a visualization from a graphics prationioner:, while the former needs to construct *graphics* primitives to be rendered from data or an abstraction of data which is oftentimes non-graphical, the latter focuses on techniques to render information which is inherently perceivable. The emergence of GPU, for visualization researchers, opens new possibilities to design novel algorithms since the construction of the graphical representation of the data can be performed or tightly integrated in the rendering stage. This allows many algorithms become interactive and thus greatly improves their usability and effectiveness.

Jobard et al. [JEH00] pioneer the idea of using graphics hardware to synthesize flow textures. Their method, called Lagrangian-Eulerian Advection (LEA), utilizes the *dependent texturing* capability to warp the texture using backward advection at every pixel and then blend the warped texture with the previous one to create the illusion of flow coherence. The reason of using backward advection instead of forward is that the latter can create holes in the domain. Since the blending and dependent texturing are provided by the hardware, the method is capable of running at very high frame rates. However, the visual impression of spatial and temporal coherence in visualizations generated by LEA are limited, mainly due to the short convolution path (only between consecutive frames) and exponentially decay of contribution in the past caused by successive alpha blending.

Image Based Flow Visualization (IBFV) by van Wijk [vW02] provides a GPU-based framework capable of generating a wide range of dense texture representation of flows. The idea is very similar to [JEH00] but instead of advecting every pixels in the domain only vertices of a Cartesian grid are translated by the flow. The warped grid is then used to warp the dense texture using standard texture mapping. The warped texture is then blended with the previous ttexture. At the next iteration the grid is reset before repeating the process just described. In IBFVS [vW03] it is extended to visualize flows defined on polygonal surfaces in 3D. Instead of warping a Cartesian grid in 2D, vertices of a polygonal mesh are translated by the flow in 3D. The projection of the polygonal mesh on the image plane is used to sample and blend the flow textures. Since the texture is defined in the image place rather than the on the object, the patterns generated in this method are not stationary on the object. This can be very confusing when the user interactively rotate or translate the object. Laramee et al [LJH03] introduce a scheme called Image Space Advection(ISA) which is also capable of visualizing flows on 3D surfaces. In ISA the flow is first projected onto image space using standard rasterization provided by the graphics hardware and then is used to warped a 2D Cartesian grid in a manner similar to IBFV. Laramee et al. [LvWJH04] provides an in-depth analysis and comparison of ISA and IBFVS. Please refer to Figure 4.

Among all these similar GPU-based flow visualization techniques proposed in the past few years, Weiskopf et al. [WEE03] is the one of the few providing a framework to dissect this approach in a mathematical and signal processing perspective. In this view, the dense texture approach of visualizing unsteady flow is composed of two components, i.e. 1) the construction of the property field, and 2) the convolution performed upon it. The property field for a 2D unsteady flow is a stack of stochastic noise textures aligned along the time axis where each one is evolved from the previous one by the flow. To compute the visualization, a convolution path in this 3D (x, y, t) domain is used to average the value for every pixel. This framework is capable of explaining several techniques, including ISA, IBFV, and LEA. However, the forward scattering nature of UFLIC does not quite fit into this framework without discrete approximation. The authors also propose a scheme called Unsteady Flow Advection-Convolution (UFAC), which, in the framework they propose, performs convolution along local streamlines with length adjusted to temporal unsteadiness at every pixel location. In [WSEE05] the framework is further improved and the authors propose to use forward advection scheme with radial bais functions to construct the property field.

In general, GPU-based texture techniques are capable of running at high frame rates. This is especially true for image-based techniques such as ISA and IBFVS. The biggest advantage of the image-based approach is that the performance is dependent only on output resolution. Moreover, boundary flows on arbitrary surfaces can be visualized without the explicit knowledge of a surface parameterization, which is difficult if not impossible to construct. The problem of this approach, however, is mainly related to the discontinuity in physical space. Silhouettes and boundaries of the objects are lost in the image-space vector field so that patterns across them can be developed. Both the methods utilize image-processing techniques to detect object silhouettes in the image space used to stop cross-advection. Weiskopf [WSEE05] proposed a hybrid approach where the image-space vector field and the particle positions are maintained in 3D. Alongside with a solid procedural 3D noise, the 3D particle trace will follow the surface of the object hence the patterns developed will never cross silhouettes or boundaries.

Despite high frame rates, the quality of the visualization delivered by most of GPU-based dense texture techniques described so far is not on par with previous off-line techniques such as UFLIC. Li et al [LTH06] proposed a reformulation of UFLIC enabling an efficient GPU implementation. The new algorithm, called GPUFLIC, constructs particle traces iteratively and uses advanced fragment and vertex programmability of modern GPU to perform numerical integration and then achieve value scattering by drawing and blending line segments. Please refer to Figure 5. The frame rate of GPUFLIC outperforms by an order of magnitude to the most efficient software implementation of UFLIC ([LM05]) to our knowledge. Although the performance of GPUFLIC is not as fast as other dense texture approaches mentioned before, it is still interactive while is able to deliver highly coherent patterns with smooth temporal transition at each image in the animation sequence. In constrast, the patterns in each still image in the sequence generated by algorithms such as ISA and IBFVS areis oftentimes not well defined. Therefore the user needs to resort to animation and view the entire animation to better comprehend the flow behavior. This can greatly reduce the effectiveness of visualization when the dataset of interest is spatially/temporally complicated. However, just as the original algorithm it is unclear how to apply UFLIC to an arbitrary surface that lacks a global parameterization.

4 Conclusions and Future Prospect

Texture-based flow visualization have been applied to visualize a wide range of fluid flow datasets and have achieved a fair degree of success. As of now, 2D unsteady flows can be visualized with high quality at interactive frame rates using algorithms such as UFAC and GPUFLIC. The image-space approach such as ISA and IBFVS can visualize unsteady boundary flows, however we believe there is still room to improve the visual expressiveness. As for 3D volumetric flows, the effectiveness of the texture-based approach may be greatly limited due to occlusion and visual perception issues. Techniques such as cutting plane and heptics have been utilized together with dense texture representation of 3D flows [RSHTE99]. When looking at experimental techniques used by researchers in various engineering disciplines, we believe that interactive spatially-selective texture representations of 3D flow, such as dye/smoke advection, may be an alternative interesting venue for future exploration.

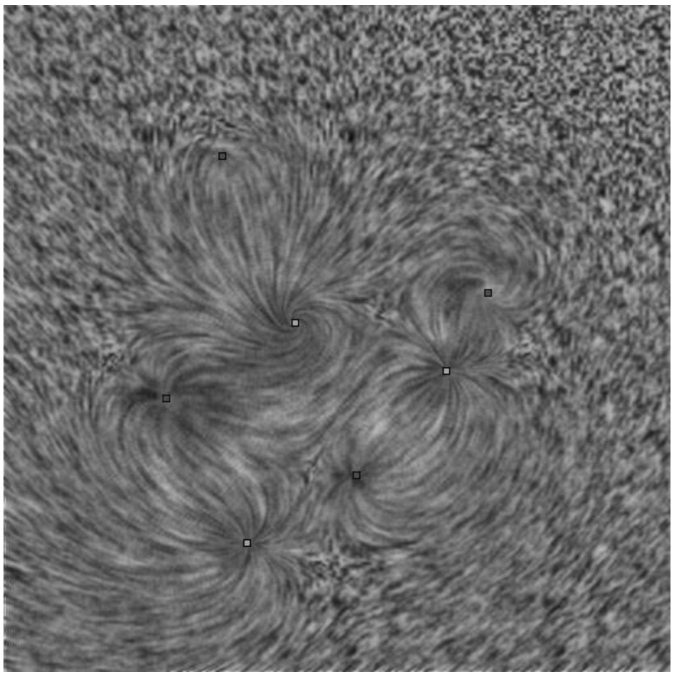


Figure 1: Spot Noise [vW91]. Image courtesy of van Wijk.

References

[BSH97] Henrik Battke, Detlev Stalling, and Hans-Christian Hege. Fast line integral convolution for arbitrary surfaces in 3D. pages 181–ff., 1997.

[CL93] B. Cabral and C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of SIGGRAPH 93*, pages 263–270. ACM SIGGRAPH, 1993.

[dLP$^+$95] W. de Leeuw, H. Pagendarm, , F. Post, and B. Waltzer. Visual Simulation of Experimental Oil-Flow Visualization by Spot Noise Images from Numerical Flow Simulation. In *Visualization in Scentific Computing '95*, pages 135–148. Springer-Verlag, 1995.

[dLvL97] Wim de Leeuw and Robert van Liere. Divide and conquer spot noise. In *Supercomputing '97: Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–13, New York, NY, USA, 1997. ACM Press.

[dLvW95] Wim C. de Leeuw and Jarke J. van Wijk. Enhanced Spot Noise for Vector Field Visualization. In *IEEE Visualization*, pages 233–239, 1995.

[FC95] L.K. Forssell and S.D. Cohen. Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation, and Unsteady Flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, 1995.

[HS98] Hans-Christian Hege and Detlev Stalling. Fast LIC with Piecewise Polynomial Filter Kernels. In H.-C. Hege and K. Polthier, editors, *Mathematical Visualization - Algorithms and Applications*, pages 295–314. Springer, 1998.

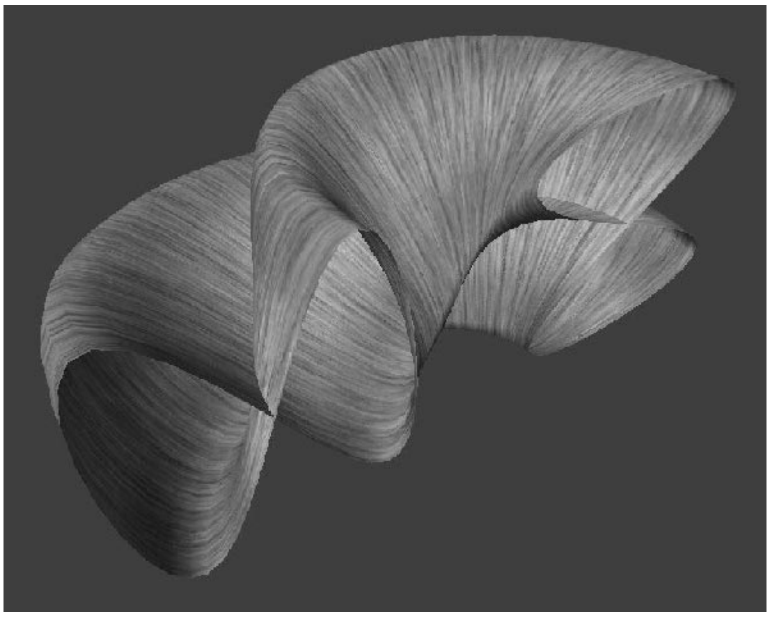


Figure 2: LIC on polygonal surface in 3D [BSH97]. Image courtesy of Battke.

[Hul92] J. Hultquist. Constructing Stream Surfaces in Steady 3D Vector Fields. In *Proceedings of Visualization '92*, pages 171–178. IEEE Computer Society Press, 1992.

[IG97] V. Interrante and C. Grosch. Strategies for Effectively Visualizing 3D Flow with Volume LIC. In *Proceedings of Visualization '97*, pages 421–424. IEEE Computer Society Press, 1997.

[JEH00] Bruno Jobard, Gordon Erlebacher, and M. Yousuff Hussaini. Hardware-accelerated texture advection for unsteady flow visualization. In *Proceedings of the conference on Visualization '00*, pages 155–162. IEEE Computer Society Press, 2000.

[JL00] B. Jobard and W. Lefer. Unsteady Flow Visualization by Animating Evenly-Spaced Streamlines. *Computer Graphics Forum (Proceedings of Eurographics 2000)*, 19(3), 2000.

[LBS03] Guo-Shi Li, Udeepta D. Bordoloi, and Han-Wei Shen. Chameleon: An Interactive Texture-based Rendering Framework for Visualizing Three-dimensional Vector Fields. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 32, Washington, DC, USA, 2003. IEEE Computer Society.

[LHD$^+$04] R.S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F.H. Post, and D. Weiskopf. The State of the Art in Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum*, 23(2):143–161, 2004.

[LJH03] Robert S. Laramee, Bruno Jobard, and Helwig Hauser. Image Space Based Visualization of Unsteady Flow on Surfaces. In *Proceedings of the 14th IEEE Visualization 2003*, page 18, Washington, DC, USA, 2003. IEEE Computer Society.

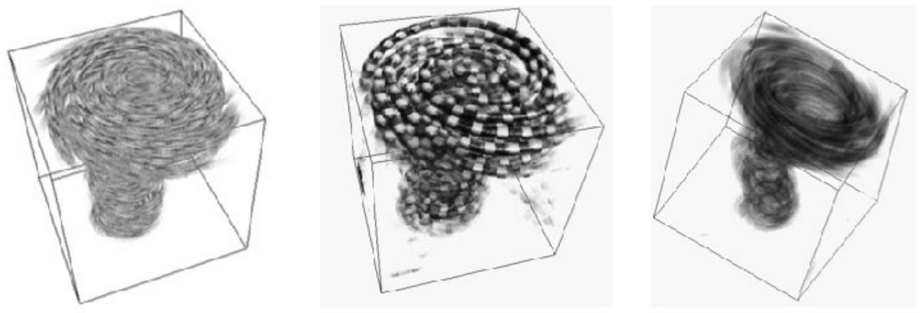


Figure 3: Left: LIC representation using Chameleon [LBS03] [SLB04]. Middle: solid patterns. Right: NPR patterns on the same dataset. Image courtesy of Li et al.

[LM02] Zhanping Liu and Robert James Moorhead. AUFLIC: An Accelerated Algorithm for Unsteady Flow Line Integral Convolution. In *Proceedings of Data Visualization.*, 2002.

[LM05] Zhanping Liu and Robert J. Moorhead. Accelerated Unsteady Flow Line Integral Convolution. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):113–125, 2005.

[LTH06] Guo-Shi Li, Xavier Tricoche, and Charles Hansen. GPUFLIC: Interactive and Accurate Dense Visualization of Unsteady Flows. In *Eurographics/IEEE-VGTC Symposium on Visualizations*, 2006.

[LvWJH04] R. S. Laramee, J. J. van Wijk, B. Jobard, and H. Hauser. ISA and IBFVS: Image Space-Based Visualization of Flow on Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):637–648, 2004.

[MAD05] Abdelkrim Mebarki, Pierre Alliez, and Olivier Devillers. Farthest Point Seeding for Efficient Placement of Streamlines. In *IEEE Visualization*, page 61, 2005.

[MBC93] N. Max, B. Becker, and R. Crawfis. Flow Volumes for Interactive Vector Field Visualization. In *Proceedings Visualization '93*, pages 19–24. IEEE CS Press, 1993.

[PVH$^+$03] 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. *Comput. Graph. Forum*, 22(4):775–792, 2003.

[RSHTE99] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. Interactive Exploration of Volume Line Integral Convolution Based on 3D–Texture Mapping. In *Proc. Visualization '99*, pages 233–240. IEEE, 1999.

[SH95] D. Stalling and H.-C. Hege. Fast and resolution independent Line Integral Convolution. In *Proceedings of SIGGRAPH '95*, pages 249–256. ACM SIGGRAPH, 1995.

[SK97] Han-Wei Shen and David L. Kao. UFLIC: a line integral convolution algorithm for visualizing unsteady flows. In *Proceedings of the conference on Visualization '97*, pages 317–323. ACM Press, 1997.

[SK98] H.-W. Shen and D.L Kao. A New Line Integral Convolution Algorithm for Visualizing Time-Varying Flow Fields. *IEEE Transactions on Visualization and Computer Graphics*, 4(2), 1998.

[SLB04] Han-Wei Shen, Guo-Shi Li, and Udeepta Bordoloi. Interactive Visualization of Three-Dimensional Vector Fields with Flexible Appearance Control. *IEEE Trans. Vis. Comput. Graph.*, 10(4):434–445, 2004.

[TB96] G. Turk and D. Banks. Image-guided Streamline Placement. In *Proceedings of SIGGRAPH '96*, pages 453–460. ACM SIGGRAPH, 1996.

[VKP00] V. Verma, D. Kao, and A. Pang. A Flow-guided Streamline Seeding Strategy. In *Proceedings of Visualization '00*, pages 163–170. IEEE Computer Society Press, 2000.

[vW91] J. van Wijk. Spot Noise: Texture Synthesis for Data Visualization. *Computer Graphics*, 25(4):309–318, 1991.

[vW02] J van Wijk. Image based flow visualization. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 745–754. ACM Press, 2002.

[vW03] Jarke J. van Wijk. Image Based Flow Visualization for Curved Surfaces. In *Proceedings of the 14th IEEE Visualization 2003*, page 17, Washington, DC, USA, 2003. IEEE Computer Society.

[WEE03] Daniel Weiskopf, Gordon Erlebacher, and Thomas Ertl. A Texture-Based Framework for Spacetime-Coherent Visualization of Time-Dependent Vector Fields. In *Proceedings of the 14th IEEE Visualization 2003*, page 15, Washington, DC, USA, 2003. IEEE Computer Society.

[WSEE05] D. Weiskopf, F. Schramm, G. Erlebacher, and T. Ertl. Particle and Texture-Based Spatiotemporal Visualization of Time-Dependent Vector Fields. In *Proceedings of IEEE Visualization 2005*, pages 639 – 646, Washington, DC, USA, 2005. IEEE Computer Society.

[ZSH96] M. Zöckler, D. Stalling, and H.-C. Hege. Parallel Line Integral Convolution. In *Proceedings of First Eurographics Workshop on Parallel Graphics and Visualisation*, pages 111–128, 1996.

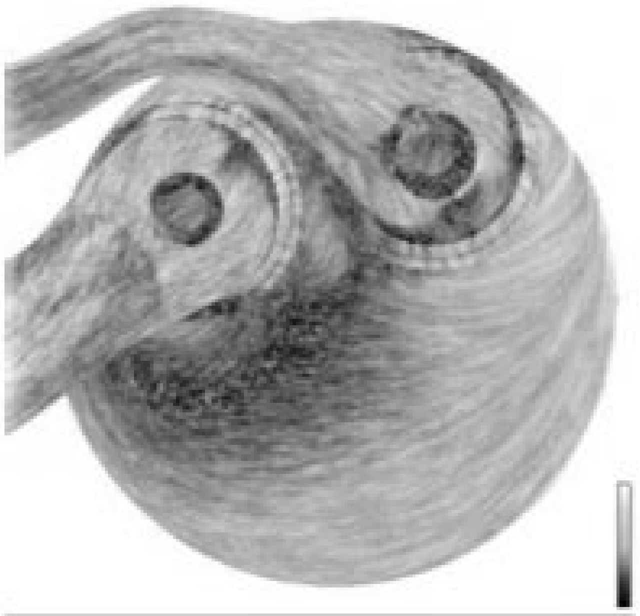
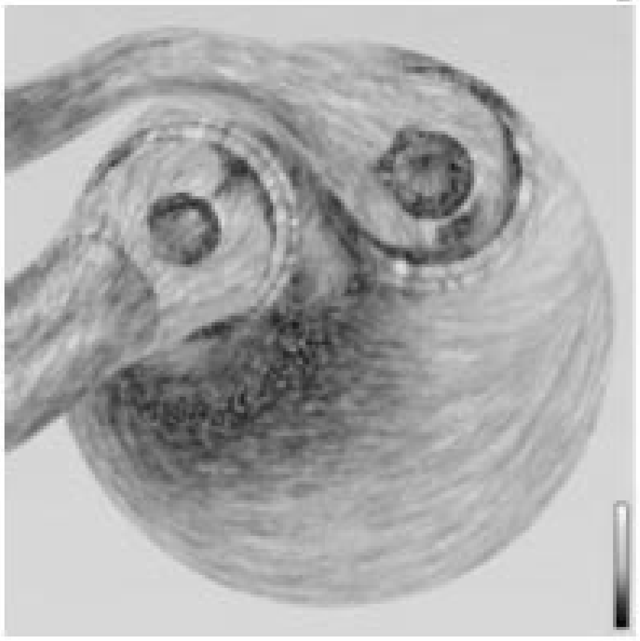
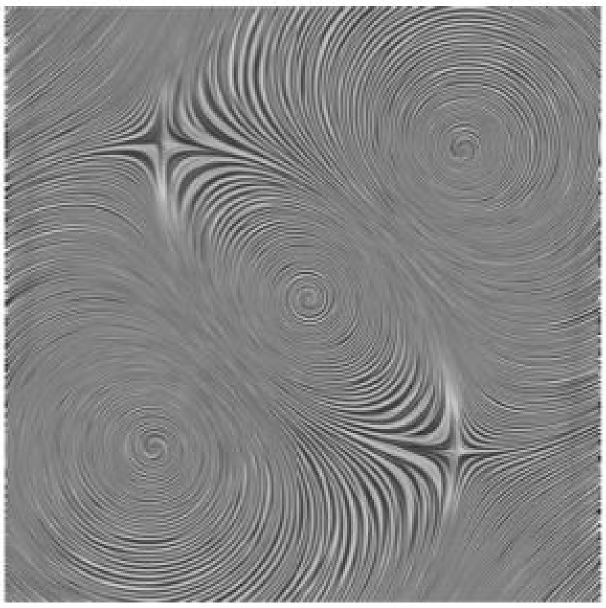



Figure 4: ISA and IBFVS [LvWJH04]. Up: ISA, Bottom: IBFVS. Image courtesy of Laramee at el.



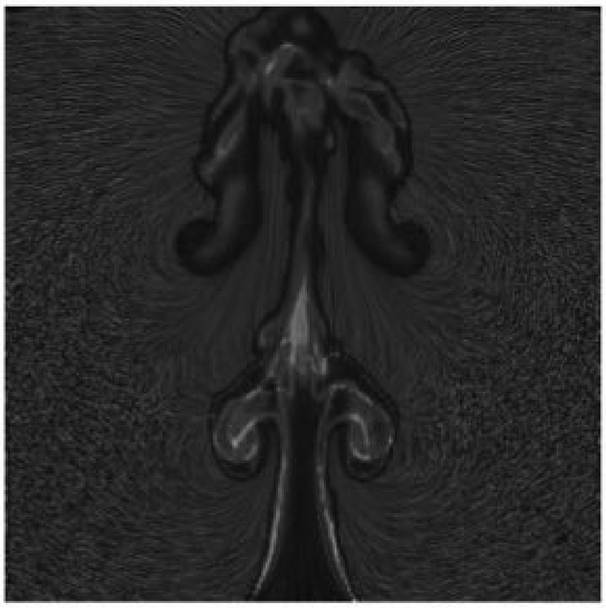


Figure 5: GPUFLIC. Image courtesy of Li at el. [LTH06]

Topology- and Feature-based Flow Visualization: Methods and Applications

Christoph Garth
Computer Graphics & Visualization Group
University of Kaiserslautern
Kaiserslautern, Germany
garth@informatik.uni-kl.de

Xavier Tricoche
Scientific Computing & Imaging Insitute
University of Utah
Salt Lake City, Utah
tricoche@sci.utah.edu

Abstract: In this paper we give a survey on topology- and feature-based flow visualization methods. After an introduction into the respective methodologies and a basic classification of techniques, we will focus on practical aspects and several application examples from different areas of flow physics. We show the utility and importance of these classes of tools in the design and analysis of technical flows.

1 Introduction

The ongoing strive for improved efficiency, performance, and safety is at the core of technical flow design. Typical examples of such flows can be found at all scales, ranging from the gas-burning chamber in household heating appliances that depends on an optimal mixture of oxygen and gas, to high-performance aircraft design. In the recent past, computers have become powerful enough to be truly instrumental in this field, primarily through the widespread use of Computational Fluid Dynamics (CFD). This methodology offers new means to rapidly simulate and evaluate new designs. Moreover, it allows for unprecedented insight into complex fluid dynamics phenomena observed in practical experiments by generating very high resolution data sets that accurately reproduce the entirety of the flow. This evolution emphasizes the need for analysis tools that are both effective and efficient. Scientific Visualization has become essential in this context.

The dedicated research effort is called Flow Visualization. Its major task is to provide tools that allow the user to visually explore and assess the properties of the ever-increasing amount of numerical information that results from CFD computations. One classical approach is to focus the visualization on features of interest that engineers and fluid dynamicists consider essential for both scientific and industrial applications. Prominent examples are vortices, shock waves, and separation or attachment lines. The corresponding visualization techniques are very useful in practice because they yield a simplified representation of involved flow phenomena made of patterns that directly match the intuition of the observer. Their limitation, however, follows from the loose notion of feature that, in most cases, is essentially application specific. In general, different methods rely on different (if not contradictory) definitions of the same feature and therefore yield heterogeneous results.

Vortices constitute a typical illustration of that problem: many definitions have been proposed over the years but none of them is able to properly characterize vortices in all types of flows [Lug96]. Following a different approach topology-based methods extract global flow structures defined with respect to the limit sets of streamlines. The corresponding techniques are built upon the rigorous mathematical formalism of the qualitative theory of dynamical systems, which guarantees objective results. Unfortunately, the connection between topological structures and the practical properties of the flow is sometimes unclear and the resulting pictures tend to lack the intuitive appeal of feature-based representations.

The objective of the paper is two-fold. First, it provides an introduction to state-of-the-art feature and topology-based flow visualization methods. Beyond the techniques traditionally used in practice it describes recent contributions made by the authors, following an approach aimed at combining the strengths of both topological and feature-based techniques to yield more effective visualizations. Second, it demonstrates the use of these algorithms for practical applications. In particular it discusses their ability to meet the needs raised by the analysis of large-scale multi-field CFD data sets.

The contents of this paper are organized as follows. Our description starts with an overview of feature-based flow visualization techniques in section 2. We focus our presentation on the feature types that are most prominent in practice. Topology-based methods are introduced in section 3. Basic notions are defined along with algorithms relevant for visualization purposes. After these initial considerations we adopt a more practical viewpoint and consider successively two different visualization applications. One is concerned vortex breakdown analysis, as discussed in section 4. The other is dedicated to flow analysis and optimization in engine components, section 5.

2 Feature-based Visualization

The goal of feature-based visualization methods is to generate images that restrict the depiction of complex flow data to a limited set of points, lines, and volumes representing features of particular interest for the considered application. This yields fairly abstract pictures that convey significant flow properties in a concise and compact form. The most prominent examples of features in CFD applications include vortices, separation and attachment lines, shock waves and recirculation zones.

The loose, empiric nature of the definition of those feature explains the variety of algorithms available to locate, identify, and visualize them and requires the user to determine experimentally which method is best suited for the needs of his particular application. Further restrictions on the type of method can be imposed by the size or the structure of the data.

In this section we present visualization methods dedicated to vortices on one hand, and separation and attachment lines on the other hand. This choice is motivated by the major practical significance in practice as explained below and illustrated in sections 4 and 5. For a more general introduction to the topic of feature-based visualization, refer to [PVH+02].

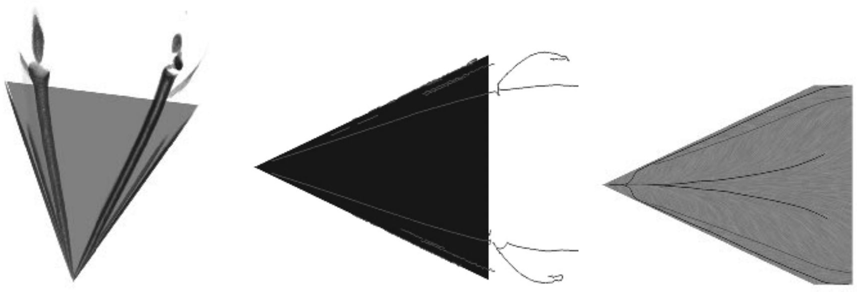


Figure 1: Samples of feature-based visualization methods. **Left:** Volume rendering of the λ_2-criterion to illustrate vortices above a delta wing. **Middle:** Vortices as extracted by the Sujudi-Haimes algorithm. Note the false positives and the generally bad quality of the results. **Right:** Separation and attachment lines of the shear flow on the wing surface, computed using the scheme from[TGS03] (cf. 4 for a detailed discussion of this dataset).

2.1 Vortices

The extraction of vortical structures has been a major topic in visualization for quite some time. Although a vortex is most intuitively conceived as the superposition of a flow along an axis and a flow around this axis, there is no satisfying and objective definition that exists for this flow pattern. As a result, vortex extraction methods are essentially characterized by the type of vortex criteria they are built on. This is either a region-based criterion (identifying regions of vortical flow behavior) or a line-type description (focusing on the vortical axis or vortex core line). Region definitions include high vorticity, helicity, low pressure. Most often used in engineering is the λ_2 definition by Jeong and Hussain [JH95]. The physical meaning behind this method is a similarity measure of the local flow structure to that induced by a pressure valley line. The major limitation of λ_2, however, lies in its incapacity to isolate individual structures.

Among the line-type definitions, the approach of Sujudi and Haimes [SH95] is most widely used. The idea here is to perform on a cell-wise basis the pattern matching of a rotation motion on the vector field and to extract locally sections of the rotation axis that can be patched together to approximate the vortex core line. Because of the linear nature of the sought pattern, the method has issues with vortex core lines that are strongly curved. Roth and Peikert proposed a higher-order scheme that can extract curved core lines reliably [RP98]. They also showed in a subsequent paper that this and other similar methods can be formulated in a unified framework involving their *parallel operator* [PR99].

Additionally, some approaches try to identify a vortical or swirling flow behavior by examining the evolution of particles [JMT02]. Recently, Garth et al. presented a more general stream surface[1]-based approach [GTSS04] that allows for the extraction of a vortex core line approximated as the medial axis of a stream surface that exhibits vortical behavior. The stream surface can be seeded along a closed curve surrounding the center of a two-

[1]A stream surface is the surface spanned by an infinite set of streamlines starting on an arbitrary seeding curve

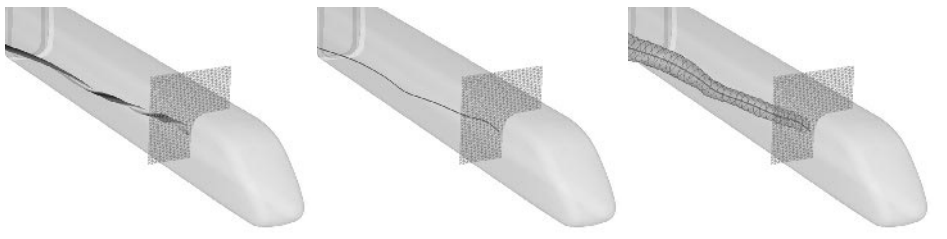


Figure 2: An illustration of the vortex extraction algorithm from [GTSS04]. **Left:** Manually seeded stream surface showing vortical behavior. **Middle:** Stream surface medial axis as a smooth approximation to the vortex core line. **Right:** Vortex surface grown outward from the medial axis.

dimensional rotation identified on a probing plane that the user can position arbitrarily in the flow. The starting curve can also be adjusted to the vortex core line approximation provided by another scheme. The authors combine this technique with a region-growing algorithm that relies on a common vortex model from fluid dynamics to identify a vortex region. Overall this method proves significantly more robust than alternative solution when applied to slowly swirling vortices. An overview of the successive steps of the method is proposed in Fig. 2.

While most of the methods mentioned above provide satisfactory visualizations for simple datasets, the extraction of vortices in modern CFD datasets remains essentially challenging and a significant amount of user interaction is required to obtain useful results.

2.2 Separation and Attachment Lines

Separation and attachment lines are another major feature type. They are defined as the lines along which the flow attaches or separates from the surface of an embedded body (e.g. an aircraft). This phenomenon is induced by viscous effects that take place in direct proximity of the object. In that setting, the flow has so-called no-slip boundary condition which means that the velocity magnitude goes to zero as one approaches the surface along a normal direction. Therefore, the analysis is focused on the non-zero, tangential shear-stress vector field defined over the surface that exhibits the same flow patterns as nearby located streamlines. In particular flow separation and attachment induce the creation of curves of asymptotic streamline convergence which are visible in the shear stress vector field. The corresponding three-dimensional flow pattern is characterized by the presence of a stream surface starting or ending along the feature line that, on the other hand, swirls around a nearby located vortex. As a matter of fact, flow separation and vortex genesis are two closely related phenomena.

Following the original idea of Sujudi and Haimes for vortex core lines, Kenwright et al. proposed a simple and fast method for the extraction of separation and attachment lines [KHL94]. Their basic observation is that these feature lines are present in two lin-

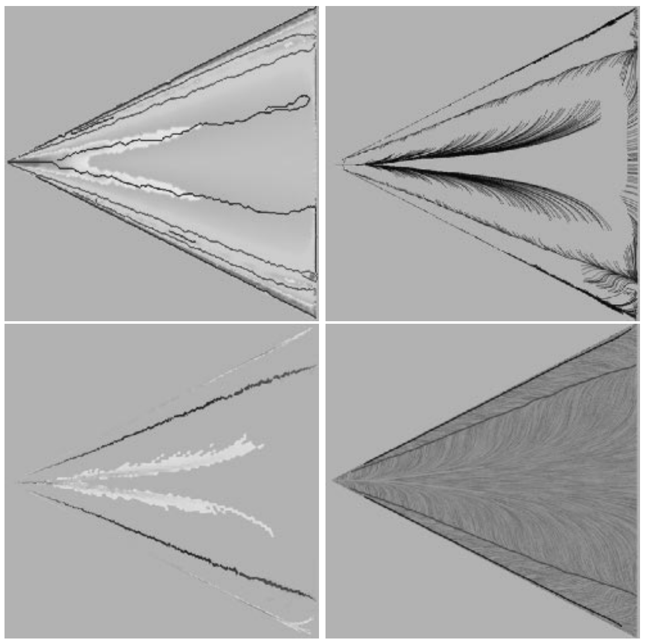


Figure 3: Successive steps of the extraction method for separation and attachment lines [TGS03]. **Top left:** ridge and valley lines of density function. **Top right:** streamlines seeded along ridge and valley lines. **Bottom left:** cell-wise accumulation monitoring. **Bottom right:** separation and attachment lines.

ear patterns, namely saddle points and nodes (see section 3), where they are aligned with an eigenvector of the Jacobian. The original method works on a cell-wise basis and extract these pattern within each triangle. Hence it results in disconnected line segments, caused to the discontinuity of the Jacobian. Yet, applying the parallel operator leads to the reformulation of the features in terms of lines of zero curvature and yields connected lines. However this definition is quite restrictive because it assumes that separation resp. attachment lines always have zero curvature. Moreover, since it requires derivative computation it is very sensitive to noise. Consequently strong pre-smoothing of the data is often necessary which in turn can deform and shift the features. Another approach was proposed earlier by Okada and Kao [OK97] who extend the classical Line Integral Convolution (LIC) algorithm [CL93] by color coding the flow direction so as to highlight the fast changes in flow direction that occur as streamlines approach separation resp. attachment lines. The weakness of this approach lies in the heavy computation associated with LIC on one hand, and in the fact that the geometry of the feature lines is not extracted. Instead, the method computes a density function that indicates the proximity / likelihood of these feature lines.

Using a different approach, Tricoche et al. recently proposed a scheme [TGS03] designed to overcome the restrictions imposed by the purely local analysis used in the algorithms mentioned previously. Their method can be decomposed in three stages. The first one applies a local criterion to estimate the likelihood of each grid vertex to lie close to a separation resp. attachment line. In essence, this stage is comparable to the method of Okada and Kao [OK97]. However the density computed here is not directly visualized but serves as input for the next step of the algorithm. Moreover the local criterion used can be chosen arbitrarily by the user. In particular, the pattern matching idea underlying Kenwright's method [KHL94] can be reformulated to yield a distance value. The second step leverages these local estimates to monitor the global convergence of streamlines toward separation resp. attachment lines. This step is justified by the asymptotic streamline convergence that takes place along separation and attachment lines in the shear stress vector field. Observe that streamline integration is global in nature, which makes the method both more robust and more flexible than Kenwright's approach. Practically, the ridge and valley lines of the density function computed previously are extracted. The resulting skeleton is then used to restrict the candidate seed points for streamline integration to a set of isolated lines that are discretized at a predefined resolution. Streamline convergence is measured on a cell-wise basis by incrementing a local counter every time a streamline crosses a cell. The final step consists in extracting the ridge and valley lines of this convergence map. This yields an approximation of the location of separation and attachment lines, the accuracy of which is determined by the grid resolution. The actual geometry is eventually provided by streamline integration. An overview of these successive steps is shown in Fig. 3.

3 Topology-based Visualization

Vector field topology is a powerful approach for the visualization of planar flows. Topology-based methods leverage basic results of the qualitative theory of dynamical systems to generate effective depictions characterized by a high level of abstraction and an accurate segmentation of the domain in regions where the flow exhibit a uniform behavior. Formally, this classification is defined with respect to the limit sets of the streamlines. Additionally, parametric topology and the notion of bifurcation can be used to extend this technique to time-dependent flows and account for the structural transformations that their topology undergoes over time.

Unfortunately, the application of this methodology to three-dimensional problems has not so far demonstrated the same usefulness in visualization applications. One explanation is the intricacy of the resulting pictures: the topology of volume flows involve stream surfaces that are plagued by self-occlusion and visual clutter. Another problem concerns the lack of intuitive connection between topological structures and major features of interest in fluid dynamics problems, as described in the previous section. Neither vortices nor separation lines are, in general, topological in nature. Thus topology-based methods fail to extract them properly.

In the following we provide a short introduction to essential notions of planar and three-dimensional vector field topology. Our presentation is driven by the needs of visualization algorithms discussed in the next section. For a more complete survey of existing methods in topology-based flow visualization, we refer the reader to [ST05].

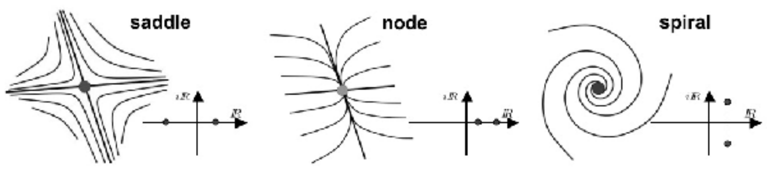


Figure 4: Linear planar critical points

3.1 Topological Skeleton of Steady Flows

The topology of a vector field is the decomposition of its phase portrait into regions where all streamlines have the same limit sets. The *phase portrait* considers all points located along the same streamline as a single equivalence class. In other words, it interprets the domain in terms of its dense coverage by streamlines. A *limit set* is defined as a set of points that constitute the asymptotic limit of a streamline, either forward or backward. We restrict our considerations to the two most common types of limit sets: critical points and cycles.

3.1.1 Limit Sets

The *critical points* of a steady vector field are the locations where the field magnitude vanishes. Because of the uniqueness of the solution of a dynamical system with respect to its initial conditions, critical points are the only locations where streamlines can meet asymptotically. In the non-degenerate, linear case, the nature of a critical point is determined by the eigenvalues of the Jacobian matrix. The different types are illustrated in Fig. 4 and Fig. 5 for the planar and three-dimensional case, respectively. In both cases, the eigenvalues are shown along with the associated configuration. When all the eigenvalues have positive (resp. negative) real parts, the critical point is a source (resp. sink). If both positive and negative real parts are present, the critical point exhibits both source and sink behavior and is called a saddle point.

Cycles are closed streamlines that correspond to periodic solutions of the dynamical system. The non-degenerate case corresponds to cycles that act as sink or sources with respect to the surrounding streamlines.

3.1.2 Separatrices

The limit sets present in a vector field induce a segmentation of the domain into regions where all streamlines share the same limit sets for forward and backward integration. The boundaries between such regions are called separatrices.

Specifically, in the planar case two major types of separatrices exist: cycles and streamlines that attach to a saddle point along its eigenvectors. Refer to Figure 4. As in the planar setting, separatrices of the three-dimensional topology are either periodic manifolds or

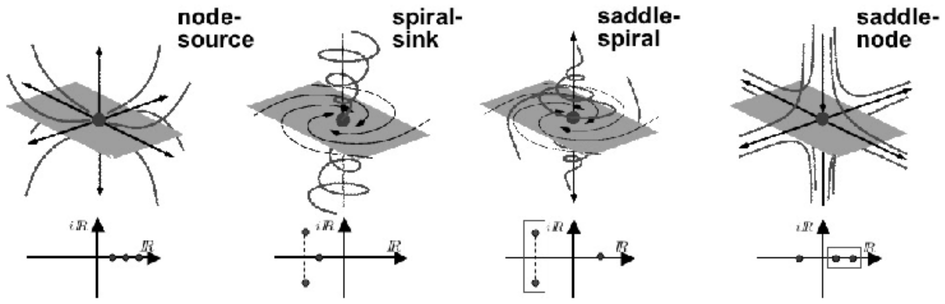


Figure 5: Linear critical points in three-dimensions

they start at saddle points along their eigenvectors. However these separatrices are either 1D (streamlines) or 2D (stream surfaces). The latter are spanned by both eigenvectors associated with the eigenvalues whose real parts have same sign. Again, refer to Fig. 5.

3.2 Topology Extraction

In practice critical points are extracted on a cell-wise basis. In the piecewise linear case, the equation to solve in each cell to determine the position of a zero vector is linear. In the planar, bilinear case, the same problem leads to a quadratic equation. Trilinear interpolation requires numerical schemes like Newton-Raphson to locate critical points. If such a point is found in the interior of a given cell, its type is determined by solving the associated Eigensystem. In the case of saddle points, this analysis also provides the eigenvectors along which the integration of separating streamlines or stream surfaces is carried out.

The extraction of cycles is more involved. Wischgoll and Scheuermann proposed the first method for that purpose [WS01]. The basic idea of their algorithm is to first identify a cell-wise cycle in which a streamlines appears to be captured. The next step consists in verifying this property by integrating streamlines from the boundary of the cell cycle to ensure that the vector field does indeed prevent any streamline from leaving the corresponding region. This result is then combined with the Poincare-Bendixson theorem which states that in the absence of any critical point in the cell cycle, the limit set present in this region must be a cycle. A method derived from the same principle was later proposed by the same authors for the 3D case [WS02]. Observe that tori that are stable under the flow are another possible generalization of cycles in a three-dimensional setting.

3.3 Parametric Topology and Bifurcations

When a vector field depends on a parameter (e.g. time), changes in the parameter value induce changes to its topology. These transformations are called bifurcations and exist

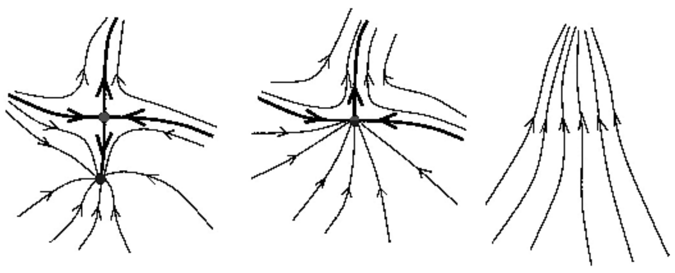


Figure 6: Fold bifurcation

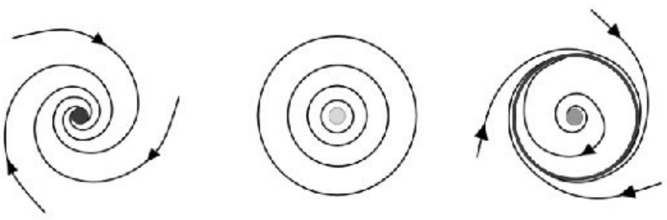


Figure 7: Hopf bifurcation

in an infinite variety. Their common property nonetheless is to replace a stable structural configuration by another stable configuration through an instantaneous, unstable pattern. Here, stability is defined with respect to the ability of a given structure to remain qualitatively unchanged after a small but arbitrary modification of the vector field. Bifurcations are either local or global depending on the extent of the region they impact.

For the need of this presentation, we consider only local bifurcations. The most common ones in the planar case fall in two categories: fold and Hopf bifurcation. The former corresponds to the pairwise annihilation (resp. creation) of a saddle point and a sink or source. An example is shown in Fig. 6. The latter is characterized by the transformation of a sink into a source (and vice versa) and the simultaneous creation (resp. annihilation) of a cycle surrounding the critical point. See Fig. 7.

Similar transformations occur in the 3D case. A simple example can be derived from a 2D fold bifurcation by adding a one-dimensional source behavior to a saddle point and a source affected by the transformation. This creates two 3D saddle points that merge and vanish in the very same way. This is illustrated in Fig. 8.

3.4 Topology Tracking

A simple algorithmic solution to track the continuous evolution of the topology and detect the associated bifurcations was proposed by Tricoche et al. in [TWSH02]. The method was designed for two-dimensional unsteady flows. Its basic idea is to embeds the discrete space-time domain of the data in a three-dimensional grid. More precisely, assuming that the grid is a fixed triangulation, each triangle is expanded along the time axis which creates a prism that connects two consecutive time steps. In each prism, a linear interpolant along the time axis extends to 3D the planar, piecewise linear interpolation defined at each time step.

In this prism grid, the singularities are tracked in a cell-wise manner. Given a critical point present in a cell at a time step the algorithm uses the equation of the 3D interpolating function to determine the path of this critical point throughout the prism. Observe that the interpolation is chosen such as to ensure that at most one critical point is present in a cell at any given time. Therefore only two cases are possible for this path: it can either cross the prism from one time step to the next (i.e. move within the same triangle cell in a 2D perspective) or cross the side faces of the prism before reaching the next time step (that is, leave the triangle where it was located initially between two time steps). When a critical point leaves a prism, its path must be followed further in the corresponding neighbor.

As far as bifurcations are concerned, they can take two forms in that framework. The only local bifurcation that can occur in the interior of a cell (thus involving a single critical point) is a Hopf bifurcation. It is detected by checking the persistence of the sink/source type of the critical point along its path. The cycle originating at the bifurcation point can be tracked by extracting it at each subsequent time step using the scheme of Wischgoll and Scheuermann [WS01]. Fold bifurcations are constrained to take place on the side faces of a prism. In that case, the path of a saddle point contained in a prism connects with the path of a sink (resp. source) located in a neighboring prism. As explained previously, this configuration can either cause the creation or the annihilation of both critical points. The separatrices emanating from a saddle point can be integrated at each point along the path and reconnected along the time axis to span surfaces in the space-time domain.

The extension of this scheme to three-dimensional vector fields was described by Garth et al. in [GTS04]. The basic idea is the same as in the 2D case. However the space-time domain is decomposed into 4D simplices in their implementation. This choice greatly simplifies the equations to be solved to track the critical points and determine their type everywhere along the time axis. The application of this method to the temporal analysis of a vortex breakdown bubble is discussed in section 4.3. Remark that an alternative technique for topology tracking in two and three dimensions called *Feature Flow Field* was proposed by Theisel and Seidel [TS03]. However, their method requires all time steps to be available in memory at once which makes it non suitable to handle the large data sets considered in the next section.

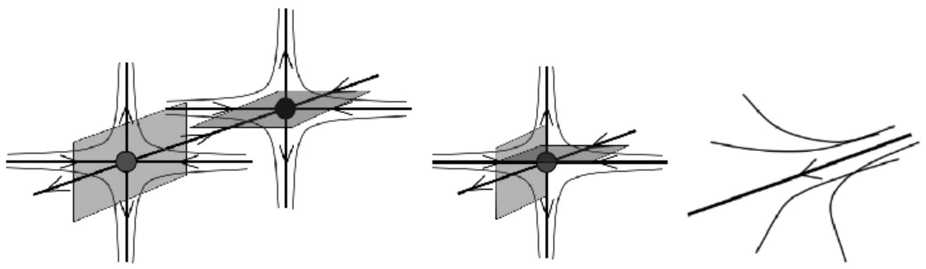


Figure 8: Three-dimensional fold bifurcation

3.5 Cutting Plane Topology

To finish this overview of topology-based flow visualization methods, we briefly present a technique called *Cutting Plane Topology* introduced recently by Tricoche et al. [TGK +04]. It is based on the topology tracking algorithm mentioned previously and permits to extract and explore complex three-dimensional flow structures. Specifically, a steady three-dimensional flow is investigated through the parametric topology of its 2D projection onto a plane that is swept along a prescribed curve across a volume of interest. In other words, the curve controls how the cutting plane is moved to span a 3D volume and is interpreted as the parameter space for topology tracking. The choice of this curve is therefore application specific. We describe in section 4.2 how the inherent symmetry of the considered flow structures can be exploited to create effective visualizations that unravel intricate flow structures. Another important aspect to consider here is the orientation of the plane as a continuous function of its position along the curve. Once again this choice must be dictated by the flow to yield meaningful results. The solutions proposed in [TGK +04] range from a fixed orientation imposed by a symmetry axis to a direction chosen in order to maximize the amount of flow crossing the plane. Once the curve and the plane orientation have been decided, topology tracking can be carried out in a computational space where the successive positions of the plane and the associated sampled vector values are aligned to satisfy the original configuration of [TWSH02]. The extracted paths and bifurcations often require low-pass filtering in post-processing to discard the short-term artifacts introduced by the choice of plane orientation. Additional details can be found in [TGK +04].

4 Vortex Breakdown Analysis

4.1 Background

In both the civil and military fields, the demand for shorter flight times and faster aircrafts has been a driving force behind research in recent years. Although it is not a recent development, together with supersonic speeds becoming more attractive, the delta wing

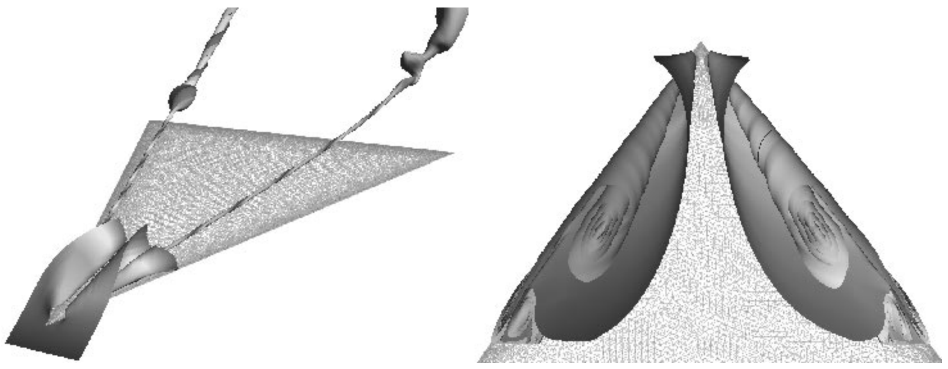


Figure 9: **Left:** Overview of the delta wing dataset with vortex creation at apex and the two primary vortices, breaking down differently. **Right:** Formation of primary, secondary and tertiary vortices at wing apex. Note how the shape of the tertiary vortex is strongly elliptic. For both images, the surface color coding depends on the starting curve location of individual streamlines (s-parameter).

design has found its way back into aircraft construction, as is demonstrated by a number of military aircrafts and the transatlantic passenger jet Concorde. An increased perception of security and the ever-shortening take-off and landing intervals on modern airports mandate a thorough examination of delta wing configurations with the aim of controllable flight even in exceptional flight situations, e.g. at a high angle of attack at subsonic speed. Furthermore, in military airflight, exceptional maneuverability is of prime importance. Due to this, the understanding of flow phenomena related to delta wing setups has become a major point of research activity. Among the most interesting of these is *vortex breakdown* due to its severe impact on flight stability. Delta wing designs differ from more conventional wing designs in that a substantial part of the lift on the wing is not only created by Bernoulli's principle (i.e. the velocity and hence pressure difference above and below the wing), but by large vortex systems above the wing. Breakdown of these vortices, i.e. the sudden loss of coherent vortical flow structures above the wing, can have catastrophic consequences, due to the sudden loss in lift and structural failure of the wing construction. While the phenomenon has been known for quite some time, there is still a lack of good theories of its genesis. Numerical research can be extremely helpful in allowing to study the occurrence of vortex breakdown in computer simulations. In the following, we describe a dataset from a simulation of a delta wing configuration that exhibits breakdown and apply a number of visualization techniques to study it. The dataset results from an unsteady computation of the Navier-Stokes equations in a volume surrounding a delta wing. As the simulation progresses, the angle of attack increases. The data is given on an unstructured adaptive-resolution grid with about 12 million cells and includes 1000 time steps. This dataset poses a significant challenge for visualization algorithms, owing to both size and numerical resolution issues.

4.2 Formation of Vortex Systems

In order to verify the correctness of the simulation and get an insight into the vortex systems above the wing, the extraction of vortices above the wing surface is an important task. The vortices above the wing actually form two symmetric vortex systems consisting of primary, secondary and tertiary vortices each.

Figure 9 (left) provides an overview showing the basic configuration. A stream surface started just below the wing apex illustrates the flow of air going over the edges and rolling up into the two primary vortices. The right image provides a view of the same surface from an opposing angle, revealing the three vortices on each side of the wing. It is interesting to note that while the primary and secondary vortices are almost circular in shape, the tertiary vortex is extremely elliptic. In the left image, another pair of stream surfaces wraps around the primary vortex core lines and exhibits spiraling behavior, as illustrated by proper color coding. About two thirds along, the coherent motion of these surfaces is interrupted and replaced by a bubble shaped structure. Despite the general symmetry of the dataset, the vortex breakdown it exhibits is asymmetric. On the (in direction of flight) right side, a so-called *breakdown bubble* forms, while the left-side structure looks chaotic.

The stream surfaces make for a good illustration of the major phenomena in the dataset. Since the vortex axes move over time, the seeding curves for the streamlines need to be determined manually for every time step, which is tedious. Automatic vortex core line extraction can help to some extent. Figure 10 (left) shows the results of an application of the Sujudi-Haimes algorithm (dark lines). The results are of mixed quality in that they include the vortex systems as well as some false positives. Using stream surfaces and the vortex surface technique detailed above, it is possible to discard the false positives, extract smooth vortex core lines (magenta lines) and obtain good vortex regions for each of the three vortices (red, green and blue surfaces). The right image shows a closeup. The separation between the individual vortices is excellent, and the elliptic shape of the tertiary vortex is extracted well.

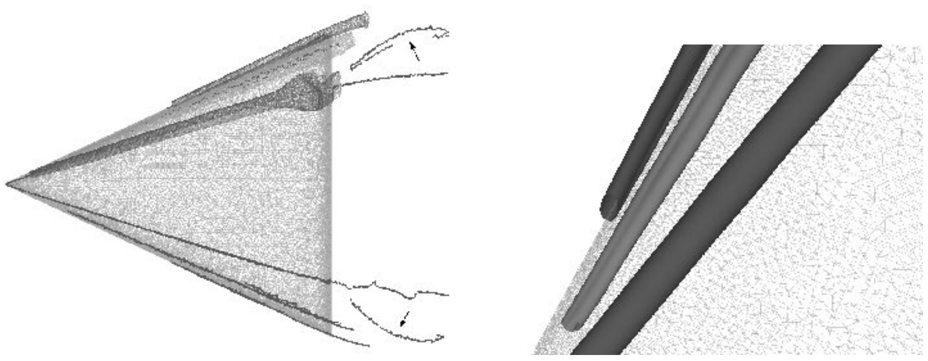


Figure 10: Vortices as extracted by the vortex surface technique. False positives (indicated by arrows) are reliably discarded. The closeup (right image) shows clean separation between the three vortices that form the vortex system. Note the strongly elliptic shape of the tertiary vortex (red).

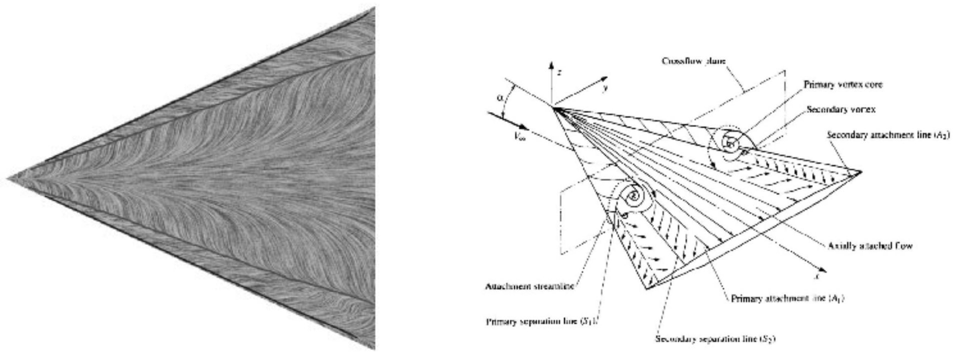

Crossflow plane
Primary vortex core
Secondary vortex
Secondary attachment line (A_2)
z
y
α
V_∞
Axially attached flow
x
Attachment streamline
Primary attachment line (A_1)
Primary separation line (S_1)
Secondary separation line (S_2)

Figure 11: **Left:** Separation and attachment lines in the wing shear flow as extracted with the method from [TGB+04]. **Right:** Illustrative sketch of the three-dimensional flow structure above a delta wing and the resulting shear flow structure on the wing (from [Dal83]).

To further understand the relation between the different vortices in each system, we have applied cutting-plane topology. The resulting structures are shown in Figure 13. The left image gives an overview of the structures that are revealed when the cutting plane travels along the symmetry axis of the wing. Although it is not orthogonal to the vortical structures, a good overall picture of the flow situation results. The primary vortex cores can be identified as the paths of critical points as the plane travels. The middle and right images show a closeup of the vortex system structure. More detail is extracted as the plane travels orthogonal to the primary vortex core. The interaction of the different vortices can be inferred from the topology. Effectively, the visualization of the vortex system is here reduced to two-dimensions, where it is much easier to comprehend. The separation in the wing shear flow appears as a natural part of the vortex system (individual vortices indicated by green arrows) and is extracted as the paths of saddle critical points close to the wing surface (red arrow in the right image). This *primary separation* is essentially the boundary of the influence regions of the primary and secondary vortices.

The separation and attachment structures can also be extracted directly, although with some difficulty. Figure 11 (left) shows results obtained using the approach described in Section 2. The primary separation is clearly visible. Together with the topological visualizations presented before, a complete picture of the complex vortex dynamics above the delta wing can be obtained and compared to a theoretical model (cf. [Dal83]). Although it is not of direct use in the analysis of vortex breakdown, it is of great value with respect to a validation of simulation results. In the next section, we focus on a direct visualization of the breakdown structure.

4.3 Structure and Evolution of the Breakdown Bubble

The complicated structure of the breakdown bubble presents a serious challenge for established visualization techniques. Since it is three-dimensional in nature, naive geometric

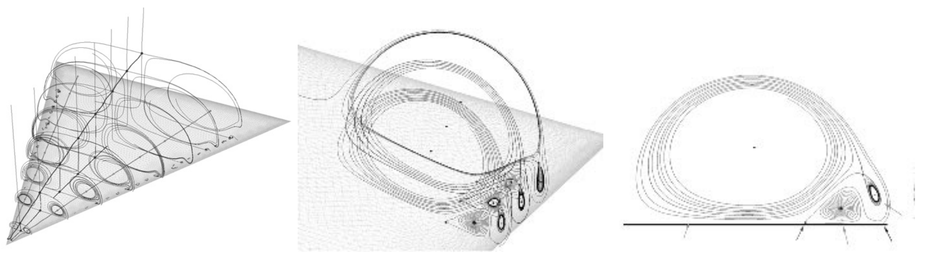


Figure 13: Application of cutting-plane topology to a delta wing simulation. **Left:** Plane travels along the wing symmetry axis, revealing the primary vortices and extracting their core lines as spiral-type critical point paths. **Middle:** Plane travels along the primary vortex core line. The full vortex system is visible. **Right:** A single slice (from the middle image) allows a detailed observation of the structural interaction between different vortices (marked by green arrows) and the corresponding separation and attachment (see also Fig. 11) on the wing surface (blue and red arrows).

visualizations such as streamlines suffer from issues of spatial perception. Stream surfaces can deliver better images here by providing a surface primitive that helps in understanding the three-dimensional structure of the flow. Fig. 15 provides an example. While the stream surface completely wraps the breakdown bubble, inner structure is easilyrevealed by applying a clipping plane. The observed motion is of a recirculation-type, overlaid by a simultaneous rotation around the original vortex axis.

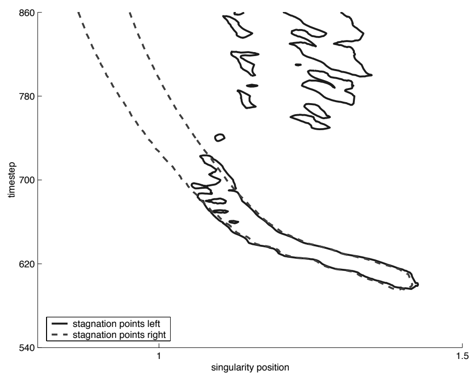


Figure 12: Structural graphs of left (blue) and right (red) side breakdown bubble evolution. While both sides start out almost identically, the left side deteriorates into chaotic breakdown and shows several rapidly oscillating bubbles in later time steps.

To better identify the recirculation zone, cutting-plane topology is an ideal tool as it allows to discard the superposed rotation. In this case, the cutting-plane rotates on the vortex axis. The recirculation can easily be identified by a closed, strongly curved vortex core (cf. Fig. 14) that appears as a spiral-type critical point path in the cutting-plane topology. It is interesting to note that these recirculation vortices are very hard to extract using conventional schemes due to their strongly curved nature. The right image shows three recirculation zones, hinting at several occurrences of vortex breakdown of the primary vortex.

Although the cutting-plane topology is immensely useful in the analysis of the breakdown bubble flow structure, it cannot provide an understanding of the dynamic of the flow in this case since it is essentially limited to a single time slice. Topological methods can still be useful in this context. It has been known that the occurrence of a breakdown bubble is accompanied by stagnation points in the flow, i.e. critical points of the flow vector

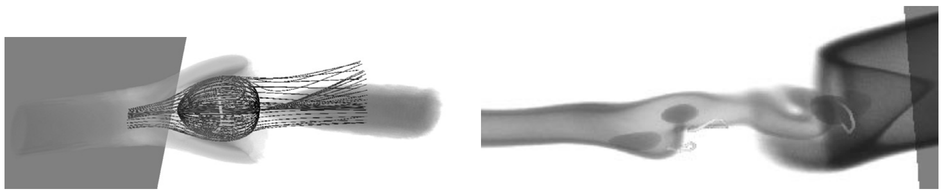


Figure 14: Application of cutting-plane topology to vortex breakdown analysis. Visualization of topology is enhanced by volume-rendered isosurfaces of velocity magnitude. **Left:** Right-side breakdown bubble. Structure is revealed through cutting-plane topology (plane revolving on vortex axis). The recirculation core is extracted as a spiral-type path (yellow). **Right:** Left-side staggered breakdown. Although a breakdown bubble is not discernible, several recirculation zones are extracted (yellow), hinting at multiple breakdown bubbles.

field, that lie on the vortex axis and essentially "delimit" the bubble. While these critical points are of saddle-type, topological visualization of the separating surfaces is essentially equivalent to a direct application of stream surfaces (Fig. 15). It is interesting, however, to apply critical-point tracking in this context. Although the resulting visualization is essentially four-dimensional (the critical points move in three-dimensional space over time), it can be reduced to two dimensions by observing that the movement of the stagnation points is limited to the axis of the primary vortex. The benefit of this procedure is twofold: First, the resulting structural graph allows to infer the structural evolution over all time steps. Second, since the dataset under consideration is asymmetric, it provides a simple and effective means to compare the evolution on both sides (Fig. 12). Comparing these results with those from Fig. 14 confirms that the left-side breakdown is chaotic (several small rapidly oscillating bubbles).

5 Evaluation of Flow in Engine Components

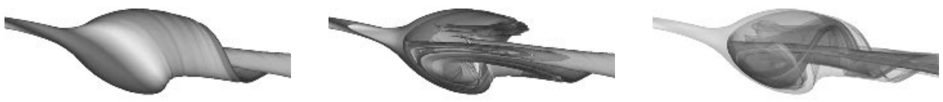


Figure 15: A stream surface illustrating the flow structure of the breakdown bubble in the delta wing dataset. While the opaque rendering (left) fails to provide insight, application of a clipping plane (middle) or transparent rendering (right) details the internal flow structure, essentially consisting of an asymmetric recirculation zone rotating around the original vortex core line.

With the general progress of state-of-the-art CFD simulations, the discipline of engine design is made accessible to both numerical simulation and visualization of the resulting datasets, allowing for rapid testing of engine designs. In the following we give an example of the application of topological and feature-based visualization methods for the analysis of prototype simulations.

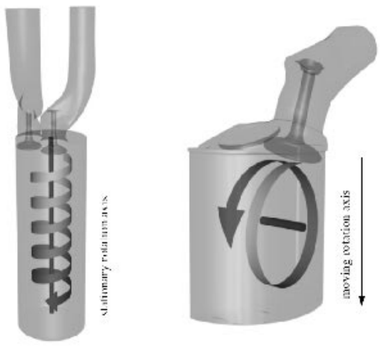


Figure 16: **Left:** Stable, circulating flow pattern in a diesel engine designated as *swirl motion*, with the cylinder axis as the axis of rotation. The flow enters tangentially through the intake ports. **Right:** Transient tumble motion in a gas engine. The axis of motion moves as the cylinder expands downwards and stays halfway between the top cylinder wall and the piston head at the bottom.

Among the many design goals of combustion engines, the mixing process of fuel and oxygen occupies an important place. If a good mixture can be achieved, the resulting combustion is both clean and efficient, with all the fuel burned and minimal exhaust remaining. In turn, the mixing process strongly depends on the inflow of the fuel and air components into the combustion chamber or cylinder. If the inlet flow generates sufficient kinetic energy during this valve cycle, the resulting turbulence distributes fuel and air optimally in the combustion chamber. For common types of engines, near-optimal flow patterns are actually known and include, among others, so-called swirl and tumble motions. We show two examples of simulation datasets showing each of these two types of flow patterns (henceforth termed “swirl-motion” and “tumble motion”). The basic geometries of the datasets and the respective desired motion patterns are shown in Figure 16.

Diesel Engine

This simulation is the result of a the simulation of steady charge flow in a diesel engine, based on a stationary geometry, resulting in a simple and stable flow. The main axis of motion is aligned with the cylinder axis and is constant in time. The spatial resolution of the single time step is high with a total of 776,000 unstructured cells on an adaptive resolution grid.

Gas Engine

This dataset results from an unsteady simulation of the charge phase of a gas engine. As the piston moves down, the cylinder volume increases by an order of magnitude and the fuel-air mixture entering the cylinder is drawn into a gradually developing tumble pattern. The overall motion is highly transient and unstable. Both spatial and temporal resolution are relatively low, with the data given on 32 time steps and the grid consisting of roughly 61,000 unstructured elements at the maximum crank angle.

For both datasets, the simulation results are given in the form of a vector field defined in the interior of the respective cylinder geometries. As is quite common in CFD simulations,

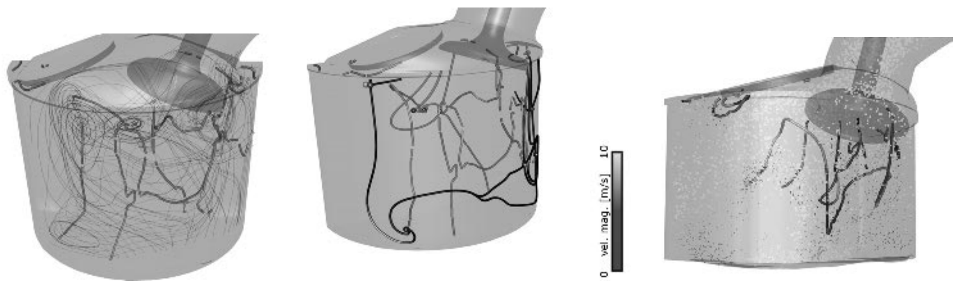


Figure 17: Application of cutting-plane topology and boundary topology to the gas engine dataset. **Left:** Cutting-plane topology provides a good overview of the overall motion pattern. Spiral-critical point paths (green) indicate rotation centers. **Middle:** In combination with boundary topology, interactions of boundary and volume flows are visible. Spiral critical points occur on the boundary where vortices intersect it. **Right:** Rotational centers enable an enhanced interpretation of conventional particle visualizations. Particles are color coded according to velocity magnitude. It is visible how the rotation centers capture particles in small-scale rotations.

the flow is required to vanish on the domain boundary (the so-called *no-slip boundary condition*) in order to correctly model fluid-boundary friction. Nevertheless, values on the boundary of the domain are easily inferred by e.g. extrapolation of volume values next to the boundary. We remark that in classical automotive engineering analysis, visualization is rarely performed for volume or boundary data but instead on two-dimensional slices. The main visualization goal in these cases is is the extraction and visual analysis of the swirl- and tumble-motion patterns. Unlike the previous dataset, in these datasets methods for the extraction of separation and attachment lines could not be applied since the resolution of the boundary flow field is too low. However, in our experiments we found that boundary topology is successful in extracting separation and attachment structures in the boundary fields due to the existence of critical points. In combination of with cutting plane topology, it is an effective approach to the extraction of swirl or tumble patterns and gives a good impression of the mutual influences of boundary and volume flows. Figure 17 illustrates this in a single time-slices of the gas engine dataset. From these images, one can recognize that the overall desired tumble motion actually consists of three independent rotational centers that together form the tumble motion. Since one of these vortices is rotating opposite to the other two, the tumble motion is much weaker than expected, and the design must be improved. Here, the axis for the cutting-plane approach is quite naturally parallel to the desired tumble axis, discarding patterns of motion that are not considered important for this application.

For the diesel engine (where a swirl-type pattern is desired), topological methods can provide results of similar quality, especially in combination with other techniques. Since the topological visualizations are mostly sparse in the sense that they provide a concise line-type depiction of the flow structure, it makes sense to combine them with feature extraction techniques that create a dense visualization (such as the λ_2-criterion) or other visualization techniques. Figure 5 gives several examples of hybrid visualizations of this type. The combination of boundary and volume visualizations gives a good understanding of the general nature of the flow. Again, we find that the overall swirl pattern is a combination of

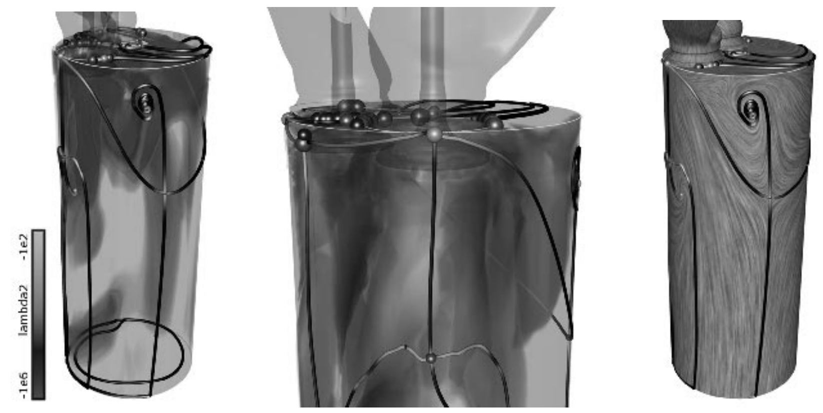


Figure 18: Visualization results for the diesel engine. Interaction of volume and boundary flows can be observed where separatrices on the boundary indicate a separation between two vortices in the volume close to the boundary. **Left:** A combination of boundary topology with a volume rendering of λ_2. The transfer function is chosen to indicate rotation strength. **Middle:** Here, the transfer function is chosen to represent the direction of the rotation (red vs. blue). Counter-rotating vortices close to the cylinder top take away kinetic energy from the formation of the main swirl pattern. **Right:** Boundary topology laid over a LIC image of the boundary flow.

several smaller vortices. There is only one large but weak vortex extending all the way to the bottom of the engine cylinder. The achieved pattern is therefor suboptimal.

For both the gas and diesel engines, the presented visualizations can be constructed without user assistance. This guarantees that visualizations are comparable between different simulation datasets of the same type, an important property when using visualization as a design analysis tool.

References

[CL93] B. Cabral and L. C. Leedom. Imaging Vector Fields Using Line Integral Convolution. In *Proc. 20th Annual Conference on Computer Graphics and Interactive Techniques*, 1993.

[Dal83] U. Dallmann. Topological Structures of Three-Dimensional Flow Separations. Technical Report 221-82 A 07, Deutsche Forschungs- und Versuchsanstalt fuer Luft- und Raumfahrt, 1983.

[GTS04] Christoph Garth, Xavier Tricoche, and Gerik Scheuermann. Tracking of Vector Field Singularities in Unstructured 3D Time-Dependent Datasets. In *Proceedings of IEEE Visualization*, pages 329–336, October 2004.

[GTSS04] C. Garth, X. Tricoche, T. Salzbrunn, and G. Scheuermann. Surface Techniques for Vortex Visualization. In *Proceedings Eurographics - IEEE TCVG Symposium on Visualization*, 2004.

[JH95] Jinhee Jeong and Fazle Hussain. On the Identification of a Vortex. *Journal of Fluid Mechanics*, pages 69–94, 285 1995.

[JMT02] M. Jiang, R. Machiraju, and D. Thompson. A Novel Approach to Vortex Core Detection. In *Data Visualization 2002*, pages 217 – 226, Aire-la-Ville, Sitzerland, 2002. Eurographics Association.

[KHL94] D. Kenwright, C. Henze, and C. Levit. Feature Extraction of Separation and Attachment Lines. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):135–144, 1994.

[Lug96] H. J. Lugt. *Introduction to Vortex Theory*. Vortex Flow Press, Inc., 1996.

[OK97] A. Okada and D. L. Kao. Enhanced Line Integral Convolution with Flow Feature Detection. In *Proceedings of IS&T/SPIE Electronic Imaging*, pages 206–217, 1997.

[PR99] R. Peikert and M. Roth. The Parallel Vectors Operator - a Vector Field Visualization Primitive. In *IEEE Visualization Proceedings '00*, pages 263 – 270, 1999.

[PVH$^+$02] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. Feature Extraction and Visualization of Flow Fields. In *Eurographics 2002 State-of-the-Art Reports*, pages 69–100, 2–6 September 2002.

[RP98] M. Roth and R. Peikert. Higher-Order Method For Finding Vortex Core Line. In *IEEE Visualization Proceedings '98*, pages 143 – 150, 1998.

[SH95] D. Sujudi and R. Haimes. Identification of Swirling Flow in 3D Vector Fields. Technical Report AIAA Paper 95–1715, American Institute of Aeronautics and Astronautics, 1995.

[ST05] G. Scheuermann and X. Tricoche. Topological Methods for Flow Visualization. In C.D. Hansen and C.R. Johnson, editors, *The Visualization Handbook*, pages 341–356. Elsevier, 2005.

[TGB$^+$04] Xavier Tricoche, Christoph Garth, Tom Bobach, Gerik Scheuermann, and Markus Rtten. Accurate and Efficient Visualization of Flow Structures in a Delta Wing Simulation. In *Proceedings of 34th AIAA Fluid Dynamics Conference and Exhibit*, June 2004. AIAA Paper 2004 – 2153.

[TGK$^+$04] Xavier Tricoche, Christoph Garth, Gordon Kindlmann, Eduard Deines, Gerik Scheuermann, Markus Rtten, and Charles Hansen. Vizualization of Intricate Flow Structures for Vortex Breakdown Analysis. In *Proceedings of IEEE Visualization*, pages 187–194, October 2004.

[TGS03] Xavier Tricoche, Christoph Garth, and Gerik Scheuermann. Fast and Robust Extraction of Separation Line Features. In *Proceedings of the Dagstuhl Scientific Visualization Seminar*, 2003. to appear.

[TS03] H. Theisel and H.-P. Seidel. Feature flow fields. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 141–148, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[TWSH02] X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen. Topology Tracking for the Visualization of Time-Dependent Two-Dimensional Flows. *Computers & Graphics*, 26(2):249 – 257, 2002.

[WS01] Thomas Wischgoll and Gerik Scheuermann. Detection and Visualization of Closed Streamlines in Planar Flows. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):165–172, 2001.

[WS02] Thomas Wischgoll and Gerik Scheuermann. Locating closed streamlines in 3D vector fields. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002*, pages 227–ff, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

Clifford Pattern Matching for Color Image Edge Detection

Michael Schlemmer, Hans Hagen
TU Kaiserslautern
Computer Science Department
D-67653 Kaiserslautern, Germany
{schlemmer, hagen}@informatik.uni-kl.de

Ingrid Hotz, Bernd Hamann
University of California, Davis
Institute for Data Analysis and Visualization
Davis, CA 95616
{ihotz, hamann}@ucdavis.edu

Abstract: Feature detection and pattern matching play an important role in visualization. Originally developed for images and scalar fields, pattern matching methods become increasingly interesting for other applications, e.g., vector fields. To apply pattern matching to vector fields the basic concepts of convolution and fast Fourier transform (FFT) have to be generalized to vector fields. A formalism supporting an elegant generalization of these concepts is provided by the Clifford Algebra, originally developed for describing geometry and geometric operations. We discuss an application of the Clifford Pattern Matching (CPM). We apply CPM to images for "Clifford Color Edge Detection" (C^2ED), an approach for detecting edges and other features in color images. The basic idea is to treat color value tripels as vectors and apply the pattern matching algorithm to the resulting vector field. We introduce vector-valued filters for edge detection and present results.

1 Introduction

Today, large amounts of data are produced by simulations and experiments. Pattern recognition methods have become more important as essential information is mostly hidden "in-between less important data". A good visualization should be capable of highlighting important facts, pointing out key features.

Recently, a new method for visualizing vector fields was introduced based on pattern matching methods. A convolution operator for pattern recognition was constructed and applied to uniform vector field data, see Heiberg et al. [HEWK03], and Ebling and Scheuermann [ES03]. The latter method is based on Clifford algebra.

In signal processing it is common to filter data in frequency domain, as the convolution operation is very expensive and is reduced to a multiplication in frequency domain. To devise a similar method for vector fields a continuous and discrete Fourier transform for multi-vector field data by using a Clifford algebra approach [Sch04, ES05] was adapted. *Multi-vectors* are elements of the Clifford algebra representing a combined set of complex-valued vector and scalar data. We implemented the discrete Clifford Fourier Transform (CFT) using the FFT for regular grids.

Besides the application to flow vector data, there are various other possibilities. One of these options is the application of these vector methods to color image edge detection. Segmentation based on edge detection is a vital part of current classification systems. These systems are used in various application areas, including astronomy, medicine, robots, etc. In most cases, edge detection is performed on gray-scale images. Therefore, there are many common methods for this task, which are for example discussed in [Jai89], for examples. One of the most important techniques was developed by Canny [Can86], which is discussed in section 2.1.2.

There are also approaches for detection of edges using color information, i.e., the component-wise Canny edge detection applied to each of the RGB color channels. In order to improve color edge detection, we had the idea of regarding the RGB triple of colors as a vector, therefore the color image as vector field. Thus, we are able to apply our Clifford Fourier transform and Clifford convolution to find features.

As the vector field is given on a planar 2d domain, a 3d convolution leads to a 3d result, being inappropriate for our needs. We decided to stay in a 2d domain and consider only a 2d vector field. This could be achieved by switching the color model from RGB to YUV and treat the luminance component (Y) as scalar and the chrominance components (UV) as a 2d vector field. Machuca and Phillips [MP83] proposed similar settings for their method. We found that using any of the filter masks we use for representing special topological features (rotation, convergence, and divergence) allows one to detect edges in a color image. The 2d Clifford algebra is especially suitable for this task, as all components fit perfectly into the mathematical concept.

2 Edge detection

For computer vision one of the basic goals is the recognition of objects in complex scenes. Given an image in digital form, a first step towards segmentation and recognition of objects can be the detection of edges. Using this information a segmentation in for- and background can be performed, and semantics can be assigned through the process of classification.

2.1 Edge detection in grey-scale images

In practice, edge detection is performed on grey-scale images. We focus on the 2d case and define a 2d-image (see Jähne [Jäh95]):

Definition 1 *A 2d-image is a discrete scalar function p on a rectangular grid having N_1 grid points in x-direction, N_2 in y-direction with distances $\Delta x, \Delta y$ and value set $\mathbb{W} \subseteq \mathbb{C}$:*

$$p : \{0, 1\Delta x, ..., (N_1 - 1)\Delta x\} \times \{0, 1\Delta y, ..., (N_2 - 1)\Delta y\} \to \mathbb{W}.$$

For pixels p_{ij} the following notation is used:

$$r_{ij} = \binom{i\Delta x}{j\Delta y} \quad p_{ij} = p(r_{ij}).$$

A 2d-image is defined by the collection $P = (N_1, N_2, \Delta x, \Delta y, \mathbb{W}, \{p_{ij}\})$.

Each grey-scale value is assigned to a scalar number $w \in \mathbb{W}$.

2.1.1 Filter operations

There exist many different possibilities for filters for image processing. We concentrate on the class of linear-and-shift-invariant (LSI) filters as their impulse response can be described as a neighborhood representation. For application of LSI filters the performed operation can be either convolution or correlation. While the correlation operator directly performs a pattern matching to a given filter on the given signal, the (in practice more important) convolution uses a mirrored version of the filter. As the neighborhood representation of a filter can also be regarded as an image, one can define (see Jähne[Jäh95]):

Definition 2 *Let g be a 2d image, h be a 2d filter. The discrete convolution $g * h$ is defined as*

$$(g * h)_{m,n} = \sum_{i=0}^{N_1-1} \sum_{j=0}^{N_2-1} h_{i,j} g_{m-i,n-j}.$$

Convolution is a computationally expensive operation. It is in practice simplified by transferring image and filter into frequency domain, where the convolution operation reduces to a multiplication. The result is then transformed back into spatial domain. As transformation between spatial and frequency domain the fast Fourier transform (FFT) can be used.

Through the application of filters different goals can be achieved. One special class of filters is used to smooth images to reduce various types of noise in an image. Examples for smoothing filters are box filters, binomial filters, and Gauss filters. Another class of filters was especially designed for edge detection. Those filters basically focus on gradients in an image. Examples for filters are the simple gradient filters (first order) and Laplace filter (second order). An optimized version of the gradient filter is the Sobel operator. The Marr-Hildreth operator [MH80] is an enhancement of the Laplace operator. It combines a noice-reducing Gauss filter with a Sobel edge detection filter.

2.1.2 Canny edge detection

Canny developed an optimal edge detector for grey-scale images [Can86]. His method is similar to the Marr-Hildreth approach, see [MH80]. First, a Gaussian is applied for smoothing, then the actual edge detection is performed. The Canny algorithm uses four gradient filter masks to detect horizontal, vertical and diagonal edges, as an edge in an image can be oriented arbitrarily. It is assumed that important edges form continuous traceable lines. The application of thresholding results in a binary image containing line segments that represent edges. The major problem of this approach is finding an appropriate threshold. Another parameter affecting the result is the size of the Gaussian blur filter applied in the beginning. With an appropriate choice of these parameters for a specific grey-scale image, ”optimal” results can be obtained.

2.2 Extensions to color edge detection

As most camera-recorded images contain colors, converting them to grey-scale images comes with a big loss of information. Therefore, several publications focus on the use of colors for segmentation and image understanding. The first color image edge detectors were presented by Robinson [Rob77] and Nevatia [Nev77]. Koschan applied the standard techniques to the three color channels and performed a comparative study [Kos95]. Tao and Huang used cluster analysis to determine better thresholds [TH97]. Most methods apply the well-known grey-scale methods to the three RGB color channels seperately. The most successful method is again the Canny color edge detection scheme using the Canny edge detector for the RGB channels. Furthermore, other authors have proposed the use of different color spaces, e.g., Weeks and Myler [WM95] as well as Machua and Phillips [MP83]. Among others, some of the mentioned methods have recently been reviewed by Koschan and Abidi [KA05].

3 Clifford algebra in two dimensions

Clifford algebra can be understood as an extension of complex numbers to vectors:

Definition 3 *Let $\mathbb{E}^2$ be the $I\!R$ vector space with basis $\{e_1, e_2\}$. The **Clifford algebra** $\mathcal{G}^2$ is the real 2^2-vector space with basis*

$$\{1, e_1, e_2, e_1 \wedge e_2\},$$

where

1. $1e_k = e_k,\ k = 1, 2,$
2. $e_k e_k = 1,\ k = 1, 2,$ *and*
3. $e_k e_l = -e_l e_k,\ k \neq l.$

The elements of a Clifford algebra are called multi-vectors. For the 2d case, they are shown in the following table:

name	grade	dimension	basiselements
scalar	0	1	1
vector	1	2	e_1, e_2
bivector	2	1	$e_1 e_2$

A multi-vector consists of a scalar, a 2d vector and the so-called bivector (pseudo-scalar). Regarding the Clifford algebra as an extension of complex numbers, it can be regarded as a tuple of two complex numbers, one having the scalar as real part and the bivector as imaginary part, the vector having the e_1 component as real and the e_2 component as imaginary part. For detailed information on Clifford algebra, see Hestenes and Sobczyk [HS99].

4 Pattern matching using Clifford algebra

A first definition of a convolution operator was proposed by Heiberg et al. [HEWK03]. It uses the inner (scalar) product for vectors. Ebling and Scheuermann [ES03] defined the convolution using the Clifford product consisting of inner and outer products. In addition, for efficient calculation a fast Fourier transform for Clifford algebra was introduced [Sch04, ES05]. These techniques enabled an image processing-like detection of patterns in vector fields. Patterns can be defined as vector valued LSI filters as neighborhood representation. Figure 1 illustrates the process of pattern matching applied to vector fields using vector-valued masks and Clifford convolution.

5 Clifford color edge detection

5.1 General idea

As color images can be represented as vector fields using the color components as vector components, the general idea of Clifford color edge detection is using pattern matching for vector fields for the detection of edges in color images. The given 3d vectors (RGB) are on a 2d grid. The application of 3d pattern matching would lead to a 3d similarity map as result, being an inappropriate representation as our result has to be projected back into two dimensions. For that reason, we decided to handle the luminance and chrominace part separately. This can be achieved by transferring an RGB image to a corresponding YUV image. The Y channel represents the luminance (grey-scale image), while the chrominance is represented by the 2d vector field UV. After filtering, we obtain two scalar similarity maps that can be combined to a final similarity map representing edges in the color image. The grey-scale part (Y) can be treated exactly as done in the common approaches, and the pattern matching applied to the color part (UV) adds additional information.

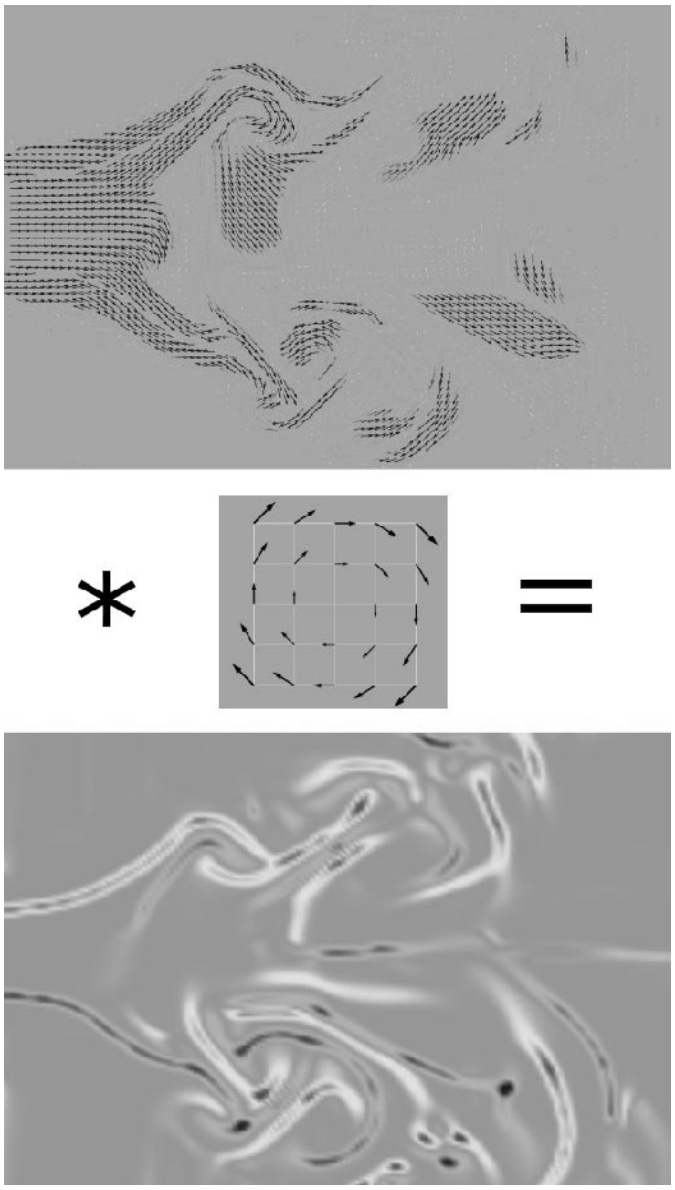


Figure 1: Pattern matching applied to vector fields: a fluid flow dataset (swirling jet entering fluid at rest) is undergoing a convolution operation with a rotation filter mask. The result is a scalar map of ”similarities”, showing regions of high vorticity.

5.2 Data structure

The Clifford multi-vectors in the 2d case are suitable for this approach. We can assign the values as follows:

name	grade	dimension	values
scalar	0	1	Y
vector	1	2	U, V
bivector	2	1	0

The Y component becomes the scalar part of our multi-vector at each grid position. The vector component is represented by UV, using U as real part with basis e_1 and V as imaginary part with basis e_2. The imaginary scalar part with basis $e_1 \wedge e_2$ is set to zero, as there is no imaginary component for the scalar part.

5.3 Choice of patterns

For the scalar part the choice of filters is simple, as it reduces to grey-scale edge detection. For the color component UV, the main question for using vector pattern matching is what patterns to search for. Since the topologically interesting features like rotation, divergence, and convergence are related to the derivation operator, these patterns are a reasonable choice. We used four different vector patterns for filtering the UV part. They are presented in Figure 2.

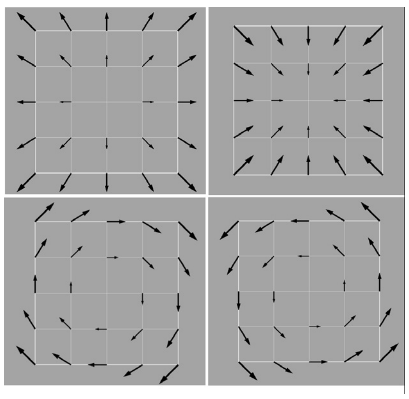


Figure 2: Four different vector pattern mask used as filters for the UV part: divergence (upper left), convergence (upper right), clockwise rotation (lower left) and counter-clockwise rotation (lower right).

5.4 Summary

We did not mentioned, that a blur filter could be applied prior to edge detection, as done in Marr-Hildreth and Canny edge detection schemes. It could be applied separately to all color channels, before converting an image to YUV, as well as to the YUV representation, since the conversion is a linear operation. As mentioned in section 2.1.2, an appropriate choice for these filters could enhance the final result.

The complete process of the Clifford color edge detection process is illustrated in Figure 3. We obtain two resulting similarity values. One real similarity value for the scalar part and a complex one for the vector part. Computing the magnitude of the complex value, the resulting two real values can either be combined or seperately further processed according to Canny's algorithm.

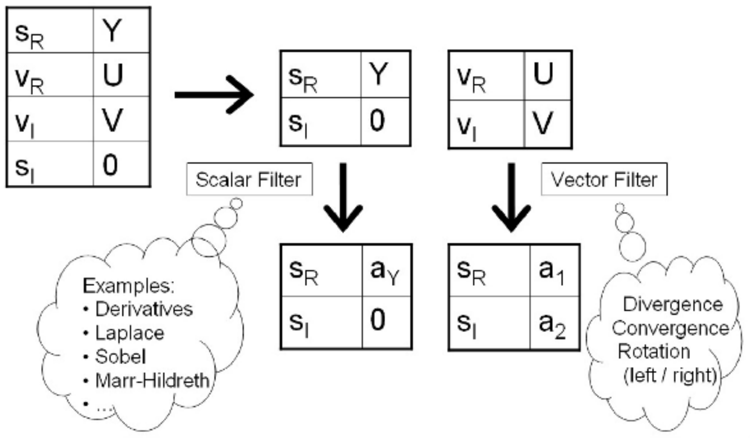


Figure 3: Illustration of Clifford color edge detection. Image given in YUV color space in a multi-vector structure, grey-scale edge detection performed as usual, UV part filtered with vector pattern matching. Result is a multi-vector of similarities.

6 Results

Given a color image, our algorithm computes two sets of similarity values: a set of real values and a set of complex values. The real set describes a fuzzy representation of the edges in the grey-scale image, while the magnitude of the complex value indicates edges in the color part of the image. The exact structure of the complex value depends on the used filter, but the unsigned magnitude of the complex similarity values turned out to be equal for all four filters that we applied (two rotation pattern, a convergence, and a divergence pattern, see Figure 2).

A user can adjust the binary threshold and the ratio of grey-scale and color contribution. It is possible to configure the identified edges as desired. We have chosen manual adjustment, as the automatic thresholding problem known from Canny's algorithm still applies to our case. Using the YUV model, we allow a user to control luminance and chrominance separately. This representation is more intuitive than a representation in RGB space, as the human sensory system processes luminance seperately from chrominance.

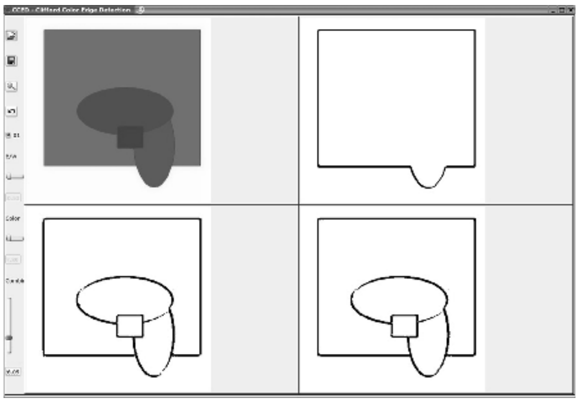


Figure 4: Example where grey-scale edge detection fails, while color edge detection succeeds. Upper left: (contrast enhanced) original image, upper right: result of a grey-scale edge detection, lower left: the color edge detection, lower right: the combined edge detection, showing all edges in the original image.

We have processed various images using this method. It generally performed better than simple grey-scale edge detection. Figure 4 shows an example where a grey-scale recognition fails. Examples for real-world image data are shown in Figures 5 and 6. We illustrate the enhancement using our color approach by comparing with the common grey-scale algorithm. Our algorithm turned out to be equal in performance to an optimally config-

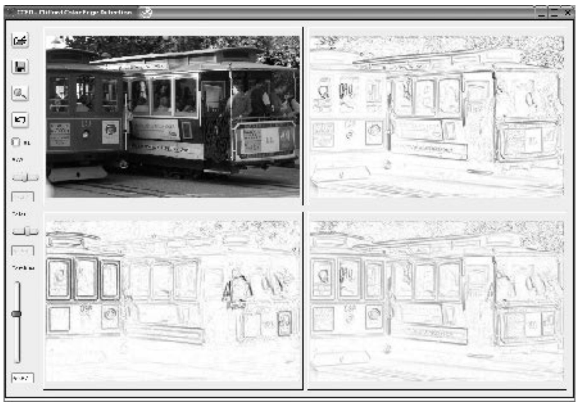


Figure 5: C^{2}ED applied to example image, resulting in a fuzzy representation of the similarity values. Upper left: original, upper right: Y filtered, lower left: UV filtered, lower right: weighted combination of Y and UV parts

ured color edge detection method using the RGB color model. The transformation from RGB into YUV space does not change the result when all component results are weighted equally. The vector-based approach in Clifford algebra for handling the UV part is an operation applied to complex-valued scalars. Those can again be rewritten component-wise, splitting them into a real and an imaginary part. Rewriting the vector filter pattern (see Figure 2) in components yields the commonly used filters for edge detection for scalar images in both axis directions. The four different types of filters only differ in their algebraic signs for the real and imaginary values. Since in the final step of our algorithm the magnitude of the complex result is computed, the search pattern can be either one of the given ones to obtain the same result.

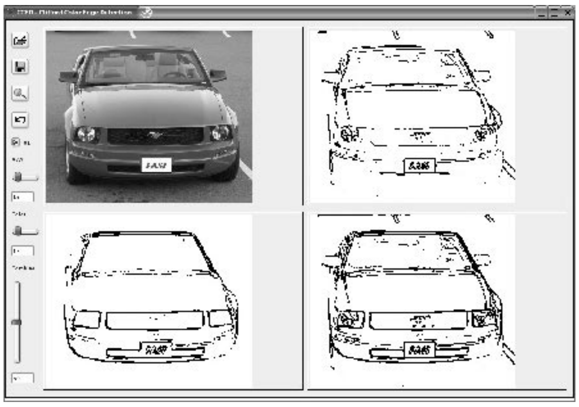


Figure 6: For this image of a car (upper left) thresholding was performed, resulting in a binary edge representation (upper right: Y, lower left: UV, lower right: combined). Results can be further improved using Canny's method.

7 Conclusions

We have presented a new method for color image edge detection for color images using Clifford algebra. We handle grey-scale data seperate from the color part of the image. The luminance part is handled using common methods, while the color part is filtered with a vector-valued filter. Those two approaches fit perfectly in the data structure of Clifford algebra's multi-vector setting. Our results have shown that this approach outperformes the grey-scale edge detection in most cases, since additional information is gained through the processing of the color part. However, it turned out to be equal in performance when compared to other color edge detection methods working component-wise on RGB images. For non-automatic detection, we implemented a framework that offers the possibility to adjust thresholds manually. For manual adjustment the YUV color model is more intuitive. Our algorithm can also be combined with the optimal Canny edge detector.

A remaining challenge is the threshold problem. There is still a need for an automatic method to determine a "good" threshold.

8 Acknowledgements

We would like to thank the members of the Visualization and Computer Graphics Research Groups at the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis, and at the University of Kaiserslautern, Germany as well as the members of our International Research Training Group (DFG IRTG 1311) for fruitful discussions, especially Christoph Garth, Younis Hijazi, and Oliver Rübel. We thank Wolfgang Kollmann, UC Davis, for providing the fluid flow data, and Manuel Heringer and Florian Morr for programming support for its visualization. The IRTG is supported by the German Research Foundation (DFG). This work was also supported in part by the National Science Foundation under contract ACI 9624034 (CAREER Award) and a large Information Technology Research (ITR) grant. We further gratefully acknowledge the support of the W.M. Keck Foundation provided to the UC Davis Center for Active Visualization in the Earth Sciences (CAVES).

References

[Can86] J. Canny. A Computational Approach to Edge Detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 8, pages 184–203, 1986.

[ES03] J. Ebling and G. Scheuermann. Clifford Convolution And Pattern Matching On Vector Fields. In *Proceedings of IEEE Visualization 2003*, pages 193–200, 2003.

[ES05] J. Ebling and G. Scheuermann. Clifford Fourier Transform on Vector Fields. *IEEE Trans. Vis. Comput. Graph.*, 11(4):469–479, 2005.

[HEWK03] E. Heiberg, T. Ebbers, L. Wigström, and M. Karlsson. Three-Dimensional Flow Characterization Using Vector Pattern Matching. *IEEE Trans. Vis. Comput. Graph.*, 9(3):313–319, 2003.

[HS99] D. Hestenes and G. Sobczyk. *Clifford Algebra to Geometric Calculus, A Unified Language for Mathematics and Physics*. Fundamental Theories of Physics. Kluwer Academic Publishers, Dordrecht, Boston, London, 1999.

[Jäh95] B. Jähne. *Digital image processing*. Springer-Verlag, Berlin, third edition, 1995.

[Jai89] A.K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[KA05] A. Koschan and M. Abidi. Detection and Classification of Edges in Color Images. *IEEE Signal Processing*, 22(1):64–73, January 2005.

[Kos95] A. Koschan. A comparative study on color edge detection, 1995.

[MH80] D. Marr and E. Hildreth. Theory of Edge Detection. *Proceedings of the Royal Society of London*, B207:187–217, 1980.

[MP83] R. Machuca and K. Phillips. Applications of Vector Fields to Image Processing. *IEEE Trans. Pattern Anal. Machine Intell.*, 5(3):316–329, May 1983.

[Nev77] R. Nevatia. A Color Edge Detector and Its Use in Scene Segmentation. *IEEE Trans. Syst. Man Cybern.*, 7(11):820–826, November 1977.

[Rob77] G. Robinson. Color Edge Detection. *Optical Engineering*, 16(5):479–484, September 1977.

[Sch04] M. Schlemmer. Fourier Transformation and Filter Design for Clifford Convolution. Master's thesis, 2004.

[TH97] H. Tao and T.S. Huang. Color Image Edge Detection Using Cluster Analysis. In *IEEE International Conference On Image Processing*, pages 834–836, 1997.

[WM95] A.R. Weeks and H.R. Myler. Edge Detection of Color Images Using the HSL Color Space. 2424:291–301, 1995.

Arrangements of Planar Curves

Younis Hijazi
International Research Training Group
University of Kaiserslautern
Computer Science Department
D-67653 Kaiserslautern, Germany
hijazi@iupr.org

Abstract: Computing arrangements of curves is a fundamental and challenging problem in computational geometry as leading to many practical applications in a wide range of fields, especially in robot motion planning and computer vision. In this survey paper we present the state of the art for computing the arrangement of planar curves, considering various classes of curves, from lines to arbitrary curves.

1 Introduction

The arrangement of planar curves is the decomposition of the plane into vertices, edges, and faces induced by the curves. It can be represented as a graph. The applications of arrangements include robotics, computer graphics, molecular modeling, and computer vision. Their study started with simple classes of geometric objects such as lines and - in current active research - tends to generalize to much more general classes such as algebraic objects of arbitrary degree or even completely arbitrary curves.

In this survey paper we present the current state of the art for computing the arrangement of planar curves. We first consider lines, which are already of great interest regarding the application side, for instance by dualizing problems involving points into line problems. We focus then on cubics - including the particular class of conics - before ending on latest results about algebraic curves and also more general curves.

In Section 2 we introduce several mathematical concepts which will be omnipresent allover the paper. Definition and motivation for arrangements are then provided in Section 3 which allows in Section 4 entering the heart of the topic, i.e. different techniques to compute the arrangement of planar curves. The studied classes are ordered with respect to their complexity: straight lines, cubics, algebraic curves, and arbitrary curves. In Section 5 we show the wide range of applications of arrangements in real world. Finally we conclude with a discussion and perspectives for on-going research.

2 Some definitions

Following are some mathematical definitions for several types of curves which are studied in this survey. For an introduction on algebraic curves, see [G85].

Algebraic curve An *algebraic curve* is an algebraic variety of dimension one. A planar algebraic curve can be represented as $P(x, y) := \sum_i \sum_j p_{ij} x^i x^j = 0$, where $p_{ij} \in \mathbb{R}$. The degree d of an algebraic curve is $d = max_{i,j}(i + j)$.

For instance, the algebraic curve defined by the equation $2x^3y + 5xy^2 - y - 1 = 0$ is of degree 4. We are now able to give the definitions of conics and cubics:

Conics and cubics A *conic (or conic curve)* is an algebraic curve of degree at most 2 and a *cubic (or cubic curve)* is an algebraic curve of degree at most 3.

Named for the French mathematician Pierre Bézier, a Bézier curve is a curved line defined by mathematical formulas. Mathematically, we can formulate this as follows:

Bézier curve Given $n + 1$ control points P_0, P_1, ..., P_n, the Bézier curve (of degree n) is defined by: $B(t) = \sum_{i=0}^{n} P_i b_{i,n}(t)$ with $t \in$ [0,1] and where $b_{i,n}$ are known as Bernstein polynomials.

3 Arrangement structures

Arrangement structures are of great interest in computational geometry as expressed in [H97] and [AS00] surveys. Early work focused especially on the arrangement of hyperplanes, with the particular case of $2D$ lines. Many efforts were put to extend the study to more general objects, i.e. conics, cubis, and even arbitrary algebraic curves, being useful on the application side by matching real world applications.

3.1 What is an arrangement?

Given a collection C of geometric objects, the arrangement $A(C)$ is the decomposition of $\mathbb{R}^d$ into connected open cells of dimensions 0, 1, ..., d induced by C. Considering a collection of curves in $2D$, we have 0-cells (vertices - which are the intersection points), 1-cells (edges), and 2-cells (faces). An arrangement can be represented as a graph where its nodes are the 0-cells and its edges, the 1-cells. The graph gives two main information: the geometry (position of the cells), and the topology (connectivity of the cells) of the collection of the considered objects.

3.2 Why arrangements?

There are many practical applications of arrangements which will be studied in Section 5 but at this stage we only give a flavour to motivate our interest in arrangement structures. For instance, by considering the simple class of lines, many problems involving points can be transformed into problems involving lines, thanks to a duality transform. The task of determining whether any three points of a planar point set are collinear could be determined in $O(n^3)$ time by brute-force checking of each triple. However, if the points are dualized into lines, then this reduces to the question of whether there is a vertex of degree greater than 4 in the arrangement, which can be computed in $O(n^2)$ time.

4 Computing $2D$ arrangements

For each considered class of curves we are first interested in counting the number of intersections. Indeed, the number of curves intersections will highly vary whether we consider lines or arbitrary algebraic curves. For n lines, we know that in a simple configuration we will have exactly $\frac{n(n-1)}{2}$ intersection points, which is also an upper bound for all possible configurations. For algebraic curves we can use Bezout's upper bound result which says that the number of intersection points of two algebraic curves of degrees respectively p and q is bounded by the product of the degrees, i.e. $d = pq$. For n algebraic curves, this implies a combinatorial complexity (counting the cells) both quadratic in n, the number of curves, and d, the degrees of the curves.

One often meets the term "sweep" when interested in computing arrangements. Usually, sweeping a planar arrangement means sweeping with a vertical line and updating the event points where curves start, end or cross. The sweep line doesn't necessarily need to be straight as shown in [EG89] by the use of a topological line. One can even think about the term "sweep" more generally, as for example adressed by Breuel in [B92]: Cass' algorithm in [Ca90] - called Critical Point Sampling (CPS) - is equivalent to a sweep of the arrangement generated by the feasible sets implied by all correspondences between model and image points, regarding computational geometry. Also, in addition to sweep line approaches, subdivision methods are emerging for the task of computing arrangements.

4.1 Lines

Arrangements of lines (more generally, hyperplanes) were already studied in the 19th century. Now - and since a long time - everything is known for the arrangement of straight lines. In particular, the arrangement of lines defined by rational numbers can be computed exactly, i.e. without any numerical error. Complete, exact, and efficient implementations for the arrangement of lines can be found in LEDA [M00].

As already mentionned, the combinatorial complexity for lines arrangement is $O(n^2)$. Due to $\log n$ time operations the "naive" overall time complexity for computing an arrangement of n lines is $O(n^2 logn)$ which can be reduced to $O(n^2)$ time by using a topological line sweep as in [EG89]. This result is proved to be the best one can achieve for lines.

4.2 Cubics

A lot of work has been carried to study arrangements of non-linear objects, starting with conics and cubics. As conics are special cases of cubics and as similar approaches are used for computing their arrangement, only cubics will be discussed in our study. Also as our concern remains on planar curves, arrangements of $3D$ quadrics (i.e. quadric surfaces) will be skipped, despite their importance in the literature. Conics and cubics are very interesting classes as we know such curves quite well from the mathematical point of view and they already provide a wide range of applications.

In [ESW02] and [EKLW04] the authors use a Bentley-Ottmann sweep-line algorithm [BO79] to compute the arrangement. Their algorithm in [EKLW04] is complete (handles all possible degeneracies), exact (provides the mathematically correct result), and efficient (in terms of complexity). At our knowledge it is the only result succeeding in both theoretical (exactness, completeness) and practical (efficient implementation) challenges for this class of planar curves.

4.3 (Semi-) Algebraic and Bézier Curves

There are several papers studying the arrangement of semi-algebraic, algebraic and Bézier curves and methods to succeed in this goal vary considerably. In [W03] Wolpert presents an approach that extends the Bentley-Ottmann sweep-line algorithm - used in [EKLW04] for cubics - to the exact computation of the topology of arrangements induced by non-singular algebraic curves of arbitrary degrees. This paper overcomes the problem of detection and location of tangential intersection points of two curves using only rational arithmetic, and extending the concept of Jacobi curves [W02]. The result is an output-sensitive algorithm.

A different approach - but also for computing the arrangement of semi-algebraic curves - is presented in [MS06] where a vertical sweep line is being used together with a module that computes approximate crossing points of the input curves. The authors provide an implementation for semi-algebraic curves based on numerical equation solver. The running time of their algorithm is $O(V logn)$ for n curves with N crossings and k inconsistencies where $V = 2n + N + min(3kn, n^2/2)$. One claim of the paper is that the algorithm performs much better than the best known published results (1000 cubics in 100 seconds, versus 2000 seconds).

In [Y06] Yap follows a new direction and presents the first complete subdivision algorithm for the intersection of two Bézier curves, possibly with tangential intersections. The adaptive algorithm uses a robust subdivision scheme based on geometric separation bounds, using a criterion for detecting non-crossing intersection of curves, and avoids manipulation of algebraic numbers and resultant computations.

In [WM06] Wintz et al. describe a new subdivision method to construct the arrangement of implicit planar algebraic curves and provide an incremental dynamic algorithm maintaining the solution of the problem as the input objects are inserted, without preliminary knowledge on the input data. The subdivision scheme is based on the bounding boxes of the input data - rather than using a classical hierarchical quadtree - keeping though advantage of quadtrees' adaptivity feature. The method combines the multivariate Bernstein's basis with Descarte's Law of Sign and uses an algebraic criterion to decide whether further subdividing a box. Experiments have been run on the algebraic modeling platform AXEL using the algebraic computation library SYNAPS. The technique proved reliable and can be extended to higher dimensions and different kinds of objects.

4.4 Arbitrary curves

Little work has been done for computing the planar arrangement of arbitrary planar curves as they are unpredictable objects and not well known mathematically. Nevertheless, there have been two attempts in this direction using a divide-to-conquer interval arithmetic-based approach. In [CdFC98] the authors are interested in determining the exact topological adjacency structure of the planar subdivision induced on a rectangle by a set of curves given in implicit form. They use a recursive method based on estimates provided by interval arithmetic, together with conditions guaranteeing the topological correctness of the arrangement.

In [HB06] the authors present a method to compute the planar arrangement of implicitly defined curves using an interval arithmetic-based recursive algorithm. In their method, interval arithmetic is used both for reliable numerical computations and as an integral part of the search. They use an adaptive subdivision of the domain: interval arithmetic is used to reliably classify each rectangle according to whether it is empty, contains a curve, or contains multiple curves and/or intersections. The resulting decomposition is used to construct a topologically well-characterized representation of the arrangement together with algebraic representations of the cell boundaries. The algorithm is applicable to the enumeration of the cells of arrangements of arbitrary collections of curves defined implicitly using interval arithmetic and generalizes to higher dimensional spaces.

5 Applications of arrangements

This section is inspired from the survey of Agarwal/Sharir [AS00], as being the reference for this topic. The reader is invited to consult the survey for details. The following non-exhaustive list of applications involves planar arrangements: range searching, transversals, geometric optimization (slope selection, distance selection, segment center, minimum-width annulus, geometric matching, center point, Ham sandwich cuts) and robotics. We develop some of them.

5.1 Range searching

Geometric range searching [AE98] is the problem of:

> Preprocessing a set S of n points in $\mathbb{R}^d$, so that all points of S lying in a query region can be counted quickly.

Range searching has important applications in Geographic Information Systems (GIS), computer graphics, spatial databases, and time-series databases. Range searching and arrangements are strongly related as point location in hyperplane arrangements can be used for range searching.

Mathematically, we can formulate this as follows: By defining the dual of a point $p = (a_1, ..., a_d)$ to be the hyperplane $p^* : x_d = -a_1x_1 - ... - a_{d-1}x_{d-1} + a_d$, and the dual of a hyperplane $h : x_d = b_1x_1 + ... + b_{d-1}x_{d-1} + b_d$ to be the point $h^* = (b_1, ..., b_d)$, then p lies above h if and only if the hyperplane p^* lies above the point h^*. Hence, halfspace range searching has the following equivalent "dual" formulation: Preprocess a set Γ of n hyperplanes in $\mathbb{R}^d$ so that the hyperplanes of H lying below a query point can be reported quickly. Using the point-location data structure for hyperplane arrangements provided in [Ch90], the level of a query point can be computed in $O(logn)$ time using $O(\frac{n^d}{log^d n})$ space.

5.2 Geometric optimization

In this subsection we focus on geometric optimization problems and how they are related to arrangements. As we will see, the area of geometric optimization is a natural extension and a good application area of the study of arrangements. We examine a sample of them starting with slope selection.

- slope selection:

  > Given a set S of n points in $\mathbb{R}^2$ and an integer k, find the line with the kth smallest slope among the lines passing through pairs of points of S.

  If points of S are dualized to a set Γ of lines of $\mathbb{R}^2$, the problem becomes that of computing the kth leftmost vertex of the arrangement $A(\Gamma)$.

- minimum-width annulus:

  > Compute the annulus of smallest width that encloses a given set of n points in the plane.

  This problem arises in fitting a circle through a set of points in the plane, though involving arrangements.

- geometric matching:

  > Given two sets S_1 and S_2 of n points in the plane, compute a minimum-weight matching in the complete bipartite graph $S_1 \text{x} S_2$, where the weight of an edge (p, q) is the Euclidian distance between p and q.

  One can use the underlying geometric structure of these graphs in order to obtain faster algorithms than those available for general abstract graphs. In term of complexity, geometric matching problems can be seen as sweeping or exploring a geometric arrangement generated by constraint sets [B92] [B01].

- Ham sandwich cuts:

  > Let $S_1, S_2, ..., S_d$ be d sets of points in $\mathbb{R}^d$, each containing n points, where n is assumed being even. A *ham sandwich cut* is a hyperplane h so that each open halfspace bounded by h contains at most $n/2$ points of S_i, for $i = 1, ..., d$.

  It is known [E87] that such a cut always exists. Let Γ_i be the set of hyperplanes dual to S_i. Then the problem reduces to computing a vertex of the intersection of $A_{n/2}(\Gamma_1)$ and $A_{n/2}(\Gamma_2)$, i.e. involving arrangements.

5.3 Robotics

Motion planning for a robot system has been a major motivation for the study of arrangements. The problem can be seen as:

> Let B be a robot system with d degrees of freedom, which is allowed to move freely within a given two- or three-dimensional environment cluttered with obstacles. Given two placements I and F of B, determine whether there exists a collision-free path between these placements.

This problem reduces to determining whether I and F lie in the same cell of arrangement of the family Γ of "contact surfaces" in $\mathbb{R}^d$, regarded as the configuration space of B. Other problems in robotics that have exploited the theory of arrangements to lead to efficient algorithms include assembly planning, fixturing, micro electronics mechanical systems (MEMS), path planning with uncertainty, and manufacturing.

6 Conclusion

Computing arrangements has been of great interest for researchers starting in the 19th century with the study of lines and many results already arised from this simple class of curves. Naturally came the curiosity of studying non-linear objects such as conics and cubics, as quite well understood mathematically, which concretized in [EKLW04].

In current research, people are also interested in computing the arrangement of algebraic curves and arbitrary ones. We can distinguish between three main approaches to overcome this problem: an approximate method [MS06] based on numerical equation solver; an exact method [W03] using rational arithmetic; and finally subdivision schemes where Yap provides a complete algorithm for intersecting two Bézier curves [Y06], Wintz et al. [WM06] compute the arrangement of implicit planar algebraic curves based on algebraic criterion, [HB06] et al. provide an output-sensitive algorithm based on interval arithmetic for computing the arrangement of arbitrary planar curves.

This interest in arrangements is motivated by a wide range of real world applications and therefore a need of more general methods being both theoretically and practically well-defined. There are still many open problems as adressed in [Y06] and a compromise is to be found between exactness and efficiency as the complexity of the input data increases.

Acknowledgements

This work was supported by the German Science Foundation (DFG IRTG 1131) as part of the International Research Training Group on "Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling, and Engineering".

References

[AE98] Agarwal P.K., and Erickson J., *Geometric Range Searching and Its Relatives, Advances in Discrete and Computational Geometry*, American Mathematical Society, Providence, 1998.

[AS00] Agarwal P.K., and Sharir M., *Arrangements and their applications*, Handbook of Computational Geometry (J. Sack, ed.), pp. 49119, 2000.

[BO79] Bentley J. L., and Ottmann T. A., *Algorithms for Reporting and Counting Geometric Intersections*, IEEE Transactions on Computers, 1979.

[B92] Breuel T. M., *Geometric Aspects of Visual Object Recognition*, PhD thesis, Massachusetts Institute of Technology, 1992.

[B01] Breuel T. M., *A Practical, Globally Optimal Algorithm for Geometric Matching under Uncertainty*, International Workshop on Combinatorial Image Analysis (IWCIA 2001), Philadelphia, CA, 2001.

[Ca90] Cass T. A., *Feature matching for object localization in the presence ofuncertainty*, Proceedings of the International Conference on Computer Vision, Osaka, Japan, 1990.

[Ch90] Chazelle B., *Cutting hyperplanes for divide-and-conquer*, Discrete Comput. Geom., 9(2), pp. 145-158, 1993.

[CdFC98] Carvalho P. C., de Figueiredo L. H., and Cavalcanti P. R., *Computing Arrangements Of Implicit Curves*, Extended abstract in Anais do VERMAC, pp. 19-22, 1998.

[E87] Edelsbrunner H., *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

[EG89] Edelsbrunner H., and Guibas L. J., *Topologically Sweeping an Arrangement*, J. Comput. Syst. Sci., vol. 38, pp. 165-194, 1989.

[EKLW04] Eigenwillig A., Kettner L., Schoemer E., and Wolpert N., *Complete, Exact, and Efficient Computations with Cubic Curves*, 20th Annual ACM Symposium on Computational Geometry, pp. 409-418, 2004.

[ESW02] Eigenwillig A., Schoemer E., and Wolpert N., *Sweeping Arrangements of Cubic Segments Exactly and Efficiently*, Technical Report ECG-TR-182202-01, 2002.

[G85] Griffiths P. A., *Introduction to Algebraic Curves*, Kuniko Weltin, trans., American Mathematical Society, Translation of Mathematical Monographs volume 70, 1985.

[H97] Halperin D., *Arrangements*, Jacob E. Goodman and Joseph O'Rourke, editors, Handbook of Discrete and Computational Geometry, chapter 21, pp. 389-412, 1997.

[HB06] Hijazi Y., Breuel T., *Sweeping Arrangements using Interval Arithmetic*, Abstract in Sixth International Conference on Curves and Surfaces - Avignon Abstracts, pp. 26-27, 2006.

[M00] Mehlhorn K., and Naeher S., LEDA: A Platform for Combinatorial and Geometric Computing, Cambridge University Press, Cambridge, UK, 2000.

[MS06] Milenkovic V., and Sacks E., *An approximate arrangement algorithm for semi-algebraic curves*, SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry, pp. 237-246, 2006.

[W02] Wolpert N., *An Exact and Efficient Approach for Computing a Cell in an Arrangement of Quadrics*, PhD thesis, Saarland University, Saarbrucken, Germany, 2002.

[W03] Wolpert N., *Jacobi curves: Computing the Exact Topology of Arrangements of Non-Singular Algebraic Curves*, 11th Europoean Symposium on Algorithms (ESA), Budapest, pp. 532-543, 2003.

[WM06] Wintz J., Mourrain B., *Subdivision method for computing an arrangement of implicit planar curves*, Preprint, Proceedings of the Algebraic Geometry and Geometric Modeling 2006 conference, to be published.

[Y06] Yap C. K., *Complete subdivision algorithms, I: intersection of Bézier curves*, SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry, pp. 217-226, 2006.

Natural Neighbor Interpolation and Order of Continuity

Tom Bobach and Georg Umlauf
University of Kaiserslautern
Computer Science Department / IRTG
D-67653 Kaiserslautern, Germany
{bobach, umlauf}@informatik.uni-kl.de

Abstract: In this paper we give a survey on natural neighbor based interpolation, a class of local scattered data interpolation schemes that define their support on *natural neighbors* in the Voronoi diagram of the input data sites. We discuss the existing work with respect to common aspects of scattered data interpolation and focus on smoothness of the interpolant.

1 Introduction

Scattered data interpolation (SDI) is the problem of finding an interpolating functional description which is as close as possible to an unknown function for which values are known only at discrete, scattered locations. Among the SDI methods existing so far, those based on natural neighbors possess the best adaption to inhomogeneous sample distributions while only building on a highly local support.

After Sibson introduced natural neighbor coordinates [Sib80] (Sibson's coordinates) and their application to SDI [Sib81], the theory of natural neighbor based local coordinates and SD interpolants built from them has received an in-depth investigation. Piper developed formulas and geometric interpretation of derivatives of Sibson's coordinates [Pip92]. Probably inspired by Sibson's original work, a less smooth type of natural neighbor coordinates (Laplace coordinates) has been proposed independently by several authors [CFL82, Sug99, BIK$^+$97]. An interesting relationship between Laplace and Sibson's coordinates has been found and generalized by Hiyoshi et al. [HS00b], yielding local coordinates of arbitrary continuity except at the data sites.

The problem of transfinite interpolation based on natural neighbor coordinates has been subject to the work of Anton et al. for Sibson's coordinates with respect to points and line segments [AMG98], of Gross et al. with respect to circles and polygons [GF99], and of Hiyoshi et al. for Laplace coordinates with respect to points, line segments, and circles [HS00a].

The geometric definition of natural neighbor coordinates is inappropriate for actual computation, especially in higher dimensions. The two main approaches to solve this are either to reformulate the geometric entities based on the Delaunay neighborhood and algebraic expressions [BS95, Sug99, Hiy05, BBU06b], or to solve the computation approximately on graphics hardware [FEK$^+$05, PLK$^+$06]. Results on the approximate computation of generalized Voronoi diagrams on graphics hardware can be found in [HCK $^+$99].

The interpolation of smooth functions requires additional efforts to overcome derivative discontinuities at the data sites inherent to all natural neighbor based local coordinates. We are only aware of two approaches, one building polynomials of the local coordinates to interpolate derivatives [Sib81, Far90, HS04], the other a construction of non-convex coordinates from a bigger natural neighborhood as explained in [Cla96, Flö03].

If the input data sites are scattered over a manifold rather than its embedding space, the Voronoi diagram and consequently the notion of natural neighbors are subject to a modified metric. That special setting received attention from [BC00, Flö03], where the local coordinate property is established for power diagrams and their restrictions to manifolds.

Outline: We will briefly review the aspects of scattered data interpolation for scalar valued functions in Section 2 to introduce the problems addressed by the natural neighbor based SDI methods which we discuss thereafter, focusing on:

- Smoothness of the local coordinates except at the data sites in Section 3,
- Smoothness of the interpolant at the data sites Section 4,
- Extension of the local coordinates to arbitrarily shaped sites Section 5,
- Extension of the local coordinates to manifolds in Section 6,
- Implementation of natural neighbor interpolation in Section 7.

We end this survey with a classifying summary of all considered methods and a comparison to some other established SDI methods in Section 8.

2 Scattered Data Interpolation

The problem of scattered data interpolation can be stated as follows: given sample data sites $X = \{x_i\}_{i=1\ldots m} \subset \mathbb{R}^n$ and data values $Z = \{z_i\}_{i=1\ldots m} \subset \mathbb{R}$, find a function $f : \mathbb{R}^n \to \mathbb{R}$ that satisfies the interpolation constraint $f(x_i) = z_i$. The subset $\{(x_j, z_j)\}_j$ on which the value at a query position q depends is called the *support* of f at q and leads to the distinction between schemes with *local* and *global* support. While schemes with global support usually have higher smoothness than local ones, their cost of computation makes them inapplicable for large scale data sets.

The aspects that are addressed by a multitude of scattered data interpolation schemes cover, among others:

Support: How is the support determined? If the size of the data set exceeds that of the available RAM, global schemes fail. Local schemes have a small memory footprint and can be computed much more efficiently, but are less smooth.

Smoothness: How often is f continuously differentiable?

Derivative Interpolation: Can f interpolate higher order derivatives at the data sites?

Polynomial Precision: Up to what order does f reproduce polynomials?

Transfinite Interpolation: Instead of points can one interpolate to curves or higher-dimensional manifolds? Transfinite interpolation leads to a continuous representation of the input data and usually requires considerably more effort in implementation.

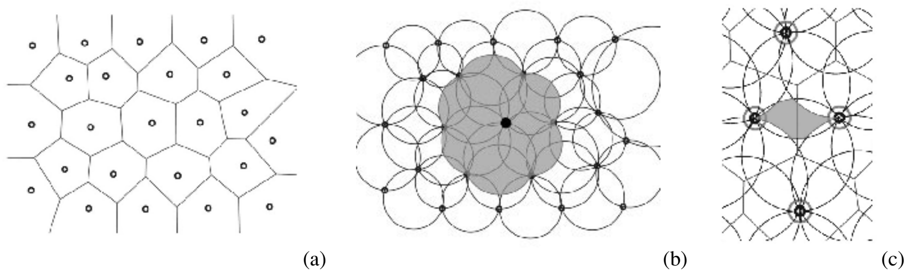


Figure 1: (a) Voronoi Diagram of points in $\mathbb{R}^2$. (b) Points in the shaded region have the bold point as one natural neighbor. (c) Points in the shaded region have the four bold points as natural neighbors.

Interpolation on manifolds: Can the interpolation scheme still be applied if the underlying space itself is a manifold, and measurements are subject to other metrics?

Computation: How can f be efficiently evaluated? Naïve implementations of interpolation methods can lead to inacceptable performance. The appropriate implementation of the interpolation schemes is important for its applicability.

Derived values: Exist formulas for derivatives or integrals of f?

Extrapolation: Local schemes typically define only $f|\mathscr{D}$ with $\mathscr{D}$ being the convex hull of X. Does f have a meaningful definition outside $\mathscr{D}$?

Approximation order: If data is sampled from a known function, how close does f get to that function with increasing sampling density? To know the approximation order of a method is to know how well it is suited to model phenomena with a certain class of governing functions.

Non-scalar values: Can function values $Z \not\subset \mathbb{R}$, e.g. $Z \subset \mathbb{R}^d$, be interpolated? For scalar data at the input sites, the space of possible functions can be described using linear combinations of neighborhood data. This does not necessarily hold for non-scalar data, where more sophisticated blending functions may be needed.

3 Natural Neighbor Coordinates

The group of interpolation schemes we discuss here exploits geometric identities of natural neighbors, building an interpolant by applying the same identity to the data values. We first repeat some facts about Voronoi diagrams and local coordinates, then focus on Laplace, Sibson's and Hiyoshi's coordinates.

3.1 Voronoi Diagrams

Let X be data sites that act as generators of a Voronoi diagram, and $d(\cdot,\cdot)$ the Euclidean distance on $\mathbb{R}^n$. The resulting Voronoi diagram (Figure 1(a)) is the partition of space into convex tiles $\mathcal{V}(X) = \{\mathscr{T}_i\}_{i=1\ldots m}$, $\bigcup_{i=1\ldots m} \mathscr{T}_i = \mathbb{R}^n$, with

$$\mathscr{T}_i = \{x \in \mathbb{R}^n | d(x, x_i) \leq d(x, x_j), i \neq j\}. \tag{1}$$

Two generators x_i and x_j are called *natural neighbors* if their associated tiles share a non-empty hyperface $s_{ij} := \mathscr{T}_i \cap \mathscr{T}_j$. To ensure boundedness of the tiles, we will restrict our considerations to the interior $\mathscr{D}$ of the convex hull of X. We denote the set of indices of the natural neighbors for a generator x_i by N_i. If $n+1$ or more tiles share a common point, the unique circumsphere through their generators is called *Delaunay sphere*, since it contains no other generator in its interior. For an overview on Voronoi diagrams the reader may refer to [Aur91, OBSC00].

We denote by $x_0 \in \mathscr{D}$ an arbitrary point, called *query point*, and define $\mathcal{V}(X \cup \{x_0\}) =: \{\mathscr{T}_i'\}_{i=0\ldots m}$. All notions from the Voronoi diagram carry over to x_0, and $\mathscr{T}_0'$ is called the *virtual tile* of the query position.

3.2 Local Coordinates

As long as the set of data sites X is not degenerate and the query point x_0 lies inside its convex hull, it is also always contained in the convex hull of its natural neighbors $\{x_i\}_{i \in N_0}$. Since $\{x_i\}_{i \in N_0}$ is in general position, i.e. contains $n+1$ affinely independent points, we can express x_0 in terms of *generalized barycentric coordinates* with respect to its natural neighbors

$$\text{(local coordinates)} \qquad x_0 = \sum\nolimits_{i \in N_0} \lambda_i(x_0) x_i, \tag{2a}$$

$$\text{(partition of unity)} \qquad 1 = \sum\nolimits_{i \in N_0} \lambda_i(x_0), \tag{2b}$$

$$\text{(convexity)} \qquad 0 \leq \lambda_i(x_0), \qquad i \in N_0. \tag{2c}$$

Then, (2a) - (2c) guarantee affine invariance for λ, yielding the linear precision scattered data interpolant

$$f(x_0) = \sum\nolimits_{i \in N_0} \lambda_i(x_0) z_i. \tag{3}$$

We will refer to the local coordinates λ_i by the $|N_0|$-tuple λ and omit the argument x_0 for the sake of brevity unless required by context. For $|N_0| = n+1$, λ reduces to the usual barycentric coordinates and f is a linear function. If $|N_0| > n+1$, there are infinitely many choices for λ that satisfy (2a)-(2c).

From (3) it is clear that f is as smooth as λ. Therefore, we concentrate on how to control the smoothness of λ. The sequence of local coordinates we discuss next will be denoted by λ^k and f^k consequently denotes the interpolant (3) based on these coordinates.

3.3 Some Properties of Natural Neighbor Coordinates

Natural neighbor coordinates are based on sizes of geometric entities in the virtual tile $\mathscr{T}_0'$ of the query position x_0. The rate at which these entities change with x_0 basically determines the smoothness of the coordinates. Whenever the query position coincides with a data site, $x_0 = x_i$, these entities are not defined. The coordinates, however, can be continuously extended for $x_0 \to x_i$, yielding C^0 continuity at the data sites.

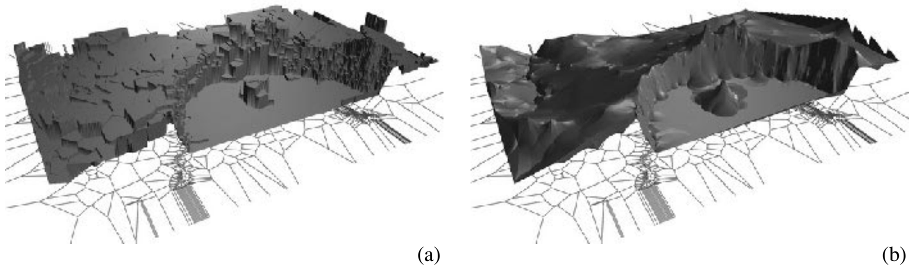

(a) (b)

Figure 2: Interpolation of 1493 scattered points sampled from the crater lake data set. The original data set is due to US geological survey with 344 · 463 points. (a) The nearest neighbor interpolant is piecewise constant and discontinuous along the edges of the Voronoi diagram. (b) The Laplace interpolant is continuous with derivative discontinuities along the Delaunay circles.

The region of influence for each data value z_i is the set of points which have x_i as natural neighbor, i.e. for which the contribution of z_i is not zero. This region is just the union of all Delaunay spheres passing through x_i, depicted in Figure 1(b) for the center data site. The regions of constant neighborhood, i.e. where N_0 does not change, are all areas that are bounded by Delaunay spheres, as depicted in Figure 1(c). These regions are considerably more complex than those appearing e.g. in barycentric interpolation in Delaunay tessellations, where they are polygonal domains.

3.4 Nearest Neighbor Interpolation

A simple scattered data interpolation scheme that uses the Voronoi diagram of data sites is *nearest neighbor interpolation*. The weights used here are simply defined as

$$\lambda_i^{-1} := \begin{cases} 1, & x_0 \in \mathscr{T}_i, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, f is discontinuous along the Voronoi edges, as can clearly be seen in Figure 2(a).

3.5 Laplacian Interpolation

A set of C^{n-2}-smooth local coordinates has been proposed by different authors as Laplace- or Non-Sibsonian coordinates [CFL82, Sug99, BIK$^+$97]. Denote by $\sigma_i := \mathrm{vol}^{n-1}(s_{0i})$ the $n-1$-dimensional area of the hyperface shared by $\mathscr{T}_0'$ and $\mathscr{T}_i'$, and $r_i := d(x_0, x_i)$. Then Laplace coordinates λ^0 are defined as

$$\hat{\lambda}_i^0 := \sigma_i / r_i \qquad \text{and} \qquad \lambda_i^0 := \hat{\lambda}_i^0 / \sum\nolimits_{j \in N_0} \hat{\lambda}_j^0.$$

These coordinates and the resulting interpolant f^0 are continuous in $\mathscr{D}$ and have derivative discontinuities at the generators. For $X \subset \mathbb{R}^n$, we find that λ_i^0 is C^{n-2} on the Delaunay

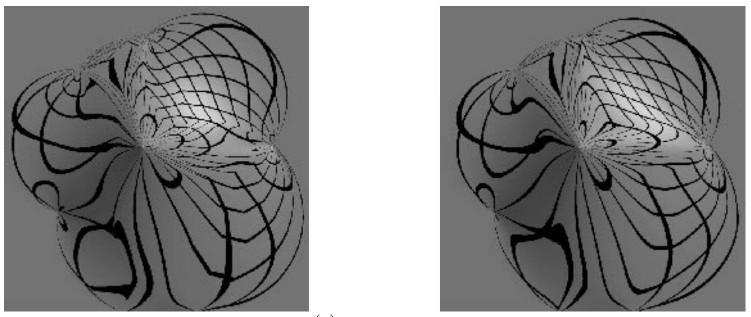

(a) (b)

Figure 3: Reflection lines on the basis function of Sibson's (a) and Hiyoshi's C^2 (b) coordinates, seen from above. (a) The cusps indicate the C^2 discontinuities at the Delaunay circles. (b) Due to C^2 continuity away from the data sites, the cusps have vanished.

spheres. For $n = 2$ this results in the C^0 artifacts that can be seen in Figure 2(b). Different proofs for λ_i^0 satisfying (2a) have been given in [HS00b, BIK$^+$97].

3.6 Sibson's Interpolation

The first appearance of natural neighbor based local coordinates is due to Sibson [Sib80], who extended this to scattered data interpolation in [Sib81]. Sibson's coordinates are based on the volumes $\nu_i := \mathrm{vol}^n(\mathscr{T}_0' \cap \mathscr{T}_i)$ via

$$\hat{\lambda}_i^1 := \nu_i, \qquad \text{and} \qquad \lambda_i^1 := \hat{\lambda}_i^1 / \textstyle\sum_{j \in N_0} \hat{\lambda}_j^1.$$

The fact that ν_i is a n-dimensional volume results in λ^1 being C^{n-1} continuous except at the generators, where they are still only C^0. See Figure 3(a) for an example.

Properties of Sibson's coordinates received close attention by Farin [Far90] and Piper [Pip92]. With c_i denoting the centroid of s_{0i} their explicit formula for the gradient of λ_i is

$$\nabla \lambda_i^1 = \sigma_i (c_i - x_0)/r_i. \tag{4}$$

Different proofs for λ_i^1 satisfying (2a) have been given by [Sib80, Pip92, HS00b].

3.7 Hiyoshi's Interpolation

Hiyoshi and Sugihara [HS00b] proposed a generalization of Laplace and Sibson's coordinates based on an integral of σ_i. In [Hiy05] Hiyoshi restated this as

$$\hat{\lambda}_i^k := \frac{1}{(k-1)!} \int_{p \in \mathscr{T}_i \cap \mathscr{T}_0'} \big((x_0 - x_i) \cdot (p - c_i)\big)^{k-1} |dp|,$$
$$\lambda_i^k := \hat{\lambda}_i^k / \textstyle\sum_{j \in N_0} \hat{\lambda}_j^k.$$

For $k = 0, 1$, Hiyoshi's coordinates coincide with Laplace and Sibson's coordinates. For $k > 1$, these coordinates are C^{k+n-2} in $\mathscr{D} \setminus X$. As Hiyoshi pointed out in [Hiy05], the limit $k \to \infty$ does not lead to C^∞ coordinates but to the piecewise linear interpolant on the Delaunay tessellation.

4 Smooth Interpolation at the Data Sites

The previous section considered interpolants building on a linear combination of data values by local coordinates as in (3), resulting in derivative discontinuities at x_i. Here we consider basically two approaches to overcome this. One is to construct polynomials of the local coordinates to control the derivatives at the data sites. Another is to construct local coordinates from a larger neighborhood, which results in smooth, non-convex weights. To apply the first approach, the derivatives at the data sites need to be known. Otherwise they can be estimated using the approach described in [BBU06b].

In the remainder of this section we will denote by f^{ab} an interpolant based on local coordinates λ^a which has smoothness C^b at the data sites.

4.1 Sibson's C^1 Interpolant

In [Sib81] Sibson described a construction of a C^1 interpolant. He generates gradients ∇_i based on the weighted least squares plane through the neighboring data values, which are then interpolated by blending first order functions with the help of coordinates λ_i^1. With $r_i := d(x_0, x_i)$, $\gamma_i := \lambda_i^1 / r_i$, define

$$\begin{aligned} \zeta_i &:= z_i + (x_0 - x_i)^T \nabla_i && \text{and} \quad \zeta := \left(\sum\nolimits_{i\in N_0} \gamma_i \zeta_i\right) / \left(\sum\nolimits_{i\in N_0} \gamma_i\right), \\ \alpha &:= \left(\sum\nolimits_{i\in N_0} \lambda_i^1 r_i\right) / \left(\sum\nolimits_{i\in N_0} \gamma_i\right) && \text{and} \quad \beta := \sum\nolimits_{i\in N_0} \lambda_i^1 (r_i)^2. \end{aligned}$$

Blending this with Sibson's C^0 interpolant $f^1(x_0)$ yields Sibson's C^1 interpolant

$$f^{11}_{Sib}(x_0) = (\alpha f^1(x_0) + \beta\zeta)/(\alpha + \beta),$$

which does not easily generalize to higher orders of continuity.

4.2 Farin's C^1 Interpolant

A much more general approach which is not restricted to natural neighbor coordinates but can be applied to all local coordinates having properties (2a)-(2c) was proposed by Farin [Far90]. λ can be seen as barycentric coordinates in a l-variate Bézier simplex which projects to the convex hull of $\{x_i\}_{i\in N_0}$, where $l = |N_0|$. In Bézier simplexes it is easy to model directional derivatives at the vertices x_i by appropriately choosing the Bézier control net. From prescribed derivatives at x_i, a certain number of control points is fixed.

The remaining control points can be chosen arbitrarily without interfering with the interpolation property. Exploiting the concept of degree elevation, these can be chosen to yield polynomial precision. In the following, we will denote by D_v the directional derivative along v. By $\alpha \in \mathbb{N}^n$, $\sum \alpha_i = d$, we denote the n-dimensional multi-index that enumerates the d-th degree Bézier control points b_α. The indexes of the vertices of the Bézier simplex are denoted by α^i, where the i-th entry is set to d, and e^i is a multi-index with zero entries except for the i-th component which is one.

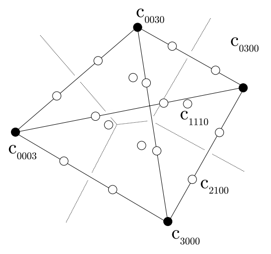


Figure 4: Planar projection of the control net of a cubic Bézier simplex in $\mathbb{R}^3$.

Farin presented the implementation of the above idea for cubic Bézier simplexes over λ^1 to interpolate gradients ∇_i, yielding a globally C^1 interpolant. By setting the directional derivatives in each vertex x_i towards its neighbor x_j,

$$D_{x_j - x_i} = \nabla_i (x_j - x_i), \qquad j \in N_0 \setminus \{x_i\},$$

we constrain x_i and all inner control points $b_{\alpha^i - e^i + e^j}$, to be coplanar, i.e.

$$b_{\alpha^i} = z_i, \qquad \text{and} \qquad b_{\alpha^i - e^i + e^j} = z_i + \frac{1}{3}(x_j - x_i)^T \nabla_i \ \text{ for } i \neq j.$$

This fixes all control points except those on simplex faces. By degree elevation for Bézier simplices, these are chosen to ensures quadratic precision of the resulting interpolant, see [Far90, Flö03]. Let $\beta = e^i + e^j + e^k$ for $i < j < k$, then b_β is an inner control point. Set $u = \frac{1}{3}(b_{\alpha^i} + b_{\alpha^j} + b_{\alpha^k})$ and $v = \frac{1}{6} \sum_{a,b \in \{i,j,k\}, a<b} b_{\beta - e^a + e^b}$, the average of the remaining fixed control points, then $b_\beta = \frac{3}{2}v - \frac{1}{2}u$ yields quadratic precision for the interpolant. The resulting interpolant inherits C^1 continuity on $\mathscr{C} \setminus X$ from λ^1 and is given by

$$f^{11}_{Far}(x_0) := b^3(\lambda^1).$$

4.3 Hiyoshi's C^2 Interpolant

Applying the above approach to quintic Bézier simplexes over λ^2, [HS04] present a construction of control points that matches derivatives up to order two given by the *Hessian* $\mathcal{H}_i$ at generator x_i. Let i, j, k be mutually distinct and $d_{ij} = x_j - x_i$, then

$$\begin{aligned}
b_{\alpha^i} &= z_i, \\
b_{\alpha^i - e^i + e^j} &= z_i + \frac{1}{5}{\nabla_i}^T d_{ij}, \\
b_{\alpha^i - 2e^i + 2e^j} &= z_i + \frac{2}{5}{\nabla_i}^T d_{ij} + \frac{1}{20}{d_{ij}}^T \mathcal{H}_i d_{ij}, \\
b_{\alpha^i - 2e^i + e^j + e^k} &= z_i + \frac{1}{5}({\nabla_i}^T d_{ij} + {\nabla_i}^T d_{ik}) + \frac{1}{20}{d_{ij}}^T \mathcal{H}_i d_{ik}
\end{aligned}$$

fix the control points based on the prescribed derivatives. Cubic precision of the resulting interpolant is given by the choice of the remaining control points based on the degree elevation principle, see [HS04].

4.4 Clarkson's Interpolation

One special kind of local coordinates that does not directly fit definition (2) is an idea of Clarkson [Cla96, Flö03]. It is based on the two-ring neighborhood of the query position and is designed to reproduce spherical quadratics, i.e. functions of the form $x \mapsto a\|x-b\|^2$, $a \in \mathbb{R}, b \in \mathbb{R}^n$. It is the only approach so far that has a really implicit C^1 construction and does not depend on prescribed derivative information. Clarkson's local coordinates differ significantly from those of Section 3:

- They depend on $\bigcup_{i \in N_0} N_i$, i.e. the two-ring neighborhood of x_0.
- They are not convex, i.e. (2c) does not hold.
- They seem to be C^1 at x_i, which is not yet proved.

5 Transfinite Interpolation

In this section we discuss methods to interpolate line segments, polygons and circular arcs instead of points. Non-point generators lead to *generalized Voronoi diagrams*, and the geometric primitives that constitute the local coordinates from Section 3 are no longer convex polygons. The main consequence of this generalization is an increased complexity in both data handling and the computation of the interpolant, which also seems to be the reason that research in this direction has been restricted to two dimensions so far.

Interestingly, transfinite interpolation enables us to impose discontinuities along the manifold data sites by using different values for each side.

This section will first explain the main differences between ordinary Voronoi diagrams and such with curves as generators, before taking a closer look at how the identities from Section 3 extend to the transfinite case.

5.1 Generalized Voronoi Diagrams in 2D

Assume the data sites are points $x_i \in X$ and non-intersecting curves $c_i \in C$, where i runs over the combined set of data sites $X \cup C$. Definition (1) still holds with a modified distance function,

$$d(x, c_i) = \min_{q \in c_i} \|x - q\|.$$

We denote the bisectors $\mathscr{T}_i \cap \mathscr{T}_j$ by e_{ij}. Tiles induced by points are still convex, while for curves this is in general not true. As in the ordinary Voronoi diagram, the virtual insertion of a new point x_0 into $\mathcal{V}(X)$ results in a new, convex tile $\mathscr{T}_0'$. An example of a generalized Voronoi diagram and the virtual tile (shaded) can be seen in Figure 5(a). The shape of the bisectors between the various elements of the Voronoi diagram is at least as complicated as that of the elements itself. Thus, an exact computation of areas and lengths seems feasible only for simple shapes of the generators. For arbitrary shapes, the Voronoi diagram can be approximated using graphics hardware, see Section 7.3.

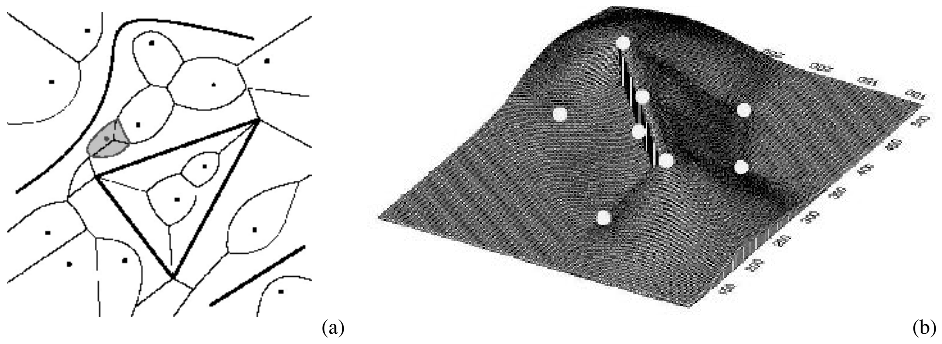


Figure 5: (a) The Voronoi diagram of a set of points, line segments and general curves (drawn bold). The virtual tile of a new point is shaded. Picture courtesy of [Hof99]. (b) Transfinite interpolation of a directed line segments and a couple of points. Picture courtesy of [AMG04].

5.2 Interpolating Data on Line Segments

If the data sites are line segments, there are bisectors between lines, between points, and between points and lines, where endpoints of line segments also count as points. The bisectors are parabolic arcs between the interior of a line segment and a point, while all other bisectors remain linear. In practice, the endpoints of a line segment are treated as separate generators, which leads to a partition of its Voronoi tile into tiles for its directed half edges and its end points, as shown in Figure 6(b).

To account for the continuous nature of the data sites, local coordinates in the transfinite setting have their identity expressed similar to

$$x_0 = \sum_{i \in N_0^X} \lambda_i x_i \; + \sum_{i \in N_0^C} \int_{q \in c_i} \lambda_i(q) q \, |dq|, \tag{5}$$

with $N_0^X \cup N_0^C$ being the union of point shaped and line shaped neighbor indices, and $\lambda_i(q)$ denoting a scalar weight function over the length of c_i. The interpolant thus is

$$f(x_0) = \sum_{i \in N_0^X} \lambda_i z_i \; + \sum_{i \in N_0^C} \int_{q \in c_i} \lambda_i(q) z_i(q) \, |dq|, \tag{6}$$

with $z_i(q)$ being the scalar value distribution over the site.

In [GF99], the interpolation of arbitrary functions along polygons is solved. Each subtile $\mathscr{T}_0' \cap \mathscr{T}_i$ can be interpreted to have a certain thickness above x_i, which is nonzero only where the subtile projects to x_i. The $\lambda_i(q)$ are taken to be this thickness, normalized by the overall area, and define a meaningful density for the accumulation of data values. The application of this interpolant to the data distributed along the non-convex polygon in Figure 6(a) is shown in Figure 6(b).

In [AMG98, AMG04], the same approach has been implemented for arbitrary arrangements of non-intersecting line segments and points. Although they restrict their approach to a linear data distribution along the lines, the approach of [GF99] can also be applied to

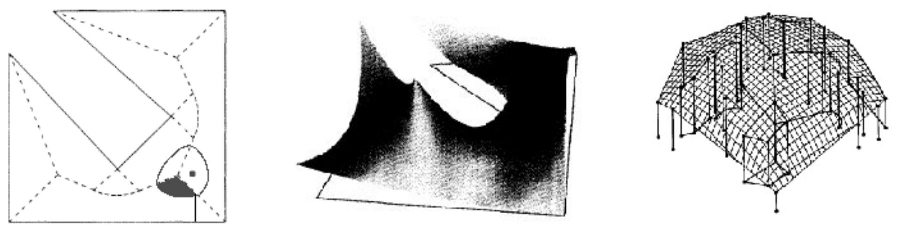


Figure 6: Transfinite interpolation of curves. (a) The Voronoi diagram of a polygon, the contribution of the lower subtile depicted by the thin lines. (b) Transfinite interpolation of the boundary values. (c) Transfinite interpolation of a collection of points, line segments and circular arcs. Pictures (a), (b) courtesy of [GF99], (c) courtesy of [HS00a].

interpolate to arbitrary scalar functions over the sites. By allowing different values on both sides of the line segments, they are able to faithfully model discontinuities as they arise in e.g. geology. See Figure 5(b) for an example.

While the last two approaches focus on a generalization of Sibson's coordinates, [HS00a] generalizes Laplace interpolation to arrangements of multiple classes of curves. The main difference lies in the definition of $\lambda_i(q)$, which can now be interpreted as a density function over the bisectors boundary of $\mathscr{T}_0'$. The result of this interpolant applied to an arrangement of points, line segments, and circles is shown in Figure 6(c).

5.3 Interpolating Data on Circles, Lines and Points

In case the input consists of data distributed over a circle, the tile $\mathscr{T}_0'$ is an ellipsis. Consequently, Sibson's transfinite interpolant takes a simple form. Let x_1 be the circle centered at 0, $z_1(\Theta)$ the data, parameterized over $\Theta \in [0, 2\pi)$, and $x_0 = (\rho, \theta)$ be expressed in polar coordinates with respect to 0. Then in [GF99] a Sibson's transfinite interpolant on circles is defined as

$$f(x_0) = \frac{(1-\rho^2)^{3/2}}{2\pi} \int_0^{2\pi} \frac{z_1(\Theta)}{(\rho\cos(\theta-\Theta)-1)^2} d\Theta \qquad \begin{cases} 0 \le \rho < 1 \\ 0 \le \theta \le 2\pi \end{cases} .$$

Based on a similar idea, [HS00a] formulated an identity and the thereby defined interpolant for Laplace coordinates.

6 Natural Neighbor Coordinates on Manifolds

The Voronoi diagram is defined by a set of points and a distance measure. For points on a manifold, this definition still holds, at the expense of potentially non-convex tiles due to a non-Euclidean metric, see [LL00]. The manifold setting results in bisectors of arbitrary complexity and computing areas (volumes) becomes tedious for non-trivial geometries. To the author's best knowledge, there has been no work carried out on natural neighbor based interpolation on continuous manifolds.

In [BC00], however, it is shown that if the manifold has a sufficiently dense sampling, a less complicated approach is possible. The data sites on the manifold induce a Voronoi diagram in the manifold's embedding space, $\mathbb{R}^n$. The intersection of that Voronoi diagram and the manifold gives a partition of the manifold that locally converges to the Euclidean Voronoi diagram when the sampling density goes to infinity. Furthermore, the main result in [BC00] states that Sibson's identity holds for an infinitely dense sampling of the surface.

Based on this work, natural neighbor based interpolation on point clouds issued from manifolds is developed in [Flö03]. As a main result, a point on a manifold can be expressed in local coordinates in the tangent plane at that point, given the manifold is sampled densely enough. The intersection of the n-dimensional Voronoi diagram of the data sites with the tangent plane defined by the normal vector at the query position produces a power diagram in the tangent plane. [Flö03] proves Sibson's identity for power diagrams and develops natural neighbor coordinates for point clouds.

7 Implementation of Natural Neighbor Interpolation

Natural neighbor based interpolants are based on an underlying identity that provides generalized barycentric coordinates in the natural neighbors. The definition of those local coordinates is motivated geometrically on the Voronoi diagram of the input data sites. The computation, however, can often be carried out in a more elegant and also more stable way. These approaches can be classified as geometric, algebraic and approximate. For simple settings, the geometric approach is still feasible. For higher dimensions, higher orders of continuity and more complex input data sites, algebraic and approximate approaches yield more efficient and more stable solutions.

In the rest of this section we describe the computation of natural neighbor coordinates, since they are the main building block for all interpolants in this survey. The implementation of the C^1 and C^2 constructions at the data sites from Section 4 is straightforward.

7.1 Geometric Computation

The dual to the Voronoi diagram is the Delaunay tessellation. Therefore, evaluation and traversal of the Voronoi diagram of a set of points can be carried out on its Delaunay tessellation, for which the adjacency information is known. One drawback of this approach is the numerical instability in cases where both nominator and denominator in the formulas of Section 3 tend to infinity.

Laplace and Sibson's coordinates relate to areas and volumes of intersections of Voronoi tiles which are easily computed in two dimensions, and implementations are known for three dimensions as well [Owe93]. In case of line segment shaped data sites, the constrained Delaunay tessellation can be used. Input data sites of arbitrary shape are difficult to handle in classical geometric data structures and usually require more intricate representations of the Voronoi diagram. The common solution to this is a tessellation of the input data sites, once again allowing for the constrained Delaunay tessellation to be applied.

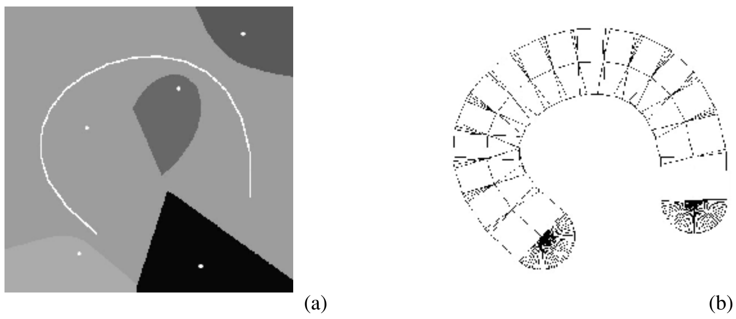


Figure 7: (a) Generalized Voronoi diagram computed on graphics hardware. (b) Polygonal approximation of a distance function. Pictures courtesy of [HCK$^+$99].

In three or more dimensions, the data structures required for storing the Delaunay tessellation and its adjacency graph become very complex, and traversing the topological neighborhood becomes error prone and cumbersome.

7.2 Algebraic Computation

For algebraic computation the explicit construction of the Voronoi diagram is avoided, and computation is carried out directly on the geometric entities by which it is defined. Note that except for Hiyoshi's approach [Hiy05], also algebraic approaches suffer numerical instabilities when both nominator and denominator tend to infinity.

To compute Laplace coordinates in the two-dimensional setting, the calculation presented by Sugihara only assumes the natural neighbors of the query position to be given in counterclockwise order [Sug99]. The resulting identity also holds in the more general case of star-shaped neighborhoods, making this approach robust against topological inconsistencies as they appear from numerical noise.

Watson [Wat92] explains Sibson's coordinates as the signed decomposition of the area of a triangle, spanned by the circumcenters of the involved triangles. Building on this idea, [Hiy05] proposed a way to stably compute λ^k in $\mathbb{R}^2$ by encoding the construction of Voronoi entities into algebraic expressions in Delaunay entities that circumvent numerical instabilities based on zero denominators as they might appear in the equations in Section 3.

A straightforward computation of Laplace and Sibson's coordinates in any dimension exists once the one-ring Delaunay neighborhood is known. The content of the corresponding tile facets and tile intersections can be expressed as an intersection of half-spaces that are entirely defined by the query position and its Delaunay neighbors [BS95]. Thus, the computation of Laplace and Sibson's coordinates reduces to the determination of Delaunay neighbors [Wat81] and volume computation in n dimensions [BEF00]. This approach has been applied to derive a construction of Hiyoshi's coordinates in $\mathbb{R}^2$ in [BBU06b]. Note that the average number of Delaunay neighbors grows exponentially with dimension, and so does the complexity of volume computations.

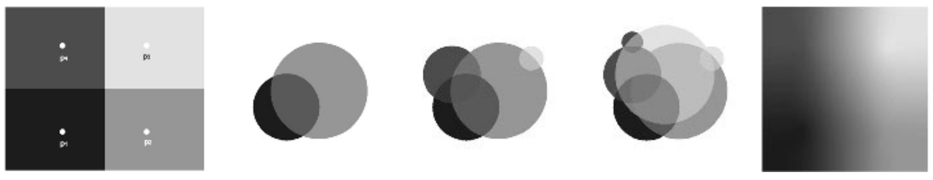

(a) (b) (c) (d) (e)

Figure 8: Discrete computation of Sibson's interpolant for the setting in (a). Each point in the domain gives rise to a disc colored with the value of the nearest generator and the distance to that generator as its radius, depicted in (b)-(d). The translucent overlay of all discs is the discrete Sibson's interpolant. Pictures courtesy of [PLK+06].

7.3 Approximate Computation

By allowing a certain error for the local coordinates, an approximate formulation of natural neighbor coordinates can be given based on a discretization of the Voronoi diagram. The discrete version of the two-dimensional Voronoi diagram can efficiently be computed by rendering appropriate primitives to frame- and z-buffer, utilizing the capabilities of recent graphics hardware.

The computation of generalized Voronoi diagrams with the help of graphics hardware is discussed in [HCK+99]. Basically, the distance function of each of the generators is represented by a geometric object. Rendering these leaves the minimum distances in the z-buffer and the associated generator in the color buffer. An example is shown in Figure 7.

The computation of Sibson's coordinates can now be performed by counting pixels in the approximate Voronoi diagram with added query position [FEK+05]. However, this does not allow for an efficient or stable evaluation of Laplace or Hiyoshi's C^k coordinates.

If, instead of evaluating single point queries, a whole field is to be evaluated, the influence of the data values at the generators can directly be distributed to the domain in a more efficient manner. The way described in [FEK+05] requires the Delaunay triangulation to be known, while [PLK+06] do without tessellation at all solely using a k-d tree to provide nearby points. This is illustrated in Figure 8.

8 Summary and Comparison

The properties of all schemes discussed in this paper are summarized in Table 8. A discussion of the four blocks is given below.

Point Based Interpolation Schemes: Schemes with global smoothness have only been proposed for the setting of data sites. Both Farin's and Sibson's C^1 constructions operate on Sibson's coordinates and yield fairly straightforward implementations. Farin's construction has quadratic precision and adapts to a wider range of input constellations.

Hiyoshi's C^2 scheme provides a high quality interpolant but is computationally expensive and tedious to implement even though explicit guidelines for its implementation in $\mathbb{R}^2$ exist. The C^2 construction at the data sites requires the construction of quintic Bézier control nets and bears considerable combinatorial complexity. In our experiments, we found

	Shape of x_i [(6)]	Smoothness of λ in $\mathscr{D} \setminus X$	Deriv. at x_i	Smoothness at x_i	Gener. to $\mathbb{R}^d$	Gener. to C^k	Precision	Comput. Compl.[(4)]	Support[(5)]	Cont. dep. on X	References
Point based											
Lap. coord.	$\top$	C^{d-2}	-	C^0	+	-		++	1	+	[CFL82, Sug99, BIK$^+$97]
Sib. coord.		C^{d-1}	-	C^0	+	-	linear	++	1	+	[Sib81]
Hiy. coord.		C^k	-	C^0	(+)[(1)]	+		-	1	+	[HS00b]
Sibson C^1	points	C^0	∇	C^1	+	-	s.q.[(3)]	+	1	+	[Sib81, Far90, Pip92]
Farin C^1		C^1	∇	C^1	+	+ [(2)]	quad.	+	1	+	[Far90]
Hiyoshi C^2		C^2	$\nabla, \mathscr{H}$	C^2	(+)[(1)]	+ [(2)]	cub.	–	1	+	[HS04, Hiy05]
Clarkson C^1	$\perp$	C^1 in $\mathscr{D}$	-	C^1	+	-	s.q.[(3)]	?	2	+	[Cla96, Flö03]
Transfinite											
Gross	pol, ci	C^1	-	C^0	-	-		-	1	+	[GF99]
Anton	pt, li	C^1	-	C^0	-	-	linear	-	1	+	[AMG98, AMG04]
Hiyoshi	pt, li, ca	C^1	-	C^0	-	-		-	1	+	[HS00a]
Manifold											
Flötotto	points	C^1	-	C^0	+	-	n.a.	-	1	n.a.	[BC00, Flö03]
Other meth.											
Nearest n.	points	C^{-1}	-	C^∞	+	-	-	++	0	-	[OBSC00]
FEM	points	C^k	-	C^0	+	-	lin.	++	0	-	-
RBF	points	C^∞	-	C^∞	+	+	polyn.	- -	g	-	-

(1) ongoing research. (2) based on the Bézier simplex approach. (3) spherical quadratics. (4) ++ low, - - high.
(5) $\{012\}$-ring, g(lobal). (6) pt=points, li=lines, pol=polygons, ci=circles, ca=circular arcs.

Table 1: Overview of considered interpolation schemes.

considerable increases in computation time for extreme situations with more than 20 natural neighbors, which very likely becomes an issue in higher dimensions. Besides these drawbacks, the C^2 interpolant provided the best results when applied to data representing a smooth function, which has been verified in [BBU06a].

In contrast to the schemes above, Clarkson's construction does not interpolate prescribed derivatives but achieves C^1 smoothness implicitly. Since the final interpolant is a linear combination of data from the two ring neighborhood it is similar to the other C^0 schemes over natural neighbor coordinates, but requires a larger support and results in non-convex coordinates.

Transfinite Interpolation: Research on transfinite natural neighbor interpolation has so far only concentrated on expressing the identity of Laplace and Sibson's coordinates with respect to line- and circle-shaped generators in two dimensions. Consequently, the resulting interpolants remain C^0 across the data sites. In simple cases like line segments and circular arcs, closed form integration is possible, but more general shapes require approximations. In spite of these restrictions the improved flexibility provided by transfinite interpolation is useful e.g. for fault modeling in geosciences.

Interpolation on Manifolds: Sibson's identity holds on smooth manifolds for an infinitesimal sampling, but not necessarily for arbitrary samples of the manifold. The Voronoi diagram in that non-Euclidean metric does not have the same, simple geometric structure.

However, local restriction of the Euclidean Voronoi diagram to the tangent plane reveals the lower-dimensional power diagram, for which Laplace and Sibson's identity hold. As a result, Sibson's interpolant can be used on manifolds with sufficiently dense sampling.
Other Scattered Data Schemes: Many other scattered data interpolation schemes have been proposed, among them finite elements, radial basis functions with global or local support, subdivision and bivariate splines, all of which have advantages in certain applications. Yet, natural neighbor based interpolation offers a unique combination of the properties

- locality,
- support determined by truly automatic neighborhood,
- continuous dependency on positions of input sites.

Radial basis functions offer very good mathematical properties in terms of approximation order and smoothness and do like natural neighbor based schemes not depend on a particular tessellation. But even the construction of a compactly supported interpolant requires the solution of a global linear system. Finite elements can be constructed with high orders of continuity but are defined over one fixed choice of elements, i.e. the tessellation of the domain, and thus do not continuously depend on the positions of the data sites. Similar arguments apply to bivariate splines and subdivision. In the more relaxed setting of scattered data *approximation* approaches like hierarchical B-splines, thin plate splines exist, or moving least squares exist. Of these, only the latter has properties similar to natural neighbor based schemes and can even be integrated with natural neighbor coordinates as a replacement for the inverse distance weights.

9 Conclusion

Natural neighbor based interpolation offers some unique properties that make it appealing in settings where the sample distribution is inhomogeneous or changes over time. Its local support is an advantage in large scale data processing, its automatic neighborhood definition and the continuous dependency on the data site positions is especially interesting for meshless methods in mechanical engineering and computational fluid dynamics, where they have been successfully applied in two- and three-dimensional settings. Sibson's identity as well as Laplace coordinates have been generalized to data sites of arbitrary shape, which is useful in geological and terrain modeling.

The main drawback of natural neighbor based interpolation so far lies in the lack of smoothness of the local coordinates at the data sites. For point shaped data sites, this can be solved. One remedy is a non-convex coordinate construction by Clarkson. Another are schemes that interpolate prescribed derivatives, which are unknown in most settings and must therefore be estimated. Furthermore, local coordinates with higher smoothness so far only exist for two dimensions and are tedious to implement.

Some unsolved aspects about natural neighbor based interpolation remain interesting. The smoothness across the data sites in transfinite interpolation could be improved by adopting one of the approaches from the point shaped setting. The implementation of higher-dimensional local coordinates with C^{2+}-smoothness is an open problem and subject to current research. Furthermore, direct formulas for derived values such as integrals would certainly add to the attractivity of natural neighbor based interpolation.

References

[AMG98] F. Anton, D. Mioc, and C. Gold. Local coordinates and interpolation in a Voronoi diagram for a set of points and line segments. In *Proceedings of the 2nd Voronoi Conference on Analytic Number Theory and Space Tillings*, pages 9–12, 1998.

[AMG04] F. Anton, D. Mioc, and C. Gold. Line Voronoi diagram based interpolation and application to digital terrain modelling. In *ISPRS - XXth Congress, Vol.2*. International Society for Photogrammetry and Remote Sensing, 2004.

[Aur91] F. Aurenhammer. Voronoi Diagrams - A survey of a fundamental geometric data structure. *ACM Computing surveys*, 23(3):345–405, 1991.

[BBU06a] T. Bobach, M. Bertram, and G. Umlauf. Comparison of Voronoi Based Scattered Data Interpolation Schemes. In *Proc. Visualization, Imaging and Image Processing*, 2006.

[BBU06b] T. Bobach, M. Bertram, and G. Umlauf. Issues and Implementation of C^1 and C^2 Natural Neighbor Interpolation. In *Proceedings of the* 2^{nd} *International Symposium on Visual Computing*, 2006.

[BC00] J. Boissonnat and F. Cazals. Natural neighbour coordinates of points on a surface. Technical Report 4015, INRIA-Sophia., 2000.

[BEF00] B. Bueler, A. Enge, and K. Fukuda. Exact volume computation for polytopes: a practical study. *DMV Seminar*, 29:131–151, 2000.

[BIK$^+$97] V. Belikov, V. Ivanov, V. Kontorovich, S. Korytnik, and A. Semenov. The Non-Sibsonian interpolation: A new method of interpolation of the values of a function on an arbitrary set of points. *Comp. Math. and Math. Physics*, 37(1):9–15, 1997.

[BS95] J. Braun and M. Sambridge. A numerical method for solving partial differential equations on highly irregular grids. *Nature*, 376:655–660, 1995.

[CFL82] N. H. Christ, R. Friedberg, and T. D. Lee. Weights of links and plaquettes in a random lattice. *Nuclear Physics B*, 210(3):337–346, 1982.

[Cla96] K.L. Clarkson. Convex Hulls: Some Algorithms and Applications. Presentation at Fifth MSI-Stony Brook Workshop on Computational Geometry, 1996.

[Far90] G. Farin. Surfaces over Dirichlet Tessellations. *Computer Aided Geometric Design*, 7:281–292, 1990.

[FEK$^+$05] Q. Fan, A. Efrat, V. Koltun, S. Krishnan, and S. Venkatasubramanian. Hardware-assisted Natural Neighbor Interpolation. In *Proc. 7th Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2005.

[Flö03] J. Flötotto. *A coordinate system associated to a point cloud issued from a manifold: definition, properties and applications*. PhD thesis, Université de Nice-Sophia Antipolis, Sep 2003. http://www.inria.fr/rrrt/tu-0805.html.

[GF99] L. Gross and G. Farin. A transfinite form of Sibson's interpolant. *Discrete Applied Mathematics*, 93:33–50, 1999.

[HCK$^+$99] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast Computation of Generalized Voronoi Diagrams using Graphics Hardware. In *Proceedings of ACM SIGGRAPH 1999*, 1999.

[Hiy05] H. Hiyoshi. Stable computation of natural neighbor interpolation. In *Proceedings of the 2^{nd} International Symposium on Voronoi Diagrams in Science and Engineering*, pages 325–333, 2005.

[Hof99] K. Hoff. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. Siggraph'99 Presentation, 1999. http://www.cs.unc.edu/ geom/voronoi/.

[HS00a] H. Hiyoshi and K. Sugihara. An Interpolant Based on Line Segment Voronoi Diagrams. In *JCDCG*, pages 119–128, 2000.

[HS00b] H. Hiyoshi and K. Sugihara. Voronoi-based interpolation with higher continuity. In *Symposium on Computational Geometry*, pages 242–250, 2000.

[HS04] H. Hiyoshi and K. Sugihara. Improving the Global Continuity of the Natural Neighbor Interpolation. In *ICCSA (3)*, pages 71–80, 2004.

[LL00] G. Leibon and D. Letscher. Delaunay Triangulations and Voronoi Diagrams for Riemannian Manifolds. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 341–349, 2000.

[OBSC00] A. Okabe, B. Boots, K. Sugihara, and S. Chiu. *Spatial Tessellations: Concepts and applications of Voronoi diagrams.* John Wiley & Sons Ltd, 2000.

[Owe93] S.J. Owen. Subsurface Characterization with three-dimensional Natural Neighbor Interpolation. Available at `http://www.andrew.cmu.edu/user/sowen/natneigh/index.html`, 1993.

[Pip92] B.R. Piper. Properties of Local Coordinates Based on Dirichlet Tesselations. In *Geometric Modelling*, pages 227–239, 1992.

[PLK$^+$06] S.W. Park, L. Linsen, O. Kreylos, J.D. Owens, and B. Hamann. Discrete Sibson Interpolation. In *IEEE Transactions on Visualization and Computer Graphics*, volume 12, pages 243–253, 2006.

[Sib80] R. Sibson. A vector identity for the Dirichlet tessellation. *Mathematical Proceedings of Cambridge Philosophical Society*, 87:151–155, 1980.

[Sib81] R. Sibson. A brief description of natural neighbor interpolation. *Interpreting Multivariate Data*, pages 21–36, 1981.

[Sug99] K. Sugihara. Surface interpolation based on new local coordinates. *Computer Aided Design*, 13(1):51–58, 1999.

[Wat81] D.F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi Polytopes. *The Computer Journal*, 24(2):167–172, 1981.

[Wat92] D.F. Watson. *Contouring - A guide to the analysis and display of spatial data.* Pergamon, 1st edition, 1992.

A Survey of Octree Volume Rendering Methods

Aaron Knoll
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, Utah
knolla@sci.utah.edu

Abstract: Octrees are attractive data structures for rendering of volumes, as they provide simultaneously uniform and hierarchical data encapsulation. They have been successfully applied to compression, simplification, and extraction as well as rendering itself. This paper surveys and compares existing works employing octrees for volume rendering. It focuses specifically on extraction, direct volume rendering, and isosurface ray tracing.

1 Introduction

An octree is a hierarchical binary decomposition of 3-space along its component axes. A conventional octree guarantees regular, non-overlapping node spacing, thus is well-suited as a container for rectilinear scalar field data. How this hierarchical container is used, however, varies with application. For general purposes, octrees allow data to be stored with adaptive levels of resolution, and accessed quickly via a binary hash function. In volume rendering, the octree often mutates to suit the individual technique, commonly delivering occlusion, acceleration, adaptive multiresolution, compression, or a combination of several features. We will examine the role of the octree structure in three different volume rendering systems: extraction and rasterization; direct volume rendering; and interactive ray tracing. As background, we survey several varieties of octree and efficient hashing schemes for their traversal.

2 Octree Structures and Hash Schemes

The first actual octree publication is unclear [Sam90]. Among the first credited works is that of Morton [Mor66] using quadtrees for geographic indexing scheme. The term "quad-tree" was first used by Finkel and Bentley [FB74] for non-uniform point quadtrees. The first definitive octree is credited to Jackins and Tanimoto [JT80], adapting the earlier quadtree work of Hunter and Steiglitz [HS79] to three dimensions.

The principle of recursive eight-fold subdivision of space is common to all octrees. However, several varieties exist depending on the desired application. Different methods exist to store the structure in memory and provide pointers from children to parents. Implementation of the structure in turn affects the behavior of data access. In general, data is retrieved from an octree via a one-way hash function: given a point in octree space, it seeks the leaf node containing that point. This process is often referred to as *point location*. A related problem is *neighbor finding*: given a node deep in the tree, finding adjacent or nearby nodes. Often, hashing is extended such that a hash code can refer to a region of nodes, or coordinates of one node relative to another.

2.1 Pointer Octree and Hashing

In a traditional octree, a parent node stores pointers to all children. A leaf node could be indicated by a eight null pointers, but most commonly it is given a separate pointerless leaf structure, or represented implicitly within the parent node. While the pointer octree carries a higher storage footprint than pointerless varieties, it is generally simpler to hash as each pointer guarantees the path to the desired child octant. In addition, it is possible to exploit memory tricks so that a node in a pointer octree does not require a full 32-bit (or 64-bit) pointer to each of its children.

Point Location Numerous options exist for performing point location on an octree. The simplest consists of applying comparisons at each parent to determine which child octant the point lies in, until this process reaches a leaf. Comparison operators are expensive, however. Another common scheme is to store a horizontal array containing 0-7 octant indices at each element, in order of increasing depth. Thus, the traversal path is given explicitly in this locational array. The locational array can be built from simple integer coordinates by a process called interleaving proposed by Morton [Mor66]. Array-based keys were popular in early implementations of octrees [Gar82, HS79, JT80].

More recently, Frisken and Perry [FP02] observed that octree and quadtree locational codes could be simply represented by a vector of integer coordinates. Specifically, given a maximum octree depth d_{max} and $x, y, z \in ([0, 2^{d_{max}}] \cap N)^3$, one can hash directly to the correct child node until a leaf is reached. This is accomplished by a bitwise & of each x, y, z component with a 1-bit mask corresponding to the current depth of the hash function. As a result, point location can be performed by simply casting a vector to integer and hashing to the desired node. It is important to note that while Frisken and Perry did not invent the locational code, they were the first to recommend composing the code directly from integer coordinates during hash operation.

Neighbor Finding The problem of finding adjacent neighbors was addressed early on in octree research. Gargantini [Gar82] provides a concise algorithm for finding adjacent nodes in the linear quadtree, using Morton's [Mor66] array-based locational code. Samet [Sam82] proposed a more general scheme for adjacent neighbor-finding, using only an original node and a desired direction, and no knowledge of a locational code. The algorithm was later applied to accelerating ray tracing using a pointer octree [Sam89].

Samet [Sam90] conducts rigorous analysis of the neighbor finding technique, decomposing the algorithm into cases for edge, vertex and aligned neighbors. He concludes that the average cost to find an adjacent neighbor at the same depth is $O(1)$.

While neighbor-finding is demonstratably less complex than point location from the root, Samet's algorithms for pointer octrees without locational codes consist of case decompositions that could be highly optimized. Frisken and Perry noted that the "reflection" function for adjacent neighbors, implemented by Samet via a switch statement, can be accomplished by a binary exclusive "or" of the current location and the direction in which we seek a neighbor. [FP02]. Then, the neighbor itself can be retrieved by recursing up to the deepest common ancestor, and performing point location to find the target neighbor node.

Region Finding Another common goal in octree and quadtree hashing is a mechanism for recovering all nodes in a certain region; or more simply determining if a given node is within a region. With locational code schemes [Mor66, Gar82, FP02], a region can be described by coverage of implicit parent nodes [Gar82]. This is done by marking lower bits in the locational code with an appropriately high number, indicating that all children form part of that region. Alternately, with a coordinate system similar to that of Frisken and Perry [FP02], regions can more intuitively be described by a minimum and maximum pair of vectors.

Region finding has proven useful in image processing applications; thus far its application to volume rendering on octrees has been limited. However, it could conceivably be used when individual sections of an octree volume require attention, particularly when the region is not perfectly aligned with a parent block in the octree.

2.2 Octree Varieties

Other breeds of octree exist beside conventional pointer octrees, and are suited to various applications. Full descriptions of most octree varieties are given by Samet [Sam90]. For our purposes, we will only examine octree structures that are of interest in volume rendering.

Full Octree In a full octree, it is possible to compute the address of any node based on its location in the tree, and thus pointers are unnecessary. Clearly, however, a full octree is undesirable in most cases where data could be compressed or consolidated; the added space obviates most gains from not storing pointers. A common application of a full octree would be one where the encapsulated data is non-homogenous and universally significant, and where we make use of a coarser-resolution representation in interior nodes of the tree.

Linear Octree The linear octree is a variety of pointerless octree in which only leaf nodes are stored, and allocated contiguously in memory. This method was originally proposed by Gargantini for quadtrees [Gar82]. Linear octrees make use of an interleaved base-8 code similar to that proposed by Morton for traditional octrees [Mor66]. The important difference lies in storage: rather than connect the leaf nodes via interior nodes, the leaf nodes are sorted by locational code and then laid out sequentially in memory. Then, rather than matching the Morton code segment with the correct child at each depth of the octree, point location consists of a binary search on the sorted array of leaves. This binary search performs node lookup in $O(log_2(L))$ in a tree with L leaf nodes; as opposed to $O(log_8(N))$ complexity for a pointer octree with N total nodes. Except in the case of highly vertical trees with few leaf nodes relative to interior nodes, the linear octree is generally slower to hash. The major advantage is that it requires storage of neither pointers nor interior nodes. In applications where storage is of the utmost importance, and we only care about leaf nodes, the linear octree is an attractive structure.

Thus far, no one has directly applied the linear octree to volume rendering. However, it is an intriguing structure in it potential compression abilities. As compression is one of the main goals of adaptive octree methods on volume data, it is worth mentioning this structure.

Branch-on-need Octree Wilhelms and Van Gelder [WV92] proposee an incomplete pointer octree that subdivides space non-uniformly, the goal being to build an octree with the fewest-possible empty subtrees within interior nodes. In this way, the ratio of nodes to data points (scalars) is minimized, saving space. As it subdivides space non-uniformly, the BONO may require multiple parent nodes at deeper regions of the tree where a traditional octree would only require one. This potentially causes added traversal steps. In addition, the non-uniform nature of subdivision precludes the use of a pure coordinate system (e.g. Frisken and Perry [FP02]) as a direct hashing scheme.

3 Octrees in Volume Rendering

3.1 Extraction

Early interactive volume rendering commonly employed isosurface extraction via marching cubes [LC87] or a similar variant; paired with z-buffer rasterization of the resulting mesh. Before the widespread availability of GPU's, the goal was to use the octree to simplify the extraction process. In general, the purpose of the octree was to provide a structure with a small memory footprint that could encapsulate cells of a volume, and from which a mesh could be extracted.

Wilhelms and Van Gelder Perhaps the first application of an octree in isosurface extraction was by Wilhelms and Van Gelder [WV92]. Their system employed the aforementioned branch-on-need octree (BONO) to minimize the number of nodes stored relative to cells in the extraction phase. The main downside of this choice, sacrifice in speed of cell location, was a non-issue: the algorithm was bound by extraction, not octree structure traversal.

The Wilhelms and Van Gelder system is best known as the first application of a min/max tree for acceleration. While not explicity mentioned in their paper, they make use of a dual relationship between voxels of the volume and cells from which surface points are extracted. Specifically, a cell is composed of eight voxels, and always indexed by a voxel at a single corner. In the octree, voxels are grouped into eights at the maximum depth level. The min/max tree, then, is built by examining 27 voxels: the original voxels, and their neighbors that constitute the far boundaries of the cells.

During extraction, the marching algorithm need only examine octree nodes when the desired isovalue falls in between the minimum and maximum pair. These nodes can be quickly identified by traversing the octree from top-down as a min/max tree.

Livnat and Hansen Wilhelms and Van Gelder only effectively used their octree to accelerate the extraction process. However, in real-time adaptive surface reconstruction, the same octree structure can be used to order visibility based on the user-specified camera position. This was exploited by Livnat and Hansen. [LH98]. In addition to using the octree to prune empty subtrees (hence, nodes containing no surface), Livnat and Hansen perform visibility culling on nodes that are occluded by previously-traversed nodes closer to the camera. Thus, the octree is used not only to speed extraction, but rasterization as well.

Westermann et al. While the technique of Wilhlems and Van Gelder allowed large regions of 3D scalar data to be systematically ignored, their extraction technique essentially processed leaves at the deepest level of the octree. Westermann et al. [WKE99] recognized that speedups could be gained by exploiting the multiresolution nature of the octree structure; namely by adaptively deciding to extract the mesh from fewer, coarser-resolution nodes corresponding to the same region. This permits higher-resolution volumes to be processed at real-time rates with appropriate level of detail.

The main problem with adaptive surface reconstruction is that cell corner values may vary at junctions in the octree. Such junctions occur when a finer-resolution group of cells is adjacent to a coarser cell. When surfaces are extracted from adjacent cells of different resolution, they are not connected at junctions and therefore cracks appear. A major insight in Westermann et al.'s work is how to patch the resulting meshes to guarantee continuity.

For adaptive multiresolution, the authors built upon previous view-dependent extraction methods (e.g. Livnat et al. [LH98]). They proposed a single pass of the octree, building an "oracle" structure that determines the level of detail, hence the depth of the octree to be traversed, based on view-dependent and geometric criterion. Specifically, they implement one "oracle" for predicting maximum surface curvature within each leaf node. Regions of low curvature are represented with fewer polygons, hence allow shallower traversal of the multiresolution octree. Regions of high curvature merit traversal to the finest level available. In addition to curvature, the octree traversal depth is determined by a "focus point" oracle based on distance to the camera view position.

Velasco and Torres Thus far, extraction-based techniques employing octrees used the structure for indexing existing volume data, and accelerating extraction or rasterization.

The advantage to this technique is that cells can be reconstructed from the original grid data without no costly neighbor-finding [WV92]. The disadvantage is that the full original 3D scalar array must be stored in memory, in addition to a separate octree structure.

Velasco and Torres [VT01] propose a format called a *Cells Octree*, which enumerates several types of distinct octree nodes. Internal nodes contain pointers to eight children. “Single-cell leaf” nodes consist of a single cell, bordered by eight scalar values. Finally, “eight-cell” leaf nodes contain eight cells at the maximum depth of the octree, rolled into their parent node. The cells are represented entirely at every level of the octree, and no neighbor-finding is required to reconstruct cells from voxels. Unfortunately, this technique falls short of the compression achievable by storing only single-scalar voxels within an octree. Moreover, the largest volume tested has dimensions 128^3; which is sufficiently small that no compression would be necessary to store even a full octree, in addition to the original uncompressed volume, at the time of its publication.

Worse still, Velasco and Torres propose modifying the original scalar data to smooth values across cells, and thus guarantee continuous surfaces. This approach differs markedly from that of Westermann et al. [WKE99]; from a visualization standpoint it is arguably wrong as the source data is being modified to create a smooth surface. However, one could equally argue that any piecewise linear mesh is itself an inexact representation of an interpolating isosurface, no matter how well refined. In practice, the results of Velasco and Torres appear acceptable, and theirs is the first published application of an octree towards visualization of compressed structured data.

3.2 Direct Volume Rendering

An alternative to rendering a mesh is direct volume rendering (e.g. Levoy [Lev90]), which integrates rays intersecting a volume. While this is slow in ray tracing, it is effective on current GPUs by accumulating gradients across sequential cutting planes of the volume stored as 2D textures. This no longer restricts the viewer to rendering an isosurface, although choosing a singular color map yields a surface approximation if desired.

GPU volume rendering is quite fast; easily permitting medium-sized volumes (up to 512^3) to be viewed at interactive rates. The main problem is that larger volumes are absolutely limited by GPU memory. To bypass this, it is necessary to store the volume out-of-core in CPU main memory, and page it into GPU memory.

Boada et al. The notion of rendering octree-compressed data was first applied by Boada et al. [BNS01], with an important caveat: scalars are not compressed into a pure octree, but rather an octree is built around bricks of a certain size. From these bricks, “cuts” of the octree are reconstructed into textures, and then sent to the GPU for volume rendering. The octree lookup process was a bottleneck in their implementation; they report sub-interactive frame rates for medium-sized (256^3) volumes. Admittedly, however, available GPU memory at time of publication was a fraction of this size.

Ruijters and Vilanova More recently, a similar technique was applied by Ruijters and Vilanova [RV06] with stunning results. Here, the authors do not use an octree for data paging at all; instead they use the octree to speed up rendering of bricks by skipping empty space. The octree itself is stored in main memory and used in an out-of-core preprocess phase, where the desired bricks are decomposed into visible regions (non-empty octree nodes) and those in turn are mapped to triangular cuts of the volume.

The technique of Ruijters and Vilanova demonstrates that GPU volume rendering can scale to large data. However, it still requires that that data remain uncompressed in main memory, and rendered on the GPU using a paging scheme.

3.3 Ray Tracing

Ray casting involves traversing the path of a ray from an origin pixel in our frame buffer to a point on some surface in space. Then, one evaluates a shading model to color that pixel. Ray tracing is more computationally costly than rasterization, however the relative cost decreases as scene complexity rises, as is the case for large volumes. Moreover, recent advances in parallel or coherent ray tracing [PSL$^+$99, WSBW01] allow for interactive rendering rates on moderately powerful CPU hardware. Nonetheless, interactive ray tracing is generally limited to isosurfacing, due to the high complexity of direct volume rendering.

Ray tracing does entail certain advantages, however. As a purely CPU technique, it has access to full system memory and more sophisticated use of branching and caching than a GPU. As such, it can process a pure octree structure without need of a proxy. Moreover, isosurface ray tracing as proposed by Parker et al. [PSL$^+$98] promises topologically correct surfaces within each cell. Although the global surface is only g_0-continuous at cell borders, it is guaranteed to be "correct" with respect to the original data and a forward-differencing stencil.

Before 1995, octrees were popular spatial acceleration structures for general-purpose polygonal ray tracing [Gla84, Sam89, Sun91, GA93]. Afterwards, hierarchical grids and subsequently kd-trees came into favor due to their faster traversal and better adaptability to irregular geometry. Nonetheless, the octree remains potentially well-suited to volumes, whose voxels are uniformly spaced and non-overlapping.

Knoll et al. The authors combine the min/max acceleration functions of the Wilhelms and Van Gelder [WV92] octree with a compressed format similar to the "cells octree" of Velasco and Torres [VT01]. Instead of containing cells as was the choice of the latter authors, Knoll et al. [KWPH06] build the octree directly around voxels. This requires a fast neighbor-finding scheme to retrieve the values of cell corners when ray tracing; for this the authors borrow (and improve upon) the algorithm of Frisken and Perry [FP02]. Ray traversal is performed on the min/max octree structure, which is the same structure encapsulating the voxel data. For this, the authors employ a traversal similar to that proposed by Gargantini and Atkinson [GA93].

Even employing non-coherent single-ray traversal techniques, Knoll et al. achieve interactive frame rates on multicore architectures. Performance is underwhelming compared to that of GPU volume renderers; however it allows extremely large and time-variant data to be rendered that otherwise would be difficult to accomodate even with a GPU paging scheme. Octree volumes as proposed by Knoll et al. generally allow volumes to be losslessly compressed into 10-30% their original size, even though they are ordinary pointer octrees and optimized more for fast traversal than maximum compression.

4 Conclusion

In conclusion, octrees are useful for a variety of purposes in volume rendering. In extraction and direct volume rendering they traditionally serve the purposes of acceleration and adaptive multi-resolution representation. For interactive ray tracing applications, it is possible to employ a pure octree volume structure and render extremely large volume data in compressed format. Future applications of octree volume rendering could attempt to combine the pure octree volume with GPU rendering approaches, using out-of-core methods.

References

[BNS01] Imma Boada, Isable Navazo, and Roberto Scopigno. Multiresolution Volume Visualization with a Texture-Based Octree. *The Visual Computer*, 17(3), 2001.

[FB74] R.A. Finkel and J.L Bentley. Quad trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4(1):1–9, 1974.

[FP02] Sarah F. Frisken and Ronald N. Perry. Simple and Efficient Traversal Methods for Quadtrees and Octrees. *Journal of Graphics Tools*, 7(3), 2002.

[GA93] Irene Gargantini and H.H. Atkinson. Ray Tracing an Octree: Numerical Evaluation of the First Interaction . *Computer Graphics Forum*, 12(4):199–210, 1993.

[Gar82] Irene Gargantini. An Effective Way to Represent Quadtrees. *Communications of the ACM*, 25(12):905–910, 1982.

[Gla84] Andrew S. Glassner. Space Subdivision For Fast Ray Tracing. *IEEE Computer Graphics and Applications*, 4(10):15–22, 1984.

[HS79] G.M. Hunter and K. Steiglitz. Operations on Images Using Quad Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):145–153, 1979.

[JT80] C.L. Jackins and S.L Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 14(3):249–270, 1980.

[KWPH06] Aaron Knoll, Ingo Wald, Steven Parker, and Charles Hansen. Interactive Isosurface Ray Tracing of Large Octree Volumes. Technical Report UUCS-2006-026, University of Utah, School of Computing, 2006.

[LC87] William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics (Proceedings of ACM SIGGRAPH)*, 21(4):163–169, 1987.

[Lev90] Marc Levoy. Efficient Ray Tracing for Volume Data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.

[LH98] Yarden Livnat and Charles D. Hansen. View Dependent Isosurface Extraction. In *Proceedings of IEEE Visualization '98*, pages 175–180. IEEE Computer Society, October 1998.

[Mor66] G.M. Morton. A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing. *IBM Ltd.*, 1966.

[PSL$^+$98] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen, and Peter-Pike Sloan. Interactive Ray Tracing for Isosurface Rendering. In *IEEE Visualization*, pages 233–238, October 1998.

[PSL$^+$99] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen, and Peter-Pike Sloan. Interactive Ray Tracing. In *Proceedings of Interactive 3D Graphics*, pages 119–126, 1999.

[RV06] Daniel Ruijters and Anna Vilanova. Optimizing GPU Volume Rendering. *Winter School of Computer Graphics, Pilzen*, 2006.

[Sam82] Hanan Samet. Neighbor finding techniques for images represented by quadtrees. *Computer Graphics and Image Processing*, 18(1):35–57, 1982.

[Sam89] Hanan Samet. Implementing Ray Tracing with Octrees and Neighbor Finding. *Computers and Graphics*, 13(4):445–60, 1989.

[Sam90] Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Publishing Company, 1990.

[Sun91] Kelvin Sung. A DDA Octree Traversal Algorithm for Ray Tracing. In Werner Purgathofer, editor, *Eurographics '91*, pages 73–85. North-Holland, September 1991.

[VT01] Francisco Velasco and Juan Carlos Torres. Cell Octree: A New Data Structure for Volume Modeling and Visualization. *VI Fall Workshop on Vision, Modeling and Visualization*, pages 665–672, 2001.

[WKE99] Rüdiger Westermann, Leif Kobbelt, and Tom Ertl. Real-time Exploration of Regular Volume Data by Adaptive Reconstruction of Iso-Surfaces. *The Visual Computer*, 15(2):100–111, 1999.

[WSBW01] Ingo Wald, Philipp Slusallek, Carsten Benthin, and Markus Wagner. Interactive Rendering with Coherent Ray Tracing. *Computer Graphics Forum*, 20(3):153–164, 2001. (Proceedings of Eurographics).

[WV92] J Wilhelms and A Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.

Methods for Presenting Statistical Information: The Box Plot

Kristin Potter
University of Utah
School of Computing
Salt Lake City, UT
kpotter@cs.utah.edu

Abstract: The display of statistical information is ubiquitous in all fields of visualization. Whether aided by graphs, tables, plots, or integrated into the visualizations themselves, understanding the best way to convey statistical information is important. Highlighting the box plot, a survey of traditional methods for expressing specific statistical characteristics of data is presented. Reviewing techniques for the expression of statistical measures will be increasingly important as data quality, confidence and uncertainty are becoming influential characteristics to integrate into visualizations.

1 Introduction

Understanding datasets is essential to the scientific process. However, discerning the significance of data by looking only at their values is a formidable task. Descriptive statistics are a quick and concise way to extract the important characteristics of a dataset by summarizing the distribution through a small set of parameters. Typically, median, mode, mean, variance, and quantiles are used for this purpose. The main goal of descriptive statistics is to describe quickly the characteristics of the underlying distribution of a dataset through a simplified set of values. Often these parameters provide insights into the data that would otherwise be hidden. In addition, these data summaries facilitate the comparison of multiple datasets.

Methods for visually presenting summary statistics include tables, charts, and graphical plots. Graphical plots are interesting in that they pictorially convey a large amount of information in a concise way that allows for quick interpretation and understanding of the data. There are many graphical ways to present descriptive statistics, so covering all of those methods here would be impractical. This survey will focus on one of the most common techniques for summarizing data, the box plot. In addition to various ways to construct the standard box plot, modifications which increase the amount of information presented in the plot will be discussed.

2 The Box Plot

The box plot has become the standard technique for presenting the *5-number summary* which consists of the minimum and maximum range values, the upper and lower quartiles, and the median. This collection of values is a quick way to summarize the distribution of a dataset. In addition, this reduced representation afforded by the 5-number summary provides a more straightforward way to compare datasets, since only these characteristic values need to be analyzed.

The typical construction of the box plot, which can be seen in Figure 1a, partitions a data distribution into quartiles, that is, four subsets with equal size. A box is used to indicate the positions of the upper and lower quartiles; the interior of this box indicates the *innerquartile range*, which is the area between the upper and lower quartiles and consists of 50% of the distribution. Lines (sometimes referred to as whiskers) are extended to the extrema of the distribution, either minimum and maximum values in the dataset, or to a multiple, such as 1.5, of the innerquartile range [FHI89] to remove extreme outliers. Often, outliers are represented individually by symbols; this type of plot is sometimes referred to as a schematic plot [Tuk77]. Finally, the box is intersected by a crossbar drawn at the median of the dataset. The width and fill of the box, the indication of outliers, and the extent of the range-line are all arbitrary choices depending on how the plot is to be used and the data it is representing.

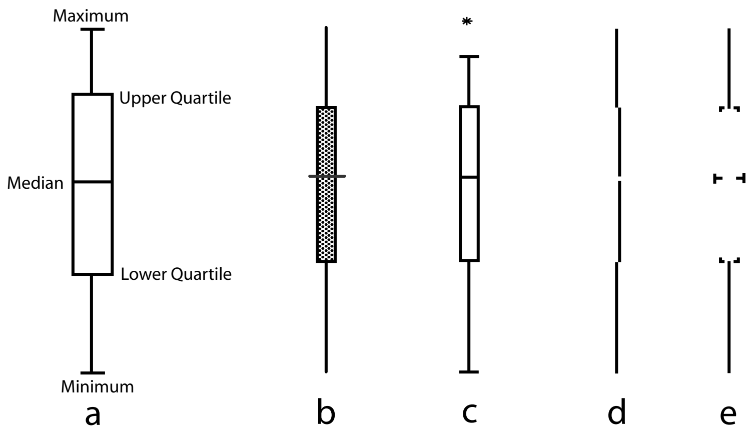


Figure 1: a) The anatomy of a box plot. b-e) Variations of the Box Plot. b) The range-bar chart. c) The box plot. d) The quartile plot. e) The abbreviated box plot.

2.1 Origins

The origins of the box plot can be traced to the range-bar chart. Haemer [Hae48] suggested the use of range-bar charts not only for the comparison of ranges of data, but also for expressing central measures such as median, mean, mode, standard deviation, and tolerance limits through annotations on the chart. This idea was extended to displaying the 5-number summary on the range-bar chart [Spe52], as seen in Figure 1b, by shortening the bar to encompass only the central 50% of the data, using a thin line to indicate the entire range, and a perpendicular line to show the median. This is the first appearance of the form of the box plot we know today. The Tukey box plot became a popular representation and was introduced in 1977 [Tuk77], Figure 1c. This plot truncated the length of the range-line to 1.5 times the length of the innerquartile range. Outliers are indicated by independently marking them on the plot. The look of Tukey's box plot is also refined from that of the range-bar chart. The box fill is removed and the end of the range-line is clearly marked. The visual refinement of the box plot continued with the introduction of the quartile plot [Tuf83], Figure 1d, which sought to reduce visual clutter and maximize the ink-to-paper ratio by removing the box completely, and indicating the innerquartile range by an offset line. The median is simply a break in the innerquartile line. Other versions of this plot indicate the median using a small square and remove the innerquartile line, letting the empty space between the two range-line segments represent the central quartiles. While these plots do reduce the amount of ink used to indicate the 5-number summary, they may also reduce the ease in interpretation of the plot due to the subtle way that the median is indicated, and the similar technique used to show both the range of the data, and the innerquartile range. Furthermore, reducing the amount of area taken up by the innerquartile representation is counterintuitive since this region contains the majority of the data, a fact which the plot should clearly express. The abbreviated box plot [PKR06], as seen in Figure 1e, is another approach which reduces the ink needed to convey the 5-number summary, specifically for the purpose of superimposing further summary statistics on top of the plot. This method maintains the original form of the box plot, but removes the sides of the box, leaving only the corners.

3 Modifications of the Box Plot

One of the major advantages of the box plot is its simplicity of design. Critical information about a dataset is quickly expressed, and the box itself is a signature of the distribution. General characteristics such as the symmetry of the distribution, the location of the central value, and the spread of the observations are immediately apparent. This concise representation allows for the inclusion of additional information about the dataset, and permits the user to customize the plot for specific purposes.

3.1 Density Information

One of the most common types of information added to the box plot is a description of the distribution of the data values. The box plot summarizes the distribution using only 5 values, but this overview may hide important characteristics. For instance, the modality (or number of most often occurring data values) of a distribution is hidden by the box plot, and distinctive distributions with varying modality may be encoded using similar looking

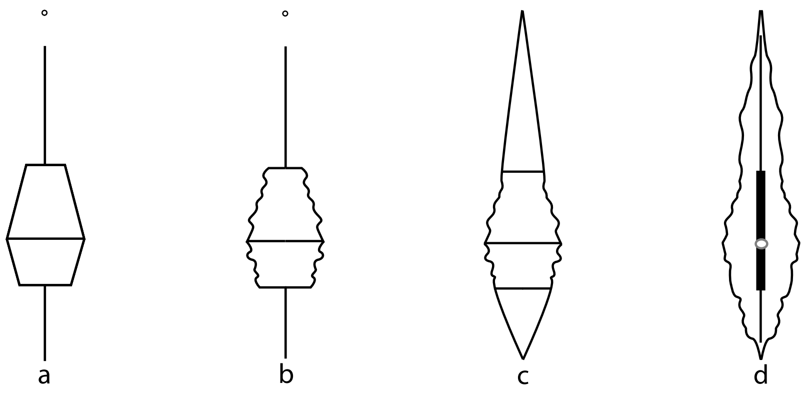


Figure 2: Examples of methods for adding density to the box plot. The a) histplot, b) vaseplot, c) box-percentile plot, and d) violin plot.

box plots. This is especially problematic when no prior information is known about the distributions, as comparing distributions with differing modalities may not be appropriate. One solution to these types of problems is to add into the box plot indications of the density of underlying distribution.

The histplot [Ben88], Figure 2a, is a simple approach for adding density information to a box plot. In the histplot, the density of the distribution is estimated at the median and the two quartiles. The width of the box plot at these locations is then modified to be proportional to the density estimation, and lines are drawn to connect these widths, essentially changing the box of the box plot into a polygon. The histplot adds a quick summary of the density of the central area of the distribution, but it is still possible for important features to be missed. The vaseplot [Ben88], Figure 2b, is a refined version of the histplot which adds in estimated densities for every point between the upper and lower quartiles. A line is drawn between each density estimation point (on both sides), and the polygon of the hist plot is replaced with something that, depending on the distribution, resembles a vase. This modification explicitly shows the density of the central 50% of the data. In addition, confidence intervals can be added to both of these plots by superimposing a light gray shaded bar over the median with the height of the bar signifying confidence.

The box-percentile plot [EB03], Figure 2c, is another method for adding the empirical cumulative distribution of the dataset into the box plot. In this type of plot, both sides of the box plot are used to plot the percentile of the distribution at each point. Thus, for each position in the plot, the width of the box is proportional to the percentile of that data value, up to the 50th percentile, at which point the width is switched to being proportional to minus the percentile. The sides of the plot are symmetric and the 25th, median, and 75th percentiles are marked with a line. The advantages of this plot are that there is no question as to how it should be drawn, it covers the entire range of data, and it does not use any arbitrary choices for its creation. Additionally, the plot is straightforward enough to be understandable by untrained readers, but includes details for trained readers.

The violin plot [HN98], Figure 2d, combines the standard box plot with a density trace to exploit the information contained in both types of diagrams. The box plot is used to show the innerquartile range, however, it is modified in two ways. The first modification changes the box plot by making the box solid black and replacing the median line with a circle; this allows for quick identification of the median and easy comparisons. The second modification removes the individual symbols for outlying data values since the outliers are contained in the density trace and individual points would clutter the diagram. A density trace is added as an alternative density estimator to the histogram and gives a smoother indication of frequency by allowing the intervals in which density is calculated to overlap, in contrast to the histogram. The density trace [CCKT83] at value y, $f(y)$ is defined by

$$f(y) = \frac{1}{hn}\sum_{i=1}^{n} W(\frac{y-y_i}{h}), \quad \text{where} \quad W(u) = \{ \begin{array}{c} 1 \text{ if } |u| \leq \frac{1}{2}, \\ 0 \text{ otherwise} \end{array}$$

where n is the sample size, and h is the interval width. The trace is added to the violin plot as two symmetric curves on either side of the box plot, making the density and magnitude easy to see. The main factor that controls the look of the density trace is the size of the interval width h. There is no specific size that works best in every situation, but an h value around 15% of the data range often produces good results; and the h values should stay between 10 and 40% of the data range to maintain a pleasing smoothness of the density trace curve.

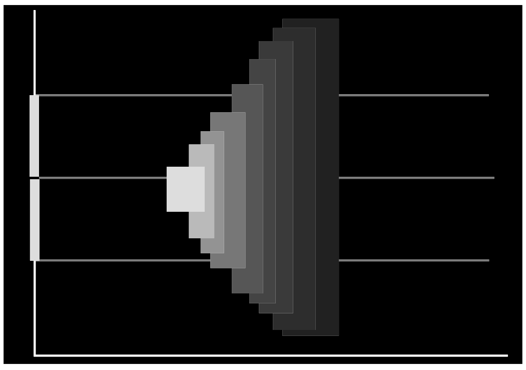


Figure 3: The sectioned density plot.

The sectioned density plot [CC06], Figure 3, exploits characteristics of the human visual system to present, in implied 3D, shape information of a data distribution and trends in variance and central tendency. The human visual system is capable of using occlusion and intensity variation as cues to spatial depth. The sectioned density plot uses these cues to display the distribution of a dataset in order to create the illusion of 3D. To create a sectioned density plot, the data is partitioned into fixed-width intervals, the number of which is variable. Each of these intervals is plotted onto a black background. From lowest density to highest, each interval is plotted using a rectangle shifted slightly to the left, occluding the previous interval, and filled with a monotonically increasing intensity. The 5-number summary is incorporated into these plots by using the coordinate axis to show the range of the values, indicating the upper and lower quartiles with thin rectangles superimposed on the axis, and the median as a break in the range line. Each of these values is extended through the graph as a thin white line.

3.2 Additional Descriptive Statistics

Often there are instances in which the 5-number summary is not enough information, however adding a density plot is not feasible or necessary. For instance, when doing a comparison of multiple datasets, adding the density distribution of each dataset may clutter the plot, however, it would be useful to have information such as the relative number of observations. Additional information may also reduce the possibility of the user making false conclusions.

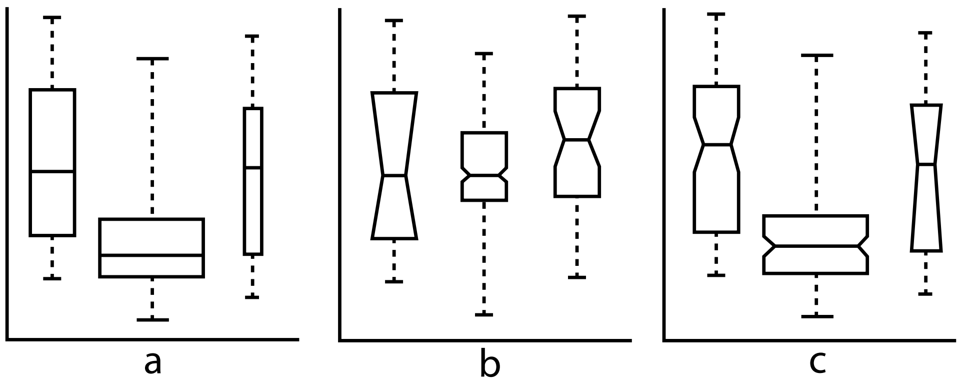


Figure 4: Variations of the box plot. a) Variable width box plot. b) Notched box plot. c) Variable width notched box plot.

McGill et al. [MTL78] suggested a few minor modifications of the original box plot to address these issues. The first variant is the variable width box plot which can be seen in Figure 4a. This plot uses the width of the box to proportionally encode the size of the dataset. The addition of this size clue easily alerts the viewer to distinctions in the number of observations in each dataset and can help the viewer avoid misinterpretation. The second variant proposed is the notched box plot as shown in Figure 4b. In this plot, notches are added to the box plot to roughly indicate the significance of differences between values or the confidence level of the data. The last proposed plot is the variable width notched plot which combines the information contained in the previous two plots and can be seen in Figure 4c.

One of the drawbacks of the simplicity of the box plot is that the box plot can hide distinguishing features of a distribution, and possibly encode very different distributions in similar plots. The addition of density information tried to solve this problem, but it is not always feasible to add in this (possibly) large amount of data. An alternative to using density information is to use statistics that describe specific characteristics of a distribution.

An example of adding descriptive statistics to the box plot is the addition of skew and kurtosis measures. Skew and kurtosis are statistics which describe the symmetry and peakiness of the distribution and can indicate modality. One method for adding these measures into a box plot thickens the sides of the box when these measures indicate skew or high kurtosis in a specific direction [CM05], Figure 5a. The topmost plot indicates that the distribution is skewed toward the right. A bimodal distribution would be skewed in both directions, and this is shown in the center plot in which both ends of the box plot are thickened. Finally, a distribution that is centrally peaked has the median line of the box plot thickened, as shown in the bottom plot. This technique quickly conveys an indication of these statistics and can be used to distinguish between differing distributions.

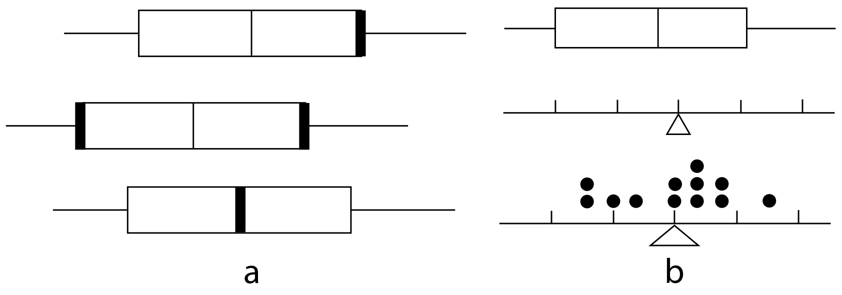


Figure 5: a) Box plots with varying skew and kurtosis. From top to bottom: right-skewed, bimodal, and centrally peaked distributions. b) A Beam and Fulcrum diagram. A dot plot is added to the bottom figure to indicate frequency and the size of the fulcrum base shows the width of a confidence interval.

The beam and fulcrum display [DT00] is a complementary diagram to the box plot and this combination can be seen as the two diagrams at the top of Figure 5b. In this type of display, the range is represented as a line (or beam) and the fulcrum, represented as a triangle, is placed at the mean. On each side of the fulcrum, tick marks are used to show standard deviation points. As seen in Figure 5b, bottom, a dot plot can be added to the beam and fulcrum display to show the frequency of data values, and the size of the fulcrum base can be modified to express the width of a confidence interval. The benefits of such a diagram when presented alongside a box plot are that the user is able to quickly pick out non-normal distributions (i.e., when the mean and median are not equal), see where the data are distributed with respect to the standard deviation scale ($\pm\sigma$, $\pm 2\sigma$, ...) , and easily find outliers, (i.e., data points outside 3 standard deviations). It is also a useful learning tool, students can easily understand that the mean balances the distribution.

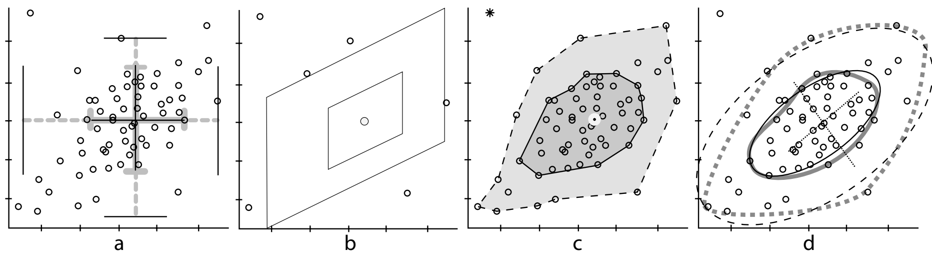


Figure 6: Bivariate extensions of the box plot. a) Rangefinder box plot. b) The Bagplot. c) The 2D box plot. d) The Quelplot and Relplot.

3.3 Bivariate Extensions

Standard implementations of the box plot focus on univariate data distributions. The 5-number summary is a useful descriptor of not only univariate, but also bivariate data distributions. The main challenge in extending the box plot for use with higher dimensional data is how to translate the 5-number summary values, which are vector values in the bivariate case, into visual metaphors with meaningful spatial positions, while maintaining the simplicity of the original box plot. A bivariate box plot can show not only the location and a summary of the data distribution, but also skew, spread and correlation.

A rangefinder box plot [BG87], as seen as the solid back lines in Figure 6a, is a simple extension of the box plot into 2D. To create a rangefinder box plot, all data values are plotted as points on a 2D graph (this is often called a scatterplot). For each variable, the 5-number summary is calculated, a line segment is drawn along the innerquartile range and perpendicular lines are placed at the adjacent values of the variable, where the 1D box plot would terminate. The intersection of the two central line segments is the cross-median value. This idea was further improved upon, as shown as the thick gray lines in Figure 6a, to emphasize the quartiles rather than the range, by moving the perpendicular lines from the adjacent values to the upper and lower quartile positions and extending whisker lines to the extrema value of the variable [Len88]. These extensions of the box plot into 2D are an unobtrusive expression of the summary of each variable, but the *correlation* between the two variables is not visible.

Other techniques for extending the box plot into 2D all use the notion of a *hinge* that encompasses 50% of the data and a *fence* that separates the central data from potential outliers. The distinctions between each of these methods are the way the contour of the hinge and fence are represented, and the methods used to calculate the contours.

The two-dimensional box plot [Ton05], as seen in Figure 6b, computes a robust line through the data by dividing the data into three partitions, finding the median value of the two outer partitions and using these points as the line. Depending on the relationship between the slope of the line and each variable, the quartile and fence lines are drawn either parallel to the robust line, or parallel to the variable's coordinate axis. The lines not comprising the outer-fence and the inner-hinge boxes are removed.

The bagplot [RRT99] uses the concept of halfspace depth to construct a bivariate version of the box plot, as seen in Figure 6c. The halfspace depth $ldepth(\theta|Z)$ of some point θ is the smallest number of data points $z_i \in Z = z_1, z_2, ..., z_n$ contained in any closed halfplane with a boundary line through θ. The depth region D_k, which is a convex polygon, is the set of all θ with $ldepth(\theta|Z) > k$ and $D_{k+1} \subset D_k$. To construct the bagplot a scatterplot of the data is first created. The *depth median* is then found which is the θ with the highest $ldepth(\theta|Z)$, if there is only one such θ, otherwise it is the center of gravity of the deepest region. This point, which is at the center of the plot, is represented as a cross. The *bag* is a dark gray region in the plot encompassing 50% of the data. The *fence* separates the outliers of the dataset, but is not drawn, and the *loop* is a light gray region of the plot that contains points outside of the bag, but inside of the fence. Outliers are highlighted as black stars. Options for reducing the visual clutter of the bagplot are to not plot data points contained in the bag, and to not fill the regions contained in the bag and the loop, but instead surround the bag with a solid line and the loop with a dashed line. In addition, a confidence region can be added into the bagplots as a *blotch* drawn around the depth median.

The relplot and the quelplot [GI92] use concentric ellipses to delineate between the hinge and fence regions. Both the relplot and quelplot can be seen in Figure 6d. The relplot uses full ellipses which assume symmetric data and are constructed using a robust estimator such as the minimum volume ellipsoid. In the figure, the relplot is shown as ellipses drawn in thin black lines. The quelplot divides the ellipses into four quarters aligned on the major and minor axes, and computed using an M-estimator. The quelplot is shown in the figure as thick, gray lines. The quelplot can show skewed data, since each quarter ellipse can be tranformed individually.

4 Conclusion

The box plot is a standard technique for presenting a summary of the distribution of a dataset. Its use has become prevalent in all forms of scientific inquiry, and understanding its construction, origins, and modifications can help not only with interpretation of the information presented by the box plot, but also in its creation and use. The concise representation provides not only insights to the important characteristics of a distribution, but permits the addition of information which enables the customization of the box plot to specific scenarios. Overall, the simplicity of the box plot makes it an elegant method for the presentation of scientific data.

5 Acknowledgments

I would like to thank Hans Hagen and the IRTG for this opportunity and Richard Riesenfeld for support and guidance, as well as Chuck Hansen, Chris Johnson, Joe Kniss, and A.N.M. Choudhury. This work was supported in part by ARO (DAAD19-01-1-0013), NSF (IIS0218809) and NSF grant (03-1247). All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

References

[Ben88] Yoav Benjamini. Opening the Box of a Boxplot. *American Statistician*, 42(4):257–262, November 1988.

[BG87] Sean Becketti and Willian Gould. Rangefinder Box Plots. *American Statistician*, 41(2):149, May 1987.

[CC06] Dale J. Cohen and Jon Cohen. The Sectioned Density Plot. *American Statistician*, 60(2):167–174, May 2006.

[CCKT83] John M. Chambers, William S. Cleveland, Beat Kleiner, and Paul A. Tukey. *Graphical Methods for Data Analysis*. Wadsworth, 1983.

[CM05] Chamnein Choonpradub and Don McNeil. Can the Box Plot be Improved? *Songklanakarin Journal of Science and Technology*, 27(3):649–657, 2005.

[DT00] David P. Doane and Ronald L. Tracy. Using Beam and Fulcrum Displays to Explore Data. *American Statistician*, 54(4):289–290, November 2000.

[EB03] Warren W. Esty and Jeffery D. Banfield. The Box-Percentile Pot. *Journal of Statistical Software*, 8(17), 2003.

[FHI89] Michael Frigge, David C. Hoaglin, and Boris Iglewicz. Some Implementations of the Box Plot. *The American Statistician*, 43(1):50–54, February 1989.

[GI92] Kenneth M. Goldberg and Boris Iglewicz. Bivariate Extensions of the Box Plot. *American Statistician*, 34(3):307–320, August 1992.

[Hae48] Kenneth W. Haemer. Range-Bar Charts. *American Statistician*, 2(2):23, April 1948.

[HN98] Jerry L. Hintze and Ray D. Nelson. Violin Plots: A Box Plot-Density Trace Synergism. *American Statistician*, 52(2):181–184, May 1998.

[Len88] Russel V. Lenth. Comment on Rangefinder Box Plots. *American Statistician*, 42(1):87–88, February 1988.

[MTL78] Robert McGill, John W. Tukey, and Wayne A. Larsen. Variations of Box Plots. *American Statistician*, 32(1):12–16, February 1978.

[PKR06] Kristin Potter, Joe Kniss, and Richard Riesenfeld. Visual Summary Statistics. *To Appear*, 2006.

[RRT99] Peter J. Rousseeuw, Ida Ruts, and John W. Tukey. The Bagplot: A Bivariate Boxplot. *American Statistician*, 53(4):382–387, November 1999.

[Spe52] Mary Eleanor Spear. *Charting Statistics*. McGraw-Hill Book Company, INC., 1952.

[Ton05] Phattrawan Tongkumchum. Two-Dimensional Box Plot. *Songklanakarin Journal of Science and Technology*, 27(4):860–866, 2005.

[Tuf83] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[Tuk77] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

PointCloudXplore: A Visualization Tool for 3D Gene Expression Data

Oliver Rübel[*,1,2], Gunther H. Weber[2,3], Soile V.E. Keränen[3], Charless C. Fowlkes[4], Cris L. Luengo Hendriks[3], Lisa Simirenko[3], Nameeta Y. Shah[2], Michael B. Eisen[3], Mark D. Biggin[3], Hans Hagen[1], Damir Sudar[3], Jitendra Malik[4], David W. Knowles[3], and Bernd Hamann[1,2]

[1] International Research Training Group "Visualization of Large and Unstructured Data Sets," University of Kaiserslautern, Germany

[2] Institute for Data Analysis and Visualization, University of California, Davis, CA, USA

[3] Life Sciences and Genomics Divisions, Lawrence Berkeley National Laboratory, Berkeley, CA, USA

[4] Computer Science Division, University of California, Berkeley, CA, USA

Abstract: The Berkeley Drosophila Transcription Network Project (BDTNP) has developed a suite of methods that support quantitative, computational analysis of three-dimensional (3D) gene expression patterns with cellular resolution in early *Drosophila* embryos, aiming at a more in-depth understanding of gene regulatory networks. We describe a new tool, called PointCloudXplore *(PCX)*, that supports effective 3D gene expression data exploration.

PCX is a visualization tool that uses the established visualization techniques of *multiple views*, *brushing*, and *linking* to support the analysis of high-dimensional datasets that describe many genes' expression. Each of the views in PointCloudXplore shows a different gene expression data property. Brushing is used to select and emphasize data associated with defined subsets of embryo cells within a view. Linking is used to show in additional views the expression data for a group of cells that have first been highlighted as a brush in a single view, allowing further data subset properties to be determined. In PCX, physical views of the data are linked to abstract data displays such as parallel coordinates. Physical views show the spatial relationships between different genes' expression patterns within an embryo. Abstract gene expression data displays on the other hand allow for an analysis of relationships between different genes directly in the gene expression space. We discuss on parallel coordinates as one example abstract data view currently available in PCX. We have developed several extensions to standard parallel coordinates to facilitate brushing and the visualization of 3D gene expression data.

[*] oliverruebel@web.de

1 Introduction

Animal embryos comprise dynamic 3D arrays of cells that express gene products in intricate spatial and temporal patterns that determine the shape of the developing animal. Biologists have typically analyzed gene expression and morphology by visual inspection of photomicrographic images. Yet, to understand animal development, we need good methods to computationally describe gene expression data. To address this challenge, the BDTNP has developed image analysis methods to extract information about gene expression from imaging data, using early *Drosophila melanogaster* embryos as a model. Confocal image stacks of blastoderm stage *Drosophila* embryos are converted into matrices specifying the position of nuclei and the expression levels of select genes around each nucleus, (see Section 3). The resulting new datasets, called PointClouds, promise to be an invaluable resource for studying development. Since available visualization tools are insufficient for comparing and analyzing 3D PointCloud data, we have developed PointCloudXplore as a tool to help biologists explore these datasets.

During embryogenesis complex regulatory networks are built up as transcription factors cross-regulate the expression of other transcription factors as well as enzymes, structural proteins, etc., guiding the development of animals [WKS^{+}03, Law92, SL05]. Since spatial regulation of gene expression directs animal morphogenesis, a major goal of the BDTNP is to decipher how spatial patterns of target gene expression are directed by the expression patterns of the transcription factors that regulate them. Because gene regulation depends on combinatorial inputs from many transcription factors, simultaneous analysis of many expression patterns is required. Therefore, PCX includes multiple visualization methods to allow specific relationships to be seen within highly complex expression data for many genes.

2 Previous Work

Generally, data can be displayed in multiple formats, or *views*, that each allow different relationships between the data components to be observed. The *linking* of multiple views is a well-established visualization method [BMMS91]. For example, it has been shown that linking abstract data displays, such as scatter plots, with physical data views, such as a 3D model of a catalytic converter, can improve data analysis significantly and provide insight into complex physical phenomena [KSH04, PKH04, DGH03]. Hauser et al. [HLD02] proposed integrating parallel coordinates with physical views for a better understanding of high-dimensional phenomena. Gresh et al. [GRW^{+}00] used parallel coordinates linked to physical views for visualizing biological data sets describing cardiac measurement and simulation experiments.

Parallel coordinates were proposed contemporaneously by Inselberg [Ins84] and Wegman [Weg90] and are a common information visualization technique for high-dimensional data sets. In a parallel coordinate view, a data set consists of a set of *samples*, which in our case are the cells in a *Drosophila* embryo. Each sample (cell) has a set of associated quantities, which in our case are the relative expression levels for multiple genes. Expression data for each gene corresponds to a dimension in the data set, with data for each gene being represented by one of a series of parallel vertical axes. Each sample (cell) defines a *data line*, i.e., a zig-zag line connecting adjacent parallel axes. The intersection point of the data line with each vertical axis corresponds to the value of the sample for the corresponding dimension (i.e., the relative expression level for the corresponding gene in that cell).

Many extensions to standard parallel coordinates have been developed to make them more useful for practical applications. Fua et al. [FWR99] proposed hierarchical parallel coordinates, including several techniques for visualization of selected subsets of the data. Distortion operations, such as dimensional zooming, for example, support a more detailed analysis of data subspaces. Color is widely used for improving parallel-coordinate views since dedicated line coloring eases following the course of data lines. Fua et al. [FWR99] and Novotny [Nov04] proposed the use of color bands for visualization of brushes in parallel coordinate views. Wegman and Luor [WL97] proposed the application of transparency and "over-plotting" translucent data points/lines. This method highlights dense areas while sparse areas fade away, thus revealing inherent data characteristics.

3 Gene Expression Data and Visualization Pipeline

A single PointCloud file contains the x,y,z location of each nucleus in one embryo and the relative concentration of gene products (mRNA or protein) associated to each nucleus [FLHK$^+$05]. These files are created in the following manner (see Figure 1). Embryos are fixed, stained, and mounted, then imaged using a confocal microscope (Figure 1, *IA*). The obtained images are processed to detect all nuclei and measure the associated gene expression levels (Figure 1, *IS*). Embryos are typically labeled with one fluorophore to detect the nuclei, and with two others to detect two gene products. It is not practical to obtain the expression of more than a few genes in a single embryo, due to the limited number of different fluorophores that can be distinguished by the microscope, as well as the difficulty in adding these labels to the embryos. Since it is critical to compare the relationships between transcription factors and many of their target genes in a common co-ordinate framework, a set of PointClouds using both morphology and a common reference gene to determine correspondences (Figure 1, *ER*) are registered into Virtual PointClouds [FLHK$^+$05]. The resulting Virtual PointCloud contains averaged expression levels for many genes mapped on the nuclei of one of the embryos in the set. PCX, see Figure 1, can be used for visualization of both single-embryo PointClouds and Virtual PointClouds.

4 Physical Views

We have developed several physical views (models) of the embryo to support analysis of spatial gene expression patterns. In all these Embryo Views, each cell is represented by

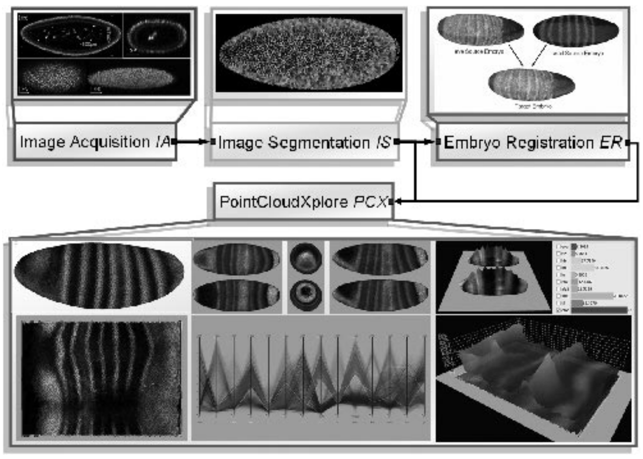


Figure 1: Gene Expression Data and Visualization Pipeline: PCX is used to visualize data from single embryo PointClouds and Virtual PointClouds.

one 3D graphical object, positioned in space according to the physical position of the cell it represents. Gene expression values are visualized using color and also by height, in the case of views using gene Expression Surfaces (Section 4.3), . During the developmental stage that the BDTNP is currently investigating — the blastoderm — all cells studied by the BDTNP are located on a surface in shape very similar to an ellipsoid. Orthographic projection is used to display views from fixed angles that display expression and morphology along the three coordinate axes (Section 4.2). In another view, a cylindrical projection is used to map cells onto a plane to gain a global overview of the entire embryo (Section 4.2). Cells of interest can be selected in any of these views to create a so called *brush*, just by drawing on the surface of the embryo. The selected cells that comprise the brush are highlighted using color. The user can interact with all embryo views via interactive zooming, panning, and rotation.

4.1 3D Physical View

In all our Embryo Views, each cell is represented by a polygon on the embryo surface, using the Eigencrust method [KSO04] for constructing an approximation of the Delauny triangulation of the surface of a PointCloud. The dual mesh of the triangulation is a tesselation similar to a Voronoi diagram, defined on the embryo surface [dBvKOS00]. Each cell is represented by a Voronoi polygon with exactly one original data point in its center. The polygon sizes depend on the cells' distribution in the embryo, whereas the shape of the polygons has no direct meaning. This results in a 3D model of the embryo which provides an intuitive way to look at the data (see Figure 2).

Each polygon is colored according to expression values measured in the cell it represents. The color mapping is based on the HSV color model [FDFH95]. The basic color hue, H,

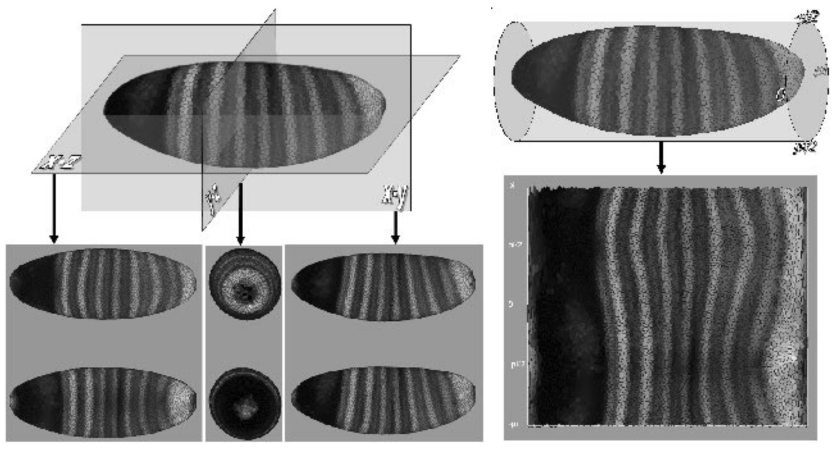


Figure 2: 2D Orthographic View (left); Unrolled View (right).

of each gene is defined by the user. Gene expression values are mapped linearly to color brightness V. Saturation of color is always one, unless specified differently. For each gene, a minimum and maximum value can be defined independently by the user. All expression values below the minimum are mapped to black, and all values above the maximum are mapped to full intensity.

4.2 2D Physical Views

To allow one to obtain a much quick overview of all cells several 2D visualizations of the embryo have been developed. As shown in Figure 2 (left), by centering the main coordinate system within the 3D embryo model, orthographic projection can be used to create three 2D views of the embryo showing the dorsal/ventral, anterior/posterior, and the left/right sides of the embryo. These Orthographic Views provide an overview of all cells while preserving the shape of the embryo as a frame of reference for a biologist. The curvature of the embryo leads to high densities of cells on the projection borders, but provides an impression of depth and shape. A general overview of all cells in an embryo can be obtained by switching between the three different Orthographic Views.

For a scientist who wants an instant overview of the whole blastoderm expression pattern, the Unrolled View uses a cylindrical projection to map all surface cells of the embryo onto a rectangular plane (see Figure 2 (right)). All cells are first projected onto a cylinder, which is unrolled in a plane. In this view, a complete overview of all cells is provided while the relative positions of cells on the anterior/posterior axis and around the embryo are preserved. Due to of the ellipsoid like shape of the embryo, cells in the anterior and posterior of the embryo are distorted by the projection in order to fill the rectangule. Shape and size of cells in the middle part of the embryo are less affected by distortion effects.

All 2D physical views are projections of the original 3D embryo model. To visualize gene expression values and brushes the same color mapping is used as in the 3D physical view (see Section 4.1).

4.3 Expression Surfaces

To support a more quantitative analysis of gene expression data, Expression Surfaces can be defined above either the Orthographic or the Unrolled Views. Each Expression Surface displays data for one gene. The xy positions of Expression Surface points are determined by the positions of cells in the underlying views, whereas the height of an Expression Surface is determined by the expression values measured for the gene it represents. Spatial relationships between several genes' expression patterns can be viewed at once using multiple Expression Surfaces. For example, Figure 3 shows the quantitative relationship between *eve* and *ftz*. The expression levels of these two genes are spatially largely non-overlapping and change relative to one another along each body axis.

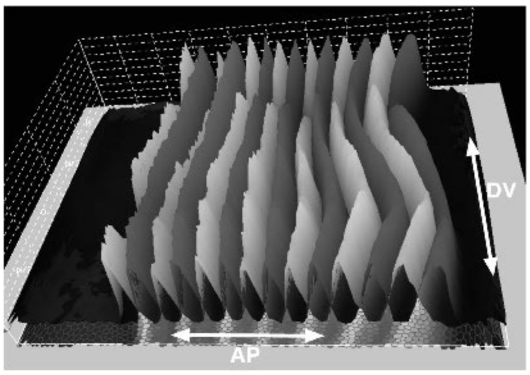


Figure 3: Gene expression surfaces for *eve* (light gray) and *ftz* (dark gray).

5 3D Parallel Coordinates

A limitation of all described Embryo Views is that when more than four or five genes whose expression overlaps are displayed at the same time, it is often not possible to distinguish each gene's expression. Therefore, we have adapted parallel coordinates to create several Parallel Coordinate Views, in which relationships between many genes' expression can be visualized. See Section 2 for an introduction to parallel coordinates. Further, we have linked Parallel Coordinate Views and Embryo Views, ensuring that all brushes defined in the Embryo Views can be displayed in the Parallel Coordinate Views and vice versa. In general, parallel coordinates introduce several other visulaization problems, such as occlusion and cluttering. To reduce these problems, several extensions to standard par-

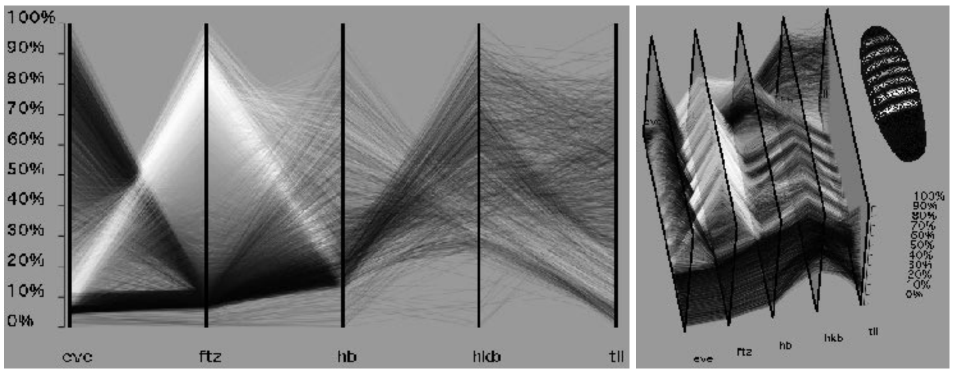
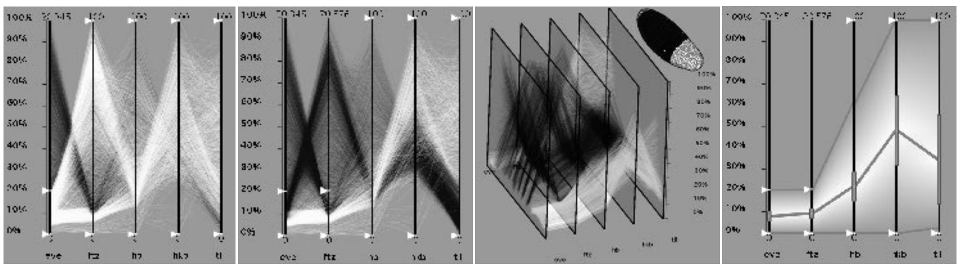


Figure 4: Expression level of four different genes visualized in 2D Parallel Coordinates (left) and 3D Parallel Coordinates (right).

allel coordinates have been developed, which have already been described in more detail in [OGS+06]. By varying color and transparency of data lines it is much easier to gain a fast overview and to detect important features and clusters in the data. Line trace highlighting and animation are additional tools which allow one to follow the course of single data lines through the graph. To allow for detailed analysis of brushes the dimensional zooming technique is used. Brushes define a subspace of the entire gene expression space. By scaling the selected ranges to the entire length of the parallel axis it is possible to analyse details in a user-defined subspace. Statistical properties of brushes such as selected minimum- and maximum values, average expression values, and standard deviations can be analyzed using brush bands (see Fig 5(d)).

Information about the spatial relationships between different genes' expression patterns is essential for the analysis of regulatory networks. Information about relative cell positions along the two main axes of the embryo, anterior/posterior (AP) and dorsal/ventral (DV), can be derived from the Unrolled View described in Section 4.2. To display this information in Parallel Coordinate Views, the coordinate axes have been extruded into the third dimension (Figure 4 (left)). Data lines are ordered from back-to-front according to cell positions along either the AP axis or the DV circumference. Along any given data line, the positional information is constant, such that data lines do not intersect each other in this third dimension. The 3D coordinate axes are drawn highly transparent with active z-buffering to prevent the addition of colors of overlapping parallel axes. This strategy guarantees a complete overview of the entire plot, with no details hidden. In addition, the outer frame of the coordinate axes are drawn with full opacity, which makes it easier to determine the position of the coordinate axis in 3D space.

By using this 3D visualization, spatial data dimensions are clearly separated from gene expression dimensions of the data, and the basic character of the spatial gene expression patterns in one dimension is preserved. For example, if data lines are sorted according to the position of cells along the AP axis, then the stripe patterns of genes like *eve* or *ftz* are visible in the plot (Figure 4 (left)). This 3D view also reveals what is not obvious



(a) (b) (c) (d)

Figure 5: Defining brushes in Parallel Coordinate View. In (a) a brush is defined to exclude all cells expressing *eve* at more than 20%. In (b) this brush is further refined to also exclude all cells expressing *ftz* at a value greater than 20%. (c) shows a 3D Parallel Coordinate View of the brush defined in (b) , where cell locations along the A/P axis of the embryo are shown in the third dimension. (d) Shows a broad color band display of the brush selected in (b), indicating the minimum, maximum, mean, and standard deviations for expression values for each gene.

in the 2D views shown in Figure 4: Cells expressing both *eve* and *ftz* at low levels are mainly at the anterior and posterior of the embryo, and a subset of these cells is found in the principal areas where *hkb* and *tll* are highly expressed. Even if tens of additional gene dimensions were added to the 3D view, these and doubtless other relationships could still be visualized.

Brushing can be executed in parallel coordinates using two sliders attached to each axis to define ranges in gene expression. In 3D parallel coordinates, two additional sliders are available to allow one to select cells also according to their relative AP- or DV position within the embryo. The physical views and the parallel coordinates are synchronized, i.e., if a brush is edited in one view, then the other view is also updated. If, for example, a brush is changed in parallel coordinates then the user can view in parallel how the spatial pattern the brush defines alters in any physical view. In Figure 5, an example for brushing in parallel coordinates is shown. One can see how additional relationships are revealed in 3D parallel coordinates where the brush splits of into two characteristic regions in the anterior and posterior of the embryo (see Figure 5(c)). Visualizing the brush just as broad color band reveals basic statistical properties of the brush (see Figure 5(d)).

6 Conclusions and Future Work

We have introduced PointCloudXplore and described a subset of its views and functionality. Dedicated physical views of the *Drosophila melanogaster* blastoderm (termed Embryo Views) make the comparison and analysis of spatial gene expression patterns possible. Expression Surfaces provide an effective and intuitive way for quantitative analysis of gene expression data. To support analysis of the relationships between genes directly in gene expression space, we have integrated parallel coordinates into the system (Parallel Coordinate Views). Parallel Coordinate Views have been extended to a 3D rendering, making it possible to present spatial and gene expression data dimensions in one plot, while both dimension types are visually separated and basic spatial properties of gene expression patterns are preserved. All views are linked via brushing. PointCloudXplore makes interactive analysis of 3D expression data possible for the first time.

We plan to integrate automatic data analysis tools into PCX. Unsupervised clustering has been used previously to analyze microarray data and can also be applied to 3D gene expression data. Singular value decomposition (SVD) and other techniques have also been used to analyze gene expression and similar data. For analysis of 3D gene expression data, these techniques need to be modified, and new ones developed. Integration of such tools should further improve the utility of PCX.

7 Acknowledgments

This work was supported by the National Institutes of Health (grant GM70444), by the National Science Foundation (CAREER Award ACI 9624034), through the NSF Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and an NSF large Information Technology Research (ITR) grant. Further support was provided by the LBNL Laboratory Directed Research Development (LDRD) program; 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, the members of the BDTNP at the Lawrence Berkeley National Laboratory (LBNL), and the members of the Visualization Group at LBNL.

References

[BMMS91] Andreas Buja, John Alan McDonald, John Michalak, and Werner Stuetzle. Interactive data visualization using focusing and linking. In *IEEE Visualization 1991*, pages 156–163. IEEE Computer Society Press, 1991.

[dBvKOS00] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry*. Springer, second edition edition, February 2000.

[DGH03] Helmut Doleisch, Martin Gasser, and Helwig Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Data Visualization 2003 (Proceedings of the EUROGRAPHICS - IEEE TCVG Symposium on Visualization 2003)*, pages 239–248. Eurographics Association, 2003.

[FDFH95] James D. Foley, Andries Van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principle and Practice*. Addison-Wesley, second edition in c edition, July 1995.

[FLHK$^+$05] Charless Fowlkes, Cristian L. Luengo Hendriks, Soile Vanamo Elisabeth Keränen, Mark D. Biggin, David W. Knowles, Damira Sudar, and Jitendra Malik. Registering Drosophila Embryos at Cellular Resolution to Build a Quantitative 3D Map of Gene

Expression Patterns and Morphology. In *CSB 2005 Workshop on BioImage Data Minning and Informatics*, August 2005.

[FWR99] Ying-Huey Fua, Matthew O. Ward, and Elke A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *IEEE Visualization 1999*, pages 43–50. IEEE Computer Society Press, 1999.

[GRW$^+$00] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung. WEAVE: A system for visually linking 3-D and statistical visualizations, applied to cardiac simulation and measurement data. In *IEEE Visualization 2000*, pages 489–492. IEEE Computer Society Press, 2000.

[HLD02] Helwig Hauser, Florian Ledermann, and Helmut Doleisch. Angular Brushing of Extended Parallel Coordinates. In *IEEE Symposium on Information Visualization (InfoVis'02)*, pages 127–130. IEEE Computer Society Press, 2002.

[Ins84] Alfred Inselberg. Parallel Coordinates for Multidimensional Displays. In *Spatial Information Technologies for Remote Sensing Today and Tomorrow, The Ninth William T. Pecora Memorial Remote Sensing Symposium*, pages 312–324. IEEE Computer Society Press, 1984.

[KSH04] Robert Kosara, Gerald N. Sahling, and Helwig Hauser. Linking Scientific and Information Visualization with Interactive 3D Scatterplots. In *Short Communication Papers Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG)*, pages 133–140, 2004.

[KSO04] Ravikrishna Kolluri, Jonathan R. Shewchuk, and James F. O'Brien. Spectral Surface Reconstruction from Noisy Point Clouds. In *Symposium on Geometry Processing*, pages 11–21. ACM Press, July 2004.

[Law92] Peter A. Lawrence. *The Making of a Fly: The Genetics of Animal Design*. Blackwell Science, 1992.

[Nov04] Mateo Novotny. Visually Effective Information Visualization of Large Data. In *Proceedings of Central European Seminar on Computer Graphics (CESCG)*, 2004. Online at `http://www.cg.tuwien.ac.at/studentwork/CESCG/CESCG-2004/`.

[OGS$^+$06] Ruebel O., Weber G.H., Keraenen S.V.E., Fowlkes C.C., Luengo Hendriks C.L., Simirenko L., Shah N.Y., Eisen M.B., Biggin M.D., Hagen H., Sudar J.D., Malik J., Knowles D.W., and Hamann B. PointCloudXplore Visual analysis of 3D gene expression data using physical views and parallel coordinate. In *Sousa Santos, B., Ertl, T. and Joy, K.I., eds., Data Visualization 2006 (Proceedings of EuroVis 2006), Eurographics Association*, pages 203–210, 2006.

[PKH04] Harald Piringer, Robet Kosara, and Helwig Hauser. Interactive Focus+Context Visualization with Linked 2D/3D Scatterplots. In *Proceedings of the 2nd International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2004)*, July 2004.

[SL05] Angelike Stathopoulos and Michael Levine. Genomic Regulatory Networks and Animal Development. *Developmental Cell*, 9(4):449–462, 2005.

[Weg90] Edward J. Wegman. Hyperdimensional Data Analysis Using Parallel Coordinates. *Journal of the American Statistical Association*, 85(411):664–675, September 1990.

[WKS$^+$03] Katrin Weigmann, Robert Klapper, Thomas Strasser, Christof Rickert, Gerd M. Technau, Herbert Jäckle, Wilfried Janning, and Christian Klämbt. FlyMove - a new way to look at development of Drosophila. *Trends in Genetics 19*, pages 310–311, September 2003. Online at `http://flymove.uni-muenster.de/`.

[WL97] Edward J. Wegman and Qiang Luo. High dimensional clustering using parallel coordinates and the grand tour. *Computing Science and Statistics*, 28:361–368, 1997.

Topographic Distance Functions for Interpolation of Meteorological Data

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

1. University of Kaiserslautern, Germany, {lehner, umlauf}@informatik.uni-kl.de

2. University of California, Davis, USA, {bhamann, slustin}@ucdavis.edu

Abstract: The reference evapotranspiration ET_0 is an important meteorological quantity in agriculture and water resource management. It is usually estimated from other meteorological quantities measured at weather stations. To estimate ET_0 at an arbitrary geographical position these quantities must be interpolated. The Center for Spatial Technologies And Remote Sensing (CSTARS) at UC Davis uses the DayMet approach for this task.

We discuss some inconsistencies within the DayMet approach and suggest improvements. One significant problem of DayMet is the lack of consideration of terrain topography. We define new distance functions that are elevation-dependent and show preliminary results of the comparison of the classic and the improved DayMet approach.

1 Introduction

To set up near-optimal irrigation schedules the amount of water that has evaporated or transpired into the atmosphere during the day must be known. This information is important to farmers that have to replace this amount of water to maintain appropriate availibility for crop growth and to administrators of water management systems so they can provide adequate water supplies for agricultural and urban needs. Also this information can be used for setting up a land use plan as a basis for decisions.

One important quantity is the evapotranspiration that is the combination of evaporation, i.e., the loss of water from the surface of the plants and the soil and transpiration, i.e., the loss of water from inside the plants to the atmosphere. This quantity depends on several factors, such as weather variables, soil conditions, and the type of vegetation.

For determining evapotranspiration ET for a certain region, a reference evapotranspiration ET_0 is defined as the evapotranspiration above a defined reference vegetation (uniform closely-cropped grass), and therefore only depends on the weather conditions. The evapotranspiration ET_c for a specific vegetation or surface type is

$$ET_c = K_c \cdot ET_0, \tag{1}$$

where K_c is the crop coefficient and can be determined from a look up table.

To estimate a spatially distributed ET_0 for the state of California, the California Department of Water Resource and the University of California, Davis developed the CIMIS project (California Irrigation Management Information System). They established about 120 automated weather stations all over California, each of which measures several climate values (such as solar radiation, relative humidity, wind speed, temperature) under defined reference conditions (2m above a dense grass surface). This data is collected and stored in a database. From these measured values ET_0 can be estimated.

This approach leads to estimated values for ET_0 only for the locations of the CIMIS weather stations. For all other places, the weather values have to be estimated by combining the measured values of nearby weather stations (interpolation), and then ET_0 can be estimated from these.

In the CIMIS project, a map is created that contains the estimated value for every point on a dense grid with grid distance of 2km ([HBT$^+$06]). To find estimates for each grid point, two different methods are used: Some of the weather values are interpolated using regularized splines with tension ([MM93], [MH93], [HPMM02]), for others the DayMet interpolation method ([TRW97]) is used.

We focus on the DayMet interpolation approach. In Section 2, we give a formal definition of this approach and show some of its deficits. In Section 3, we suggest some improvements and present some first results in Section 4. Section 5 contains a list of further research that can be done in this field.

2 The DayMet Interpolation Method

We provide an overview of the DayMet approach in Section 2.1, give a short overview of its implementation within the CIMIS project in Section 2.2, and point out some weaknesses of the approach and the implementation in Section 2.3.

2.1 Definition

As input to the DayMet interpolation we have n weather stations W_i $(i = 1, \ldots, n)$ corresponding to two-dimensional observation points $p_i \in \mathbb{R}^2$ on a planar map, elevation $z_i \in \mathbb{R}$ and the associated weather data $f_i \in \mathbb{R}$. Examples for possible weather data are temperature, solar radiation, precipitation, humidity, or wind speed, as measured at W_i.

To interpolate the value at an arbitrary query point Q with two-dimensional coordinates $q \in \mathbb{R}^2$, we define a weight function as a truncated Gaussian filter,

$$w(q,r) = \begin{cases} 0; & r > R(q) \\ \exp\left(-\left(\frac{r}{R(q)}\right)^2 \alpha\right) - e^{-\alpha}; & r \leq R(q) \end{cases}, \tag{2}$$

where r is the radial distance arround q, $R(q)$ is the truncation distance of q, and α is a unitless shape parameter.

We define the weights of the weather station W_i at a query point Q as

$$w_{q,i} = w(q, \|q - p_i\|_2). \tag{3}$$

If the truncation distance were constant, there would be a large number of observation points with non-zero weights in dense regions, whereas in regions with a sparse number of observation points all weights could be zero. Therefore $R(q)$ depends on the local density of weather stations arround q, and an iterative approach is used to find a value for $R(q)$:

1. Start with $R(q) = R$ with R a user-specified value.
2. Use $R(q)$ to calculate the weights $w_{q,i}$ of all W_i $(i = 1, \ldots, n)$ using Equation (2), and calculate the local station density $D(q)$ (number of stations / area) as

   $$D(q) = \frac{\sum\limits_{i=1}^{n} \frac{w_{q,i}}{\overline{w}}}{\pi R(q)^2}, \tag{4}$$

   where $\overline{w}$ is the average weight over the untruncated region of the kernel, defined as

   $$\overline{w} = \frac{\int\limits_{0}^{R(q)} w(q,r)\mathrm{d}r}{\pi R(q)^2} = \left(\frac{1 - e^{-\alpha}}{\alpha}\right) - e^{-\alpha}. \tag{5}$$

3. With a user-specified desired average number of observations N and the calculated value of $D(q)$, we can calculate a new value for $R(q)$ as

   $$R(q) = \sqrt{\frac{\hat{N}}{D(q)\pi}}, \tag{6}$$

   where $\hat{N} = 2N$ is chosen for every iteration except the last one, for which $\hat{N} = N$.
4. Perform I (I being user-specified) iterations of step 2. and 3. to get the final value of $R(q)$.

The value $f(q)$ at the arbitrary query point Q at two-dimensional coordinates $q \in \mathbb{R}^2$ is now estimated as

$$f(q) = \frac{\sum\limits_{i=1}^{n} w_{q,i} f_i}{\sum\limits_{i=1}^{n} w_{q,i}}. \tag{7}$$

For temperature data there exists a relationship between elevation and temperature. In [TRW97] the use of a correction term to take elevation into account is suggested. First, one estimates regression coefficients β_0 and β_1 that describe the correlation between elevation z and temperature t in absence of any other meteorological effect

$$t = \beta_0 + \beta_1 z. \tag{8}$$

To calculate the values for β_0 and β_1, a weighted least squares regression is used on every pair of observation points (W_i, W_j), weighted by the product of the interpolation weights $w(p_i, \|p_i - p_j\|_2)w(p_j, \|p_i - p_j\|_2)$ of one to the other. But instead of calculating the regression directly as in Equation (8), it was suggested to do this regression for the differences of temperature $(t_i - t_j)$ and elevation $(z_i - z_j)$

$$(t_i - t_j) = \beta_0 + \beta_1(z_i - z_j). \tag{9}$$

With temperatures $t_i = f_i$ $(i = 1, \ldots, n)$ and the estimated values of β_0 and β_1, the temperature $t(q) = f(q)$ for a query point Q at two-dimenisional coordinates $q \in \mathbb{R}^2$ with elevation $z \in \mathbb{R}$ is now calculated as

$$t(q) = \frac{\sum\limits_{i=1}^{n} w_{q,i}\left[t_i + \beta_0 + \beta_1(z - z_i)\right]}{\sum\limits_{i=1}^{n} w_{q,i}}. \tag{10}$$

2.2 Implementation of DayMet within the CIMIS Project

Within the CIMIS project, the DayMet interpolation method is implemented as a GRASS module. For temperature interpolation, an elevation map of California is used. The module reads a site file with the values of the weather stations, including exact positions, and interpolates the value for every point on a regular grid of 500×550 points. The grid distance is 2km. The resulting interpolated values are written to the GRASS database as a raster file.

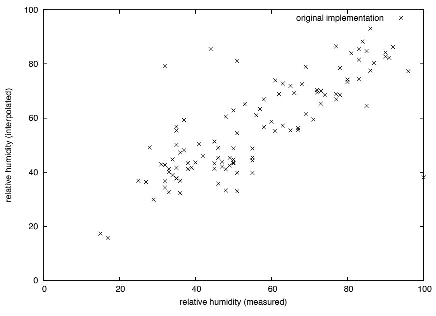
To find suitable values for the free parameters I (number of iterations for calculating $R(q)$), N (desired average number of observation points) and α (shape parameter for weight $w(q, r)$), a range for each of the three parameters is specified and every combination of values is checked via cross validation: For every observation point p_i the interpolation $f(p_i)$ is calculated, using only the $(n - 1)$ other observation points $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$. The cross validation root-mean-square error (RMSE) is

$$E_{\text{RMSE}} = \sqrt{\sum_{i=1}^{n}(f(p_i) - f_i)^2}. \tag{11}$$

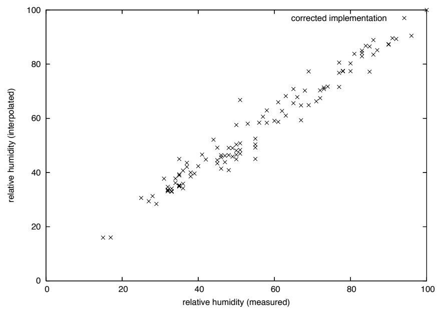
The combination of values for I, N and α that produces the least error E_{RMSE} is used for the interpolation procedure.

2.3 Deficits of the CIMIS DayMet Implementation

In the CIMIS implementation the valid ranges for I and N were interchanged: The ranges were set to $I \in \{3, \ldots, 5\}$ and $N \in \{30, \ldots, 50\}$. With these ranges the maximum number of iterations I to calculate $R(q)$ was five, too few iterations to make the calculations converge. On the other hand, the minimum number of average observations N that are taken into account was 30 and therefore too high. Figure 1(a) shows the correspondence between the measured and the interpolated values at the positions of the observation points. They are nearly unrelated. After interchanging the ranges to $I \in \{30, \ldots, 50\}$ and $N \in \{3, \ldots, 5\}$ the results were better, see Figure 1(b).



(a) Original CIMIS implementation.



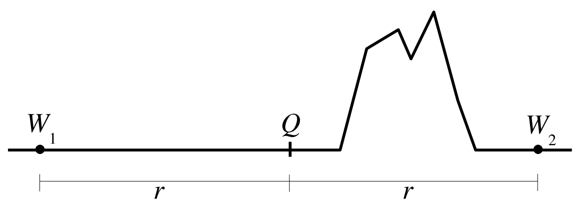
(b) Corrected ranges of I and N.

Figure 1: Measured value (horizontal axis) against interpolated value (vertical axis) at observation points.

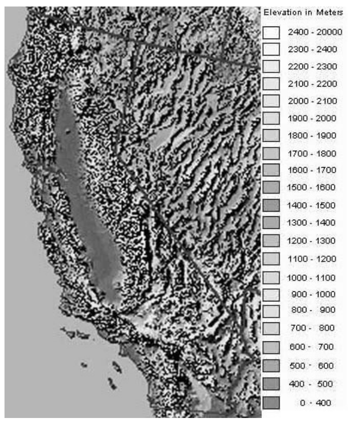
Although being called an "interpolation method" in [TRW97] the DayMet approach is not an interpolation, but only an approximation of scattered data. Calculating the value $f(p_i)$ at an observation point p_i results in a value close to f_i, but in general does not reproduce f_i exactly. This can be seen in Equation (7), since there are in general several non-zero weights w_j, so that $f(p_i)$ not only depends on f_i but also on other observation values. If it were an interpolation method, the points of Figure 1(b) would all lie on the line $y = x$.

There is another shortcoming of the method in [TRW97] related to using the regression Equation (9) for calculating the values of β_0 and β_1. When substracting two instances of the original Equation (8) $t_i = \beta_0 + \beta_1 z_i$ and $t_j = \beta_0 + \beta_1 z_j$, the absolute term β_0 vanishes, resulting in $t_i - t_j = \beta_1(z_i - z_j)$. From this equation only β_1 can be estimated by a least squares regression. With the argument of symmetry one can also conclude that $\beta_0 = 0$, because the indices of the weather stations are artificial. Having two weather stations, either of them can be (z_i, t_i) or (z_j, t_j). Only $\beta_0 = 0$ can then fulfill Equation (9).

Another drawback of the DayMet approach is the way the weights in Equation (2) are calculated: The distance r only takes the x- and y-coordinate of the query position q and the weather station position p_i into account. For temperature interpolation, also the elevation of z and z_i influences the result, see Equation (10). But the topographic structure of the terrain between q and p_i does not play any role. (Think of a terrain with a cross section as in Figure 2(a), built of a plane adjacent to a mountain. To interpolate the value at the query point Q with two-dimensional coordinates q the weights for the weather stations W_1 and W_2 have to be calculated. Since their radial distances $r = \|q - p_1\|_2 = \|q - p_2\|_2$ from Q are the same, they have the same weight $w_1 = w(q, r) = w_2$ from Equation (2). Obviously the influence of W_2 is less than the influence of W_1 since the mountain divides the terrain into two different regions that inhibits air exchange across the mountain. Therefore, the interpolation weight w_1 should be bigger than w_2. Since California has a diverse topographic structure containing high mountains and large flat valleys, see Figure 2(b), these conditions are common.)



(a) A plane adjacent to a mountain.



(b) Map of California.

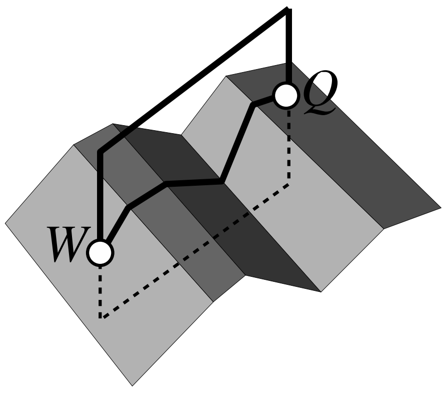
Figure 2: DayMet neglects the topographic structure of the terrain.

In the following, we introduce a way to take the topographic structure of the terrain into account to improve interpolation quality.

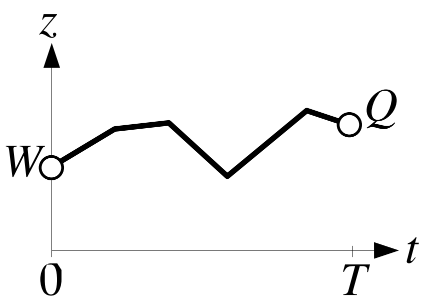
3 Improvement of DayMet

To take the topographic structure of the terrain into account we keep the general concept of the DayMet interpolation, but change the way the distance r in Equation (2) is calculated, so that the terrain elevation influences the weights. If along the path from the weather station W to the query point Q a mountain has to be crossed, the distance should be larger, resulting in a smaller weight and therefore in a smaller influence on the overall result.

We only take the direct path from W to Q into account, i.e., we calculate the intersection of the terrain surface with a plane that contains W and Q and contains the ray from W to the center of the Earth as illustrated in Figure 3(a). This intersection is a planar curve representing the profile of the direct path from W to Q as illustrated in Figure 3(b). This profile is a function $P : [0, S] \to \mathbb{R}$, returning for every (horizontal) position s on the path from W to Q the elevation at that point. The distance r from W to Q is calculated by



(a) Intersection of terrain and plane.



(b) Terrain profile between W and Q.

Figure 3: Direct path from W to Q.

using a function $d : ([0, S] \to \mathbb{R}) \to \mathbb{R}^+$ that uses the profile as input and returns r. We call such a function a *distance function*.

One can think of the original DayMet definition as the distance function d_{xy} with the property that for an arbitrary profile $P : [0, S] \to \mathbb{R}$ we have $d_{xy}(P) = S$. Thus, d_{xy} does not take the profile into account, but just returns the horizontal distance S between W and Q. We now define two different distance functions that do take the profile into account.

A mountain ridge with a height of 1000m has a larger impact on the influence of a weather station than a horizontal distance of 1000m has. Thus, the distance in vertical direction must be amplified to make it comparable to horizontal distances. For this reason, we introduce an exaggeration factor z_{exag}. We define the exaggerated profile $\hat{P}$ as

$$\hat{P}(s) = z_{\text{exag}} P(s). \tag{12}$$

3.1 Arc Length of Convex Hull: d_{ch}

The first distance function returns as the distance of a profile the length of the shortest path through the air from the start to the end point. More formally speaking, this shortest path is the upper convex hull of the profile and the distance is its arc length.

Let $\hat{P}(s) : [0, S] \to \mathbb{R}$ be an exaggerated profile as defined above. Its *upper convex hull* is the function $\hat{P}_{ch}(s) : [0, S] \to \mathbb{R}$ that fulfills the following three conditions:

1. $\forall s \in [0, S] : \hat{P}_{ch}(s) \geq \hat{P}(s)$.
2. $\forall s_1, s_2 \in [0, S], s_1 < s_2 : \hat{P}'_{ch}(s_1) \geq \hat{P}'_{ch}(s_2)$.
3. $\forall \tilde{P} : [0, S] \to \mathbb{R}$ fulfilling condition 1 and 2, $s \in [0, S] : \hat{P}_{ch}(s) \leq \tilde{P}(s)$.

While the first condition ensures that our path is always above ground, the second (monotonic decreasing derivative) ensures that the path does not have unnecessary waves, and the third ensures that the path is as low above ground as possible. The three conditions together ensure that $\hat{P}_{ch}$ is the shortest path through the air from one end to the other.

The distance $d_{ch}(P)$ is now the arc length of that shortest path,

$$d_{ch}(P) = \int_0^T \sqrt{1 + (\hat{P}'_{ch})^2}.$$

Figure 4(a) demonstrates how the distance d_{ch} for a profile is calculated.

The application of these equations results that while crossing a mountain the distance reported is increased by d_{ch}, while crossing a canyon has no effect.

3.2 Radial Distance plus Highest Peak: d_p

The second distance function d_p we define determines the highest peak that has to be crossed and adds this height to the radial distance between start and end point.

We define two slightly different functions d_{p1} and d_{p2}, differing in the base to which the height of the peak is measured. Let $P(s) : [0, S] \to \mathbb{R}$ be a profile, then

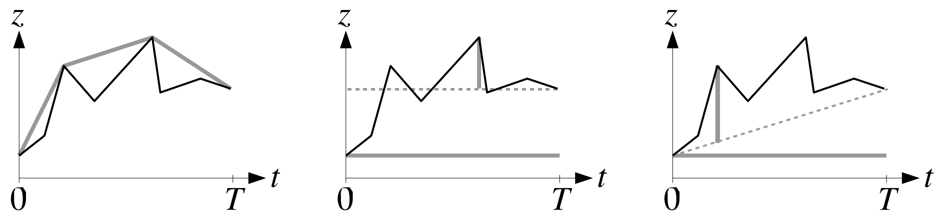
$$\begin{aligned} d_{p1}(P) =& S + \max_{s\in[0,S]} (\hat{P}(s) - \max\{\hat{P}(0), \hat{P}(S)\}) \\ =& S + z_{\text{exag}} \max_{s\in[0,S]} (P(s) - \max\{P(0), P(S)\}), \end{aligned} \tag{13}$$

and

$$\begin{aligned} d_{p2}(p) =& S + \max_{s\in[0,S]} \left(\hat{P}(s) - \left(\frac{s}{S}\hat{P}(0) + \frac{S-s}{S}\hat{P}(S) \right) \right) \\ =& S + z_{\text{exag}} \max_{s\in[0,S]} \left(P(s) - \left(\frac{s}{S}P(0) + \frac{S-s}{S}P(S) \right) \right). \end{aligned} \tag{14}$$

While d_{p1} takes the height of the highest peak relative to the hight of the start or the end point (whichever is higher), d_{p2} takes the hight relative to the linear interpolation between the start and the end point. Figures 4(b) and 4(c) show examples of how the distance with these two distance functions is calculated.

It can be seen that $\forall P : [0, S] \to \mathbb{R} : d_{xy}(P) \leq d_{p1}(P) \leq d_{p2}(P)$ and $d_{xy}(P) \leq d_{ch}(P)$. It depends on the actual profile how $d_{ch}(P)$ compares to $d_{p1}(P)$ and $d_{p2}(P)$.



(a) d_{ch} (b) d_{p1} (c) d_{p2}

Figure 4: Three new distance functions. The black line is the profile, the (sum of the) length of thick gray line(s) is its distance. The dashed line is the base for measuring the height of the peaks for d_{p1} and d_{p2}.

4 Preliminary Results

To compare the results of our distance functions with that used in the original DayMet implementation of CIMIS, we used a data set of relative humidity (values in the range from zero to 100) of the 108 weather stations that measured data that day. We used cross-validation as described in Section 2.2.

Table 1 shows the overall results for E_{RMSE} as defined in Equation (11) for the different distance functions and different values of z_{exag}. These results do not show an advantage of the new distance functions when compared to the original implementation d_{xy}. Only d_{p2} with $z_{\text{exag}} = 50$ shows a slightly improved result, but this might be random.

z_{exag}	d_{xy}	d_{ch}	d_{p1}	d_{p2}
5		12.98	12.95	12.99
50	12.96	13.09	13.32	12.64
500		13.03	13.21	12.88

Table 1: E_{RMSE} values for the different distance functions.

To understand in more detail the errors for specific structures, we searched each distance function for what weather station had the maximal improvement over the original implementation and for what weather station it had the most degradation. These experiments were done with $z_{\text{exag}} = 50$. Figure 5(a) provides an overview of the positions of the three weather stations W_{88}, W_{113}, and W_{35} we studied in detail.

The first weather station we considered was W_{88}. Figure 5(b) shows its neighbourhood, Table 2 lists the relative interpolation weights. While d_{xy} and d_{p1} produced poor results, d_{ch} and d_{p2} had reasonable values. This is the situation we wanted to improve: a mountain ridge divides the terrain into two parts, the dry northern part (W_{54}, W_{146}, W_5, W_{138}, and W_{125}) and the humid southern part (W_{64}, W_{94}, and W_{107}). W_{88} belongs to the northern part, and therefore the southern weather stations should not have any influence on it. Note that d_{p1} produced the same result as d_{xy} since W_{88} is near a mountain top, so any profile ending in W_{88} has W_{88} as highest peak. Therefore, d_{xy} and d_{p1} produce the same distances.

Weather station	W_5	W_{64}	W_{94}	W_{107}	W_{125}	W_{138}	W_{146}	W_{88}
measured rel. hum.	55.0	69.0	86.0	87.0	35.0	45.0	46.0	32.0
d_{xy}	.05	.24	.27	.28			.16	74.1
d_{ch}	.28				.22	.15	.35	45.9
d_{p1}	.05	.25	.27	.27			.16	74.1
d_{p2}	.29	.13			.17	.13	.28	49.5

Table 2: Weights used for interpolation at weather station W_{88} (d_{ch} and d_{p2} have the best results). The last column contains the measured value and the predicted values using the different distance functions at position of W_{88}.

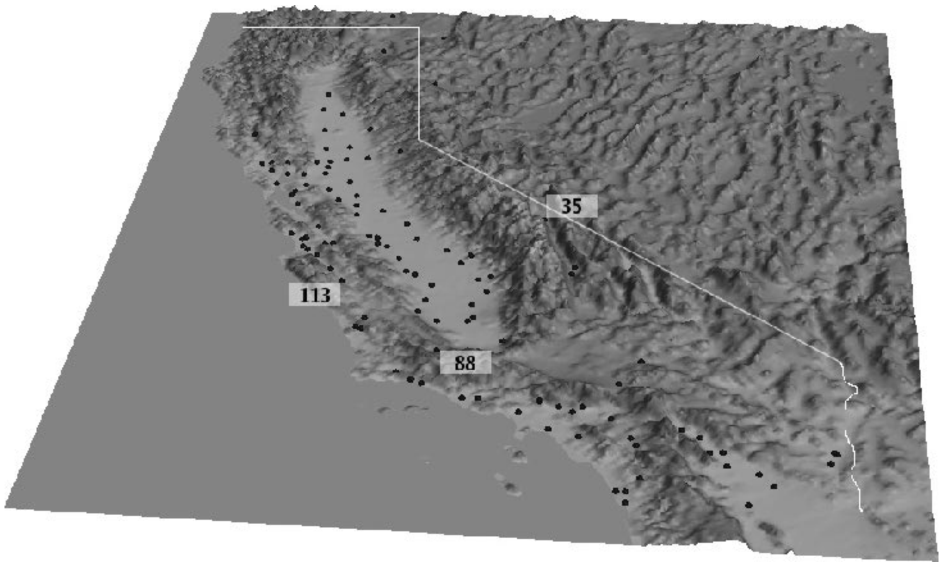
d_{p1} has its best result for W_{113}, see Figure 5(c) for the neighbourhood and Table 3 for the interpolation weights. At first sight it seems we achived the opposite of what we wanted to do: d_{p1} used W_{124} and W_{190} that are hidden behind a mountain. But it also used W_{163} with a higher weight than the other functions, which is the station at the other end of the long, small valley, which resulted in better results.

Weather station	W_{89}	W_{105}	W_{114}	W_{116}	W_{126}	W_{143}	W_{163}	W_{190}	others	W_{113}
measured rel. hum.	83.0	32.0	80.0	91.0	61.0	72.0	51.0	32.0		65.0
d_{xy}	.18		.77				.02	.03		78.7
d_{ch}	.25		.64	.07			.04			80.3
d_{p1}	.16	.05	.45	.05	.05	.05	.06	.07	.06	70.0
d_{p2}	.26		.66	.06			.02			80.8

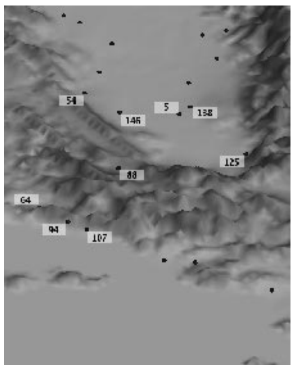
Table 3: Weights used for interpolation at weather station W_{113} (d_{p1} has its best result). The last column contains the measured value and the predicted values using the different distance functions at the position of W_{113}.

W_{35} is now the station, where d_{ch}, d_{p1}, and d_{p2} produced worse results than the original d_{xy}. Table 4 reveals that our new distance functions have heigher weights on W_{183} and W_{189}, exactly following what we wanted: The stations to the west (W_{80}, W_{39}, W_{142}, W_{33}, and W_{86}) are completely out of sight, since they are behind a tall mountain. The results are bad as a consequence of the fact that W_{183} and W_{189}, which share the same valley with W_{35}, show very dry weather (only 15 and 17, respectively), while the weather station W_{35} reports rain with a relative humidity of 100. Possibly, W_{35} suffered from a very local weather phenomenon (like a local thunderstorm), or the reported value was wrong; d_{xy} is the winner by mere chance.

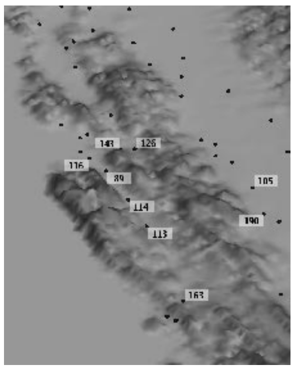
As a first result, we can state that the interpolation quality for selected areas can be increased with the new distance functions. On the other hand, significant improvements do not occur in every case. The reason might be the placement of weather stations: The density is high in valleys and low on mountains. Therefore, d_{xy} prefers stations within the same valley, since the others are too far away, and so the crossvalidation does not show improvement when using the new distance functions. Presumably, the interpolation result in the mountain regions is better using the new distance functions.



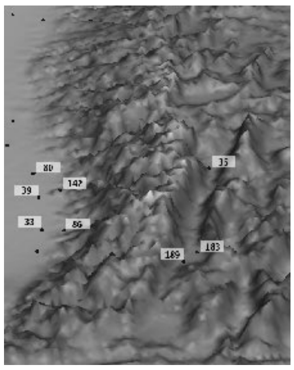
(a) Overview of CIMIS weather stations.



(b) Neighbourhood of weather station W_{88}.



(c) Neighbourhood of weather station W_{113}.



(d) Neighbourhood of weather station W_{35}.

Figure 5: Neighbourhoods of interesting weather stations.

Weather stations	W_{33}	W_{39}	W_{80}	W_{86}	W_{142}	W_{183}	W_{189}	others	W_{35}
measured rel. hum.	47.0	48.0	34.0	37.0	31.0	15.0	17.0		100.0
d_{xy}	.03	.09	.06	.12	.23	.30	.17		26.6
d_{ch}						.54	.46		16.1
d_{p1}					.23	.77			19.7
d_{p2}	.05	.04	.05	.05	.06	.39	.34	.02	23.3

Table 4: Weights used for interpolation at weather station W_{35}. (All new distance functions are bad there.) The last column contains the measured value and the predicted values using the different distance functions at the position of W_{35}.

5 Future Work

We list a few possibilities for future work:

- The tests can be processed on other data sets. The CIMIS database contains many data sets from other dates, and other weather variables different from relative humidity, e.g., temperature, precipitation, wind speed. Such other tests would also reveal whether W_{35} has just the wrong value, or if it is a local weather phaenomenon.
- z_{exag} must be optimized. $z_{\text{exag}} = 50$ seems to be a good initial value.
- Determine under what topographic situations which distance function produces the best results.
- Our results should be compared to other interpolation methods, e.g., Hardy's multiquadric, interpolating splines, or kridging.
- The program structure can be improved to allow more efficient processing. Especially the distances and the truncation distances $R(q)$ for every grid point can be calculated in advance. These have to be recalculated every time a weather station is added or eliminated, or when a weather station has a technical problem transmitting values.
- The CIMIS project interpolates the different weather variables and afterwards calculates ET_0 for every point (interpolate first, then calculate: IC). This is rather time- and space-consuming. A more efficient approach calculates ET_0 at the weather stations and interpolates only this (calculate first, then interpolate: CI), see [MKK05], where these two approaches are compared, finding out that the results are similar. At least there is no significant difference between IC and CI.

6 Acknowledgement

This work was partly funded by the DFG GK 1131 "Visualization of Large and Unstructured Data Sets – Applications in Geospatial Planning, Modeling, and Engineering".

I also want to thank the CSTARS team in Davis for the long and fruitful discussions, especially Carlos Rueda-Velasquez and Quinn Hart.

References

[HBT$^+$06] Quinn Hart, Marcela Brugnach, Bekele Temesgen, Carlos Rueda, Susan Ustin, and Kent Frame. Daily reference evapotranspiration for California using satellite imagery and weather station measurement interpolation. *Submitted to: Agricultural and Forest Meteorology*, 2006.

[HPMM02] Jaroslav Hofierka, Juraj Parajka, Helena Mitasova, and Lubos Mitas. Multivariate Interpolation of Precipitation Using Regularized Spline with Tension. *Transactions in GIS*, 6:135–150, 2002.

[MH93] Helena Mitasova and Jaroslav Hofierka. Interpolation by Regularized Spline with Tension: II. Application to Terrain Modeling and Surface Geometry Analysis. *Mathematical Geology*, 25:657–669, 1993.

[MKK05] M. G. Mardikis, D. P. Kalivas, and V. J. Kollias. Comparison of Interpolation Methods for the Prediction of Reference Evapotranspiration — An Application in Greece. *Water Ressource Management*, 19:251–278, 2005.

[MM93] Helena Mitasova and Lubos Mitas. Interpolation by Regularized Spline with Tension: I. Theory and Implementation. *Mathematical Geology*, 25:641–655, 1993.

[TRW97] Peter E. Thornton, Steven W. Running, and Michael A. White. Generating surfaces of daily meteorological variables over large regions of complex terrain. *Journal of Hydrology*, 190:214–251, 1997.

Procedural 3D Modeling of Cityscapes

Ariane Middel
University of Kaiserslautern
Computer Science Department
D-67653 Kaiserslautern, Germany
middel@informatik.uni-kl.de

Abstract: Modeling large-scale virtual urban environments remains challenging for researchers and urban planners. Cities are difficult to model in detail, since they hold diverse and complex geometries as well as complex social processes. Building large-scale 3D city models by means of photogrammetric reconstruction is a time and resource intensive process and does not provide data for modeling future cityscapes. Recently, a lot of research in the area of comuter graphics has been dedicated to developing alternative 3D modeling techniques.
This survey provides an overview of state-of-the-art procedural modeling techniques for generating virtual cities. Research projects using grammar-based and agent-based models, statistical approaches, and real-time procedural modeling techniques are presented and discussed.

1 Introduction

In *architecture* and *urban planning*, 3D models are increasingly used as standard tools for visualizing urban design and development. Furthermore, virtual cityscapes offer a critical visual dimension for decision making to participants in the planning processes. They communicate future impacts of planning decisions and allow envisioning alternative futures. 3D city modeling is also applied in *environmental planning* where it is crucial for the spatial analysis of 3-dimensional phenomena like air pollution, noise, and earthquakes. *Computer graphics* make use of virtual cityscapes in 3D computer games and visualization applications, just as *entertainment industry* benefits from 3D urban environments in animated movie productions.
The application areas mentioned all have similar demands on 3D model generation: it has to be low-cost, fast, resource-saving, and automatic. In the past decade, research has focused on the 3D building reconstruction from sensor data using photogrammetric methods, active sensors, and hybrid systems [HYN03]. Photogrammetry provides techniques for constructing 3D city models from aerial photographs and terrestrial images. Using active sensors, building geometry is recognized from dense point clouds, e.g. acquired with the airborne laser scanning technology LIDAR (Light Detection and Ranging). Despite intensive research in these areas, both reconstruction methods do not automatically deliver full 3D city models yet [Bre05]. Further, these technologies are designed to develop

models of existing built environments and have little to offer for envisioning future developments. Hybrid systems combining several sensor-based methods like the CC-Modeler developed by CyberCity AG, Zrich [Ulm06] are promising but still semi-automatic approaches.
All things considered, the generation of large-scale 3D urban environments from sensor-based data remains a significantly time-intensive and resource-consuming task. Furthermore, it is inapplicable for modeling urban futures, since it is exclusively based on the satus-quo of real-world data.
For a lot of applications the appearance and the socio-statistical representativeness of an environmental model is more relevant than geometric accuracy. Focusing mainly on land usage, building density or other socio-statistical parameters, the 3D visualization of look-alike cities is sufficient. This can be achieved with a technique known as procedural modeling. Procedural modeling utilizes code segments or algorithms to generate 3D geometries rather than storing an enormous amount of low-level primitives like points, lines or triangles [Ebe96]. Creating abstract and formalized models is an efficient and flexible way to reduce the amount of stored data. Furthermore, the models are controlled by parameters in the generation process to increase the level of detail (LoD). Procedural modeling yields good results for repeating and random processes as well as for self-similar features like fractals. The fractal nature of cities is comprehensively discussed in Michael Batty's book "Cities and Complexity - Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals" [Bat05].

The following sections of this survey address different techniques for procedural city modeling, covering grammar-based models and agent-based techniques. After briefly introducing a statistical approach, a method for real-time procedural modeling will be presented. The survey concludes with a discussion of relative advantages and disadvantages among the different techniques for procedural modeling and gives an outlook on the perspectives of procedural 3D modeling.

2 Grammar-based Models

In the context of procedural city modeling, formal grammars have proven to be a powerful modeling tool. Formal grammars are abstract structures for describing formal languages. Recent approaches use so-called L-Systems for generating a variety of geometric elements in 3D city models [CdSF05]. An L-System or Lindenmayer-System is a parallel string rewriting mechanism that alters a string iteratively according to specified production rules. The resulting string can be interpreted geometrically to produce graphical output. L-Systems were conceived by Aristid Lindenmayer [Lin68] to describe the development of multicellular organisms. In the 90's, they have become a sophisticated computer graphics tool for simulating and visualizing plant geometry [PL90].
Kato et al [KOO$^+$98] are the first to reveal a substantial similarity between the growth of branching structures and the development of street networks. They introduce a virtual city modeling technique using stochastic parametric L-Systems to generate varying road networks. Their technique supports hierarchical street systems and can produce both linear flow systems and cellular networks (see figure 1(a)).

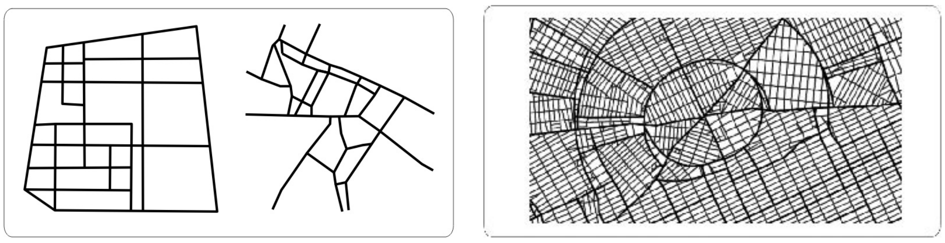

(a) Map and Tree L-Systems [KOO+98]   (b) Self-sensitive L-Systems [PM01]

Figure 1: Street networks generated with L-Systems

The city modeling system CITYENGINE by Parish and Müller [PM01] incorporates an advanced street generation algorithm based on extended L-Systems. Unlike previous Lindenmayer-Systems, the enhanced grammar allows for the creation of closed loops and intersecting road branches. This is accomplished by adding self-sensitiveness to the nature of L-Systems. CITYENGINE employs a hierarchical set of production rules and enables the generation of streets that follow superimposed patterns. To derive a large-scale road map (compare figure 1(b)), we would require information about geographical image maps on elevation, vegetation, and land-water boundaries as well as geostatistical maps on population density, zones, and land-use. Since the introduction of CITYENGINE, L-Systems have been widely used for both reproducing existing street networks [GMB06] and creating fictional road maps [HMFN04]. Yet for modeling geometrically detailed buildings, L-Systems are difficult to adapt since they emulate growth-like processes in open spaces. CITYENGINE implements an L-System to generate simple buildings consisting of translated and rotated boxes. In this way, large urban environments emerge, but with a low resulting level of detail.
INSTANT ARCHITECTURE is a technique developed by Wonka et al. [WWSR03] for automatic modeling of geometrically detailed buildings. Their approach uses parametric split grammars, derived from the concept of shape grammars [Sti80] which have been successfully applied in architecture to construct and analyze architectural designs. Split grammars operate with production rules consisting of geometric split operations. The idea is to generate geometrically rich 3D building layouts by hierarchically subdividing build-

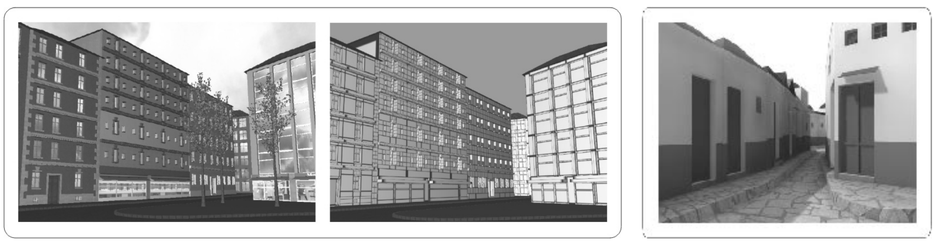

(a) Instant Architecture [WWSR03]   (b) Roman Housing [MVUG05]

Figure 2: Architectural models generated with shape grammars

ing facades into simple attributed shapes (figure 2(a)). Wonka et al. set up a large grammar rule database to model various buildings in different architectural styles. INSTANT ARCHITECTURE generates highly detailed buildings, but is only applicable for small-scale urban areas.
Inspired by INSTANT ARCHITECTURE is the virtual 3D model of Roman housing architecture presented in [MVUG05]. Müller et al. reproduce ancient sites by extending the functional range of the CITYENGINE system to shape grammars like introduced in [WWSR03]. The production rules for shape grammars are deduced from archaeological and historical data to ensure a faithful reproduction of Roman architecture. Plausibility is further enhanced by importing real building footprints and streets as ground truth.
Also integrated in the CITYENGINE framework is a sophisticated technique for procedural modeling of computer graphics architecture proposed by Müller et al. [MWH^{+}06]. Their approach uses extended set grammars, so-called CGA SHAPES, combining the benefits of [WWSR03] and [PM01]. CGA SHAPES are suited for creating large-scale and at the same time geometrically detailed 3D cityscapes. Intrinsic context sensitive shape rules are applied sequentially to building footprints, the axioms of the productions, in order to generate mass models of the buildings. A mass model is a union of simple volumes and can consist of highly complex polygonal faces. In the next step, 2D building facades are extracted from the 3D shapes and structured into their elements. Here, CGA SHAPES re-use the volumetric information to solve intersection conflicts between adjacent facades. After adding details for ornaments, doors and windows the buildings are finally roofed with different types of roof-tops. In this manner, CGA SHAPES are applicable to model diverse urban areas like office districts, suburban environments and ancient cities (see figure 3).

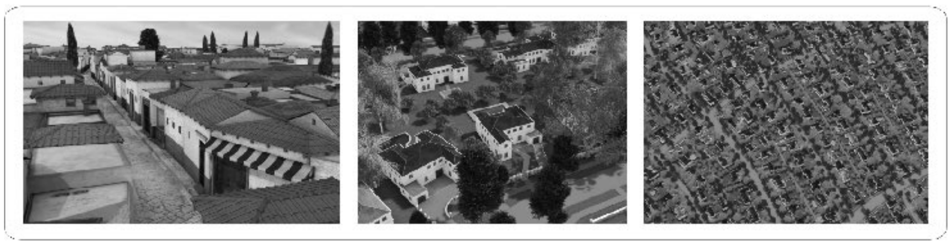


Figure 3: Pompeii and Beverly Hills, modeled with CGA SHAPES [MWH^{+}06]

3 Agent-based Models

Agent-based models are computational models for simulating real world phenomena and are closely related to cellular automata and multi agent systems. The main modules of agent-based models are rule based agents, situated in space and time. They reside in artificial environments, e.g. virtual cities, are free to explore their surroundings, and dynamically interact with their environment and other agents. Simple transition rules drive the agents' behavior and result in purposeful, intelligent, far more complex reactions. For a detailed introduction to computational agent-based modeling, particularly regarding multi-agent systems, and a broad introduction to automata-based urban modeling the reader is referred to [BT04].

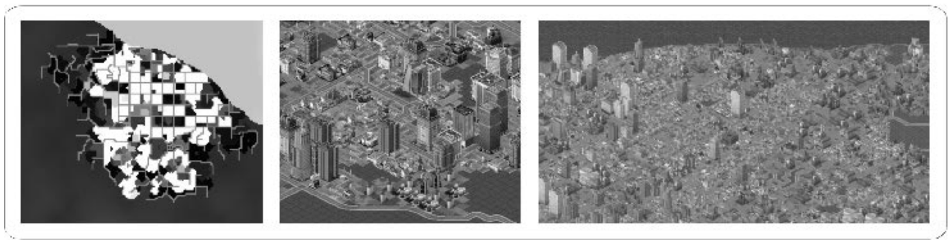


Figure 4: Agent-based modeling of virtual cities [LWR+04]

Until recently, agent-based modeling has been largely restricted to 2D. For Instance, URBANSIM [Wad02] is a sophisticated 2D agent-based simulation model for large-scale urban environments that incorporates the interactions between land use, transportation, and public policy. Lately, systems have been designed to combine agent-based urban geosimulation with 3D visualization of urban environments. Lechner et al. [LWR+04] present an approach to procedurally generate virtual cities using an agent-based simulation. Their system only depends on a terrain description as low-level input and accepts optional parameters like water level and road density. Three types of agents are responsible for generating the hierarchical road network by exploring the virtual space. Primary road agents connect highly populated regions, extenders expand the existing street network to urban areas not serviced by a road, and connectors interlink poorly accessible areas with tertiary streets. To output a land usage map (compare figure 4) developer agents generate parcels and land use for residential, commercial, and industrial zones. Finally, the map is visualized using the SimCity 3000 graphics engine, as shown in figure 4.

4 Statistical Models

As opposed to grammar-based and agent-based approaches, statistical models utilize statistical propagation techniques to procedurally generate urban environments. Statistical models are often employed to complement other procedural modeling techniques and are rarely used stand-alone. An example for the automatic generation of large geometric models based on statistical parameters is “A Different Manhattan Project” [YBH+02]. Yap et al. reconstruct the city of Manhattan based on the TIGER data set and diverse physical parameters like average size and height of buildings, zoning classification of land use, and dominant architectural styles. The parameters are statistically propagated over the city, using different parameter scripts for varying districts in order to capture the uniqueness of each neighbourhood. Landmarks such as the Empire State Building are hand-coded into the geometric model to accomplish a realistic view of the city skyline (see figure 5).

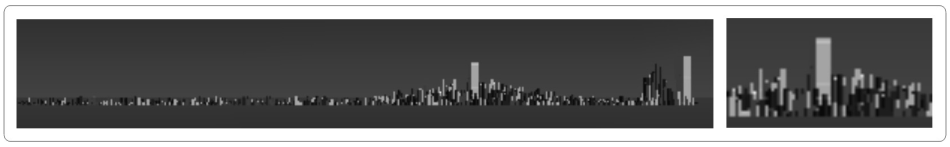


Figure 5: Statistical model of Manhattan [YBH+02]

5 Real-time Procedural Modeling

Besides believability and complexity of the scene, an important characteristic for the rendering of procedural models in computer graphics is real-time interactivity. While the modeling approach by DiLorenzo et al. [DZT04] is focused on the interactive animation of evolving cities, the technique introduced in [GPSL03] concentrates on dynamic geometry rendering in real-time. Greuter et al. have developed a method for generating pseudo-infinite virtual cities on-the-fly. Randomly generated regular polygons are merged into floor plans, extruded to parameterized buildings and textured, resulting in highly detailed office districts. The model's building geometry is generated dynamically as needed inside the user's cone of sight while the user is interactively exploring the city. This view frustum filling (see figure 6) provides for the generation of pseudo-infinite cityscapes in real-time that would take a lifetime to explore.

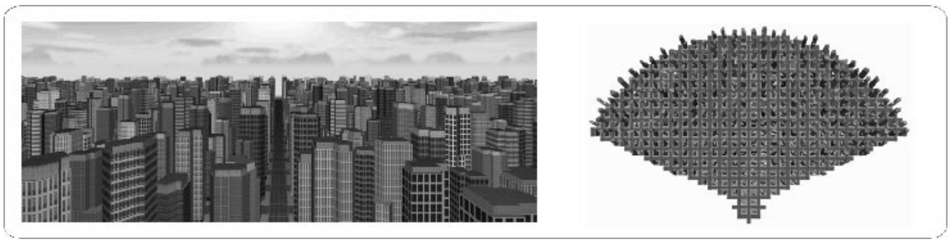


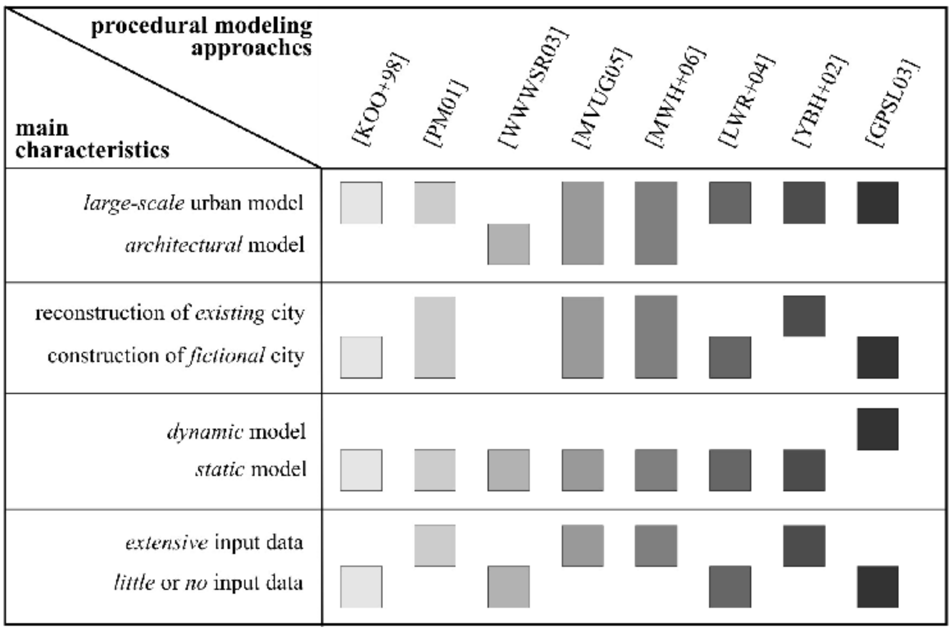
Figure 6: Real-time generation of 'pseudo-infinite' city [GPSL03]

6 Discussion

Previous sections reviewed state-of-the-art techniques in procedural modeling, including grammar-based, agent-based, statistical, and dynamic models. The presented approaches mainly differ in the following characteristics (see figure 7 for comparison):

- scalability/LoD (large-scale models vs. architectural models)
- realness (existing cities vs. fictional cities)
- dynamics (static vs. dynamic cities)
- amount of input data (extensive vs. little input data)

These characteristics are conducive to the believability of the procedural model, whereas authenticity has a slightly different focus in different application areas. Besides visual fidelity, the realistic projection of socio-statistical and economic input parameters is a decisive factor for believability in planning applications. In contrast, gaming industry attaches importance to interactive, dynamic procedural modeling in real-time.



main characteristics \ procedural modeling approaches	[KOO+98]	[PM01]	[WWWSR03]	[MVUG05]	[MWH+06]	[LWR+04]	[YBH+02]	[GPSL03]
large-scale urban model	■	■		■	■	■	■	■
architectural model			■	■	■			
reconstruction of *existing* city		■		■	■		■	
construction of *fictional* city	■	■		■	■	■		■
dynamic model								■
static model	■	■	■	■	■	■	■	
extensive input data		■		■	■		■	
little or *no* input data	■		■			■		■

Figure 7: Predominant characteristics of procedural modeling techniques

Greuter et al. [GPSL03] mainly contribute to the applicability of procedural city modeling in computer graphics and visualization applications. They dynamically generate visually interesting and complex buildings in real-time, still their interactive model is inapplicable for planning purposes. It only supports one building type, office skyscrapers, and the transportation network does not correspond to a realistic city, since streets are uniformly gridded. The procedural model of Yap et al. [YBH+02] overcomes this drawback by including a transportation system validation process using the TIGER data set. On that account, the implemented method only works for existing cities, not for future scenarios or fictitious cities. Apart from this, the "Different Manhattan Project" does not take into account the economic conditions of the modeled cityscape as well as the dynamics of residents' activities. In considering these aspects, the agent-based approach by Lechner et al. [LWR+04] simulates the development of different land use zones. Their model is suitable for planning applications where no socio-statistical and economical data is accessible. The believability of the agent-based simulation benefits from a more realistic urban layout, whereas it lacks an authentic road network with street patterns and visual attractiveness.

While Lechner et al. focus on land usage and building distribution, most grammar-based approaches aim at colonizing road networks. Kato et al. [KOO$^+$98] generate road maps holding two different street patterns, whereas Parish et al. [PM01] enhance L-systems and implement production rules which are easily extendible to several different street patterns. Their method is applicable for both reproducing existing cities and creating fictitious or future cities. Nevertheless, the L-system production rules presented do not allow a proper representation of the buildings' functionality. This considerable shortcoming is somewhat resolved by INSTANT ARCHITECTURE [WWSR03], offering a variety of different architectural styles and designs for individual buildings. The complex geometric representation is at the expense of scalability, the approach only focuses on architecture and disregards urban layout and streets. Procedural modeling techniques combining large-scale models with geometrically detailed architecture are introduced by Müller et al. In [MVUG05] grammar-based techniques are combined to recreate ancient Roman cities, in [MWH$^+$06] CGA SHAPES generate extensive urban models with up to a billion polygons. Both approaches offer high scalability as well as strong visual fidelity and are suited for the generation of ancient, existing, and future cityscapes. Yet, to visualize potential future impacts of planning decisions in urban planning applications, additional input data is needed. The models have to incorporate correctly projected demographic and economic data. Our current techniques and knowledge in the area of future 3D models of real cities is still at a very nascent stage.

7 Conclusion

Procedural content generation is a promising technique in many application areas where the visual and socio-statistical believability of a model is more important than its geometric authenticity. Procedural models are useful for visualization and computer graphics applications, entertainment industry, and especially architectural, urban, and environmental planning. With increasing hardware performance, the application of procedural modeling techniques will become more ubiquitous and more refined. Future research, especially in the context of urban planning, needs to find ways of integrating different procedural modeling techniques. A combination of agent-based simulation tools like UrbanSim with grammar-based approaches like CGA SHAPES will result in a comprehensive procedural modeling tool for the simulation and visualization of existing and developing cities.

8 Acknowledgements

This work was supported by the German Science Foundation (DFG, grant no. 1131) as part of the International Research Training Group (IRTG) in Kaiserslautern on "Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling, and Engineering".

References

[Bat05] Michael Batty. *Cities and Complexity*. MIT Press, September 2005.

[Bre05] Claus Brenner. Building Reconstruction from Images and Laser Scanning. *International Journal of Applied Earth Observation and Geoinformation*, 6(3-4):187–198, March 2005.

[BT04] Itzhak Benenson and Paul M. Torrens. *Geosimulation: Automata-Based Modeling of Urban Phenomena*. John Wiley & Sons, July 2004.

[CdSF05] António Fernando Coelho, António Augusto de Sousa, and Fernando Nunes Ferreira. Modelling Urban Scenes for LBMS. In *Proceedings of the tenth international conference on 3D Web technology*, pages 37 – 46, 2005.

[DZT04] Paul DiLorenzo, Victor B. Zordan, and Duong Tran. Interactive Animation of Cities Over Time. In *17th International Conference on Computer Animation and Social Agents*, Geneva, Switzerland, 2004.

[Ebe96] David S. Ebert. Advanced Modeling Techniques for Computer Graphics,. *ACM Computing Surveys*, 28(1):153–156, March 1996.

[GMB06] Kevin R. Glass, Chantelle Morkel, and Chaun D. Bangay. Duplicating Road Patterns in South African Informal Settlements Using Procedural Techniques. In Stephen N. Spencer, editor, *Proceedings of the 4th International Conference on Virtual Reality, Computer Graphics, Visualisation and Interaction in Africa, Afrigraph 2006*, Cape Town, South Africa, January 2006. ACM.

[GPSL03] Stefan Greuter, Jeremy Parker, Nigel Stewart, and Geoff Leach. Real-time Procedural Generation of 'Pseudo Infinite' Cities. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Austalasia and South East Asia*, Melbourne, Australia, February 2003. ACM Press.

[HMFN04] Masanobu Honda, Kaznori Mizuno, Yukio Fukui, and Seiichi Nishihara. Generating Autonomous Time-Varying Viirtual Cities. In *Third International Conference on Cyberworlds (CW'04)*, pages 45–52, 2004.

[HYN03] Jinhui Hu, Suya You, and Ulrich Neumann. Approaches to Large-Scale Urban Modeling. *Computer Graphics and Applications, IEEE*, 23(6):62–69, November 2003.

[KOO$^+$98] Nobuko Kato, Tomoe Okuno, Aya Okano, Hitoshi Kanoh, and Seiichi Nishihara. An Alife Approach to Modeling Virtual Cities. *IEEE International Conference on Systems, Man, and Cybernetics*, 2:1168–1173, October 1998.

[Lin68] Aristid Lindenmayer. Mathematical Models for Cellular Interaction in Development, parts I and II. *Journal of Theoretical Biology*, 18:280–315, March 1968.

[LWR$^+$04] Thomas Lechner, Ben Watson, Pin Ren, Uri Wilensky, Seth Tisue, and Martin Felsen. Procedural Modeling of Land Use in Cities. Technical Report NWU-CS-04-38, Northwestern University, August 2004.

[MVUG05] Pascal Müller, Tijl Vereenooghe, Andreas Ulmer, and Luc Van Gool. Automatic Reconstruction of Roman Housing Architecture. In Baltsavias et al., editor, *Recording, Modeling and Visualization of Cultural Heritage*, pages 287–297. Balkema Publishers (Taylor & Francis group), 2005.

[MWH^{+}06] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural Modelingof Buildings. *ACM Transactions on Graphics (TOG), SIGGRAPH 2006*, 25(3):614–623, July 2006.

[PL90] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *Algorithmic Beauty of Plants*. Springer-Verlag New York, Inc., 1990.

[PM01] Yoav I H Parish and Pascal Müller. Procedural Modeling of Cities. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308, New York, NY, USA, 2001. ACM Press.

[Sti80] George Stiny. Introduction to Shape and Shape Grammars. *Environment and Planning B: Planning and Design*, 7:343–351, 1980.

[Ulm06] Kilian Ulm. The Virtual City. *Position Magazine*, pages 57–58, June/July 2006.

[Wad02] Paul Waddell. UrbanSim: Modeling Urban Development for Land Use, Transportation and Environmental Planning. *Journal of the American Planning Association*, 68(3):297–314, 2002.

[WWSR03] Peter Wonka, Michael Wimmer, Francois Sillion, and William Ribarsky. Instant Architecture. *Transactions on Graphics, SIGGRAPH 2003*, 22(3):669 – 677, July 2003.

[YBH^{+}02] Chee Yap, Henning Biermann, Aaron Hertzman, Chen Li, Jon Meyer, Hsing-Kuo Pao, and Salvatore Paxia. A Different Manhattan Project: Automatic Statistical Model Generation. In *IS&T SPIE Symposium on Electronic Imaging*, San Jose, California, January 2002.

RAVis: Room Acoustics Visualization Using Phonon Tracing

Frank Michel and Eduard Deines
University of Kaiserslautern
Computer Science Department
IRTG
D-67653 Kaiserslautern, Germany
{michel, e_deines}@informatik.uni-kl.de

Abstract: We give an overview of our system for the simulation and visualization of room acoustics and sound propagation within rooms. The simulation uses phonon tracing, which was inspired by methods like photon mapping. Phonon tracing is used to simulate mid- and high-frequency sound. The result of the simulation, the phonon map is used as a starting point for the visualization of sound propagation and exploring the influence of different room surfaces on the acoustical behaviour of the room (i.e. absorption of different frequencies).

1 Introduction

Room acoustics is important for designing concert halls, theaters or class rooms. Nowadays, computer-based simulations of acoustic behavior inside a room are available [Kro68, Kul84, Vor89, FCE$^+$98, KJM04, BDM$^+$05]. The question comes up for an appropriate visual representation of the simulation results.
Stettner et. al. [SG89] visualize room acoustic quality properties such as clarity or spatial impression by use of specific icons. Furthermore, they color the room boundaries according to the pressure level. Khoury et. al. [KFW98] represent the sound pressure levels inside the room by means of color maps. Additionally, the authors analyze the precedence effect (or "law of the first wavefront") by using iso surfaces. Funkhouser et. al. [FCE$^+$98] use visualization of source points, receiver points, pyramidal beams, reverberation path etc. in order to understand and evaluate their acoustic modeling method. The system also provides the presentation of power, clarity etc. A couple of commercial systems [ODE, CAT, Bos] provide some tools for visualizing computed acoustic metrics. In [BDM$^+$05] we have visualized the spacial propagation of particles called phonons from the sound source using spheres colored according to their energy spectra.
Our work concentrates on the presentation of the simulation results of our phonon tracing algorithm [BDM$^+$05]. Here, we introduce various approaches for visualizing the phonon map. First, we represent the individual phonons on their positions at the room surfaces colored according to their energy spectra. This method provides insight how the room

influences the spread sound waves. In order to observe the propagation of different wave fronts for the first few reflections, we visualize them as deformed surfaces. For the visualization of late reflections and a time dependent look at the phonon map, scattered data interpolation is used. For the representation of acoustic behavior at a certain listener position, we use colored spheres deformed according to the received sound.
The remainder of our paper is arranged as follows. In the next section we will give a short description of the phonon tracing algorithm. In section 3 we will present our visualization approaches and provide examples. Then we will discuss our results in the last section.

2 Phonon Tracing

Photon mapping [Jen96] 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 [BDM+05, DBM+06], which is summarized in the following.

2.1 Problem Specification

Our simulation algorithm requires the following input information:

- position of sound source s
- emission distribution E of sound source
- one or 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 threshhold ε for terminating the phonon paths

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 n_p, and the material m_p at the current reflection.
Our simulation algorithm contains two steps, 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.

2.2 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 normalized 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 functions 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. $max\{e_{p,i}\}_{i=1}^{n_e} > \varepsilon$, the next phonon re-uses the path and energy of the preceding one, saving computation time. It is started at the current position with respect to the outgoing direction d_p and contributes to the phonon map at the next surface intersection. If the threshold is not exceeded and a minimum number of reflections have been computed, then a new phonon is started 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 written to a file for further visualization purposes.

2.3 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 impulse (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 raytracing [Kro68, Kul84], 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 then 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 constructed using the Fourier Transform. The filter design is described in [BDM$^+$05].

3 Visualization Methods

The phonon map as result of phonon tracing (section 2) characterizes the acoustic behaviour of a scene considering the location of a specific sound source. It consists of the reverberations of a unit pulse, coming from different directions with different time delays and specific energy distributions. How can we visualize this complex information?

Color Coding

We provide the option to consider all frequency bands in total or each of them separately, whereas we consider 10 frequency bands associated with the frequencies $40, 80, ..., 20480$ Hz.
In the first case the spheres representing the phonons are color coded by using the RGB components, such that red corresponds to the average of $e_{p,8}, ..., e_{p,10}$ (5120, 10240, 20480 Hz), green corresponds to the average of $e_{p,5}, ...e_{p,7}$ (640, 1280, 2560 Hz), and blue to the average of $e_{p,1}, ..., e_{p,4}$ (40, 80, 160, 320 Hz).
In the second case, considering only one frequency band, we color coded the energy of this frequency band using the HSV model. We interpolate the color between red (full energy) and blue (energy equals zero) corresponding to the energy $e_{p,i}$ of the i-th frequency band.

3.1 Visualizing Phonons

The first visualization focuses on the spatial propagation of a pulse response from the sound source (figure 1). The corresponding wave front traverses the room and is reflected at surfaces, altering its intensity and energy spectrum. We visualize this wave front by rendering a small sphere for every phonon path with color coded spectral energy.
When sliding through time, the spheres follow the phonon paths. At small time values, the reflected wave fronts are clearly visible. When the number of reflections increases, however, it becomes more difficult to recognize individual fronts.
In order to tackle this problem, we integrated the following function into our interactive visualization system [DMB$^+$06]:

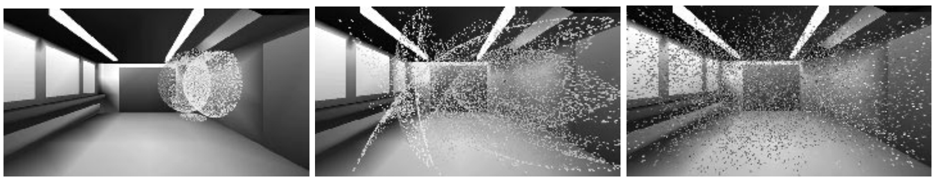


Figure 1: Wave-front propagation from a spherical sound source. The visualization shows phonons with color-coded spectral energy due to reflection at different materials.

- varying the percentage of phonons to be rendered
- rendering only the phonons reflected from a selected material
- exchanging selected materials
- varying time / traversed distance.

3.2 Visualizing Phonons on Surfaces

The second method we have implemented to examine the phonon map is the visualization of certain phonons at their position inside the given scene. Each phonon is rendered as a sphere and is colored according to its spectral energy. In order to show the phonons outgoing direction v_p we render a cone whose peak is rotated towards v_p. The color of the cone corresponds to the phonons energy, too.
Since the number of phonons in the phonon map is large we render only phonons with a given number of reflections n_p simultaneously. With this approach we examine how the surfaces of the considered scene affect the overall acoustic of the room. The phonon map contains for each phonon the number of reflections n_p, so we display only those phonons with a certain number of reflections, visualizing their frequency spectrum and outgoing direction.

Figure 2 shows an example of this visualization approach. The phonon map consists of one million phonons. In figures 2 (a) and (d) the overall frequency spectrum of the phonons after one and four reflections, respectively, is depicted using the RGB components. As we can see in figure 2 (a) the walls, the bottom and the canvas absorb high frequencies better than low frequencies(bluish color), whereas the door, for example, reflects all frequencies equally. After four reflections at the scene surfaces we can observe a shift towards lower frequencies. The representation using the RGB model shows an average of the energy spectrum at low, middle and high frequencies. Sliding through the frequency bands we can observe the absorption for each individual frequency band. Figure 2 (b, e) shows the energy at 160 Hz and figure 2 (c, f) at 10240 Hz after one and four reflections, respectively. As we can see, after four reflections there are about 75% of the energy of the phonons at 160 Hz, whereas the energy at 10240 Hz is nearly completely absorbed by the room. By depicting the outgoing direction we can guess which object the phonon will hit next.

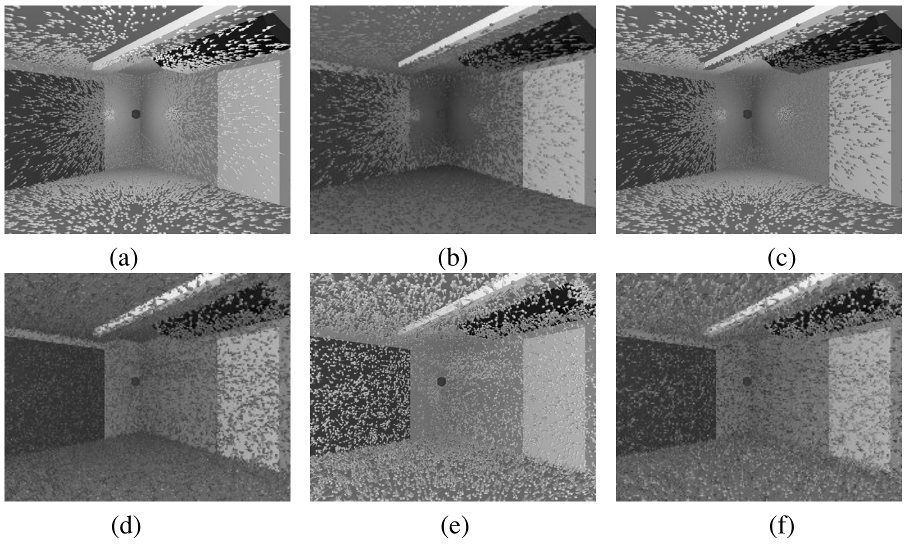


Figure 2: Visualization of particular phonons. First (a) and fourth (d) reflection color coded using the overall frequency spectrum. First (b, c) and fourth (e, f) reflection color coded using the frequency band at 160 Hz and at 10240 Hz, respectively.

3.3 Wave Front Visualization

In this section we describe an approach visualizing wave fronts reflected at the room surfaces by use of triangulated surfaces. In order to build these surfaces we need to know which phonons belong to a common wave front. Therefore we subdivide the phonons in clusters of equal history, such that phonons in the same cluster satisfy the following criteria:

- equal numbers of reflections n_p
- and for each reflection:
  - equal material indices m_p (same object of the scene)
  - equal surface normals n at the reflection position

Consequently all phonons inside a cluster have equal energy spectra.
In order to build the cluster we need to trace the phonons back to the sound source and compare their histories. This is not a difficult task since the phonon p_i re-uses the path and energy of the preceding one (see section 2).
We can construct the surface of the wave front comming from the sound source as a convex hull of phonons on the unit sphere and obtain the neighborhood relationship of particular phonons. This relation does not change in time for a set of phonons in the same cluster, so the polygonal representation of the wave front must be calculated only once. For the construction of the convex hull on the unit sphere providing a Delaunay-triangulation we use the CGAL library [CGA]. The wave front surfaces reflected at the objects inside the

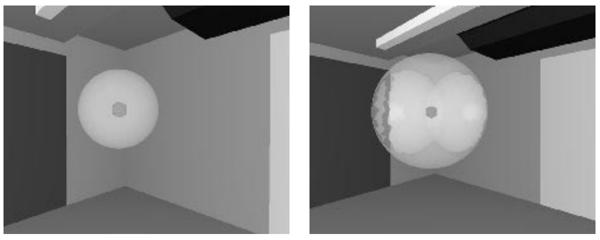

(a) (b)

Figure 3: Wave front traversed from the sound source (a), separated after reflections (b).

considered room are built by keeping only triangles of the initial wave front whose vertices (phonons) reside in the according cluster. Figure 3 shows the wave front coming from the source. Where it hits the canvas, for example (figure 3 (b)), the faces that now belong to the wave front of the reflection at the canvas are separated from the initial surface. Now we can render the resulting surfaces for visualization of the wave fronts reflected from the scene objects. To illustrate where the wave front hits the object first and in which direction it propagates we use the phonons traversed distances d_p to deform the surface. First, we determine the maximum traversed distance d_{max} of all phonons inside the cluster. Then, we render the phonons (which are the triangle vertices of the wave front surface) in an offset from the according scene surface. We color coded the wave front surfaces according to the energy spectra of corresponding phonons. Depending on whether we want to examine all frequency bands in common or each of them separately we use the RGB model or the HSV model as described in the previous section. Furthermore, we set the surface transparency according to the average energy of the wave front.
For better clarifying the propagation direction of the wave fronts and the traversed distance, we provide the option to draw colored cones at the position of the phonons. The peaks of these cones are rotated towards the phonons outgoing directions. Their color corresponds to the traversed distance and is calculated by use of the HSV model. Since displaying all wave front surfaces becomes complex, our implementation provides the alternative to select wave fronts by number of reflections, material(object) or history.

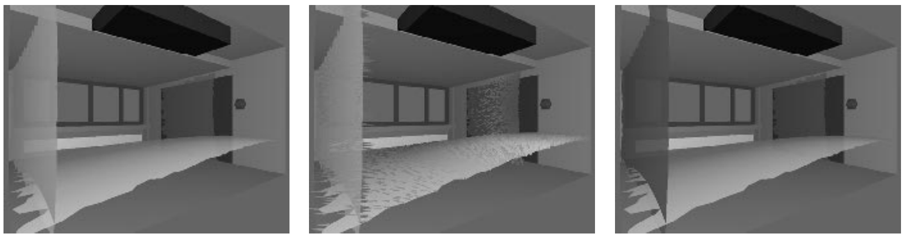

(a) (b) (c)

Figure 4: Clustered wave front. (a) first reflection color coded using the overall frequency spectrum. (b) Traversed distance represented by use of cones. (c) first reflection color coded using the frequency band at 10240 Hz.

Figure 4 shows an example of the described visualization approach. The red colored sphere in the corner of the room represents the sound source. The depicted wavefronts are the first reflections comming from the bottom, the wall and the canvas. In image 4 (a) and (b) we see the wave fronts color coded using the RGB model and overall frequency spectrum as described above. We can see that the bottom and the canvas predominantly absorb more high frequency, whereas the wall absorbs more low frequencies. Figure 4 (c) shows the wave front surfaces color coded by use of HSV model and the frequency band at 10240 Hz. This frequency band is absorbed by the canvas and bottom, but is reflected almost completely by the wall.
The surfaces are deformed according to the traversed distance of the phonons belonging to the clusters. Here, we can observe where the wavefronts hit the room surfaces first and in which direction they will further spread. A representation of the traversed distance by use of colored cones (figure 4 (b)) shows that the wave front propagated from the sound source hit the canvas and the bottom before the wall.
Since the wave front traversed from the sound source is splitted after each reflection, the visualization approach described in this section is applicable only to the first few reflections.

3.4 Scattered Data Interpolation

At higher reflection orders, the clusters become smaller and smaller, until they contain only a single phonon. In the following we develop a visualization method for the entire phonon map based on scattered-data interpolation.
The goal of the interpolation is to get a continuous representation of the emitted energy on the surfaces of the scene. First of all it is used to visualize phonons not belonging to the early reflections of wavefronts, because these cannot be visualized as cluster surfaces due to the increasing fragmentation. Rather than visualizing individual reflections, we look at the phonon map from a different viewpoint, namely discrete timesteps, i.e. ”show the energy emitted from the floor at 50 msec”. This allows us to look at the change in energy comming from a surface over time.
The interpolation is done for the energy and pathlength of the phonons. The direction is neglected, since it is not important for the visualization due to scattering.
Depending on what to visualize, different kinds of phonons are used for the interpolation, i.e. all phonons which are reflected at least 5 times or all phonons present on a surface for a certain timestep.
The interpolation results are color coded corresponding to section 3.2. This means using RGB color when visualizing the whole frequency spectrum and HSV for distinct frequency bands.
As mentioned before, the interpolation is used to visualize late reflections ($n_p \geq 4$). Therefore the energy of all phonons reflected at least n times is used to calculate the energy for the whole surface.

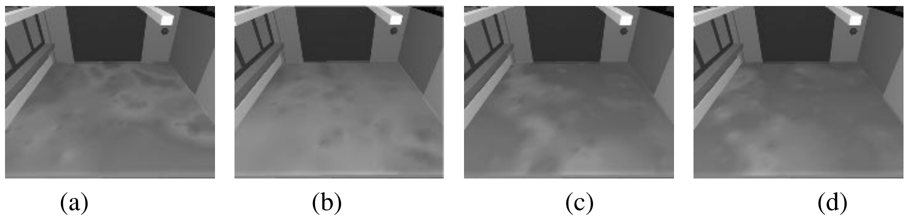


Figure 5: Energy on the floor at timesteps 20-23 msec for the 160Hz band.

An example for distinct time steps is given in figure 5 (a-d), where the change in emitted energy for the 160Hz band is shown at 20(a), 21(b), 22(c) and 23(d) msec.
Having dealt with the visualization of the phonon map on the surface in the last sections, the next section will provide information about the received energy at distinct listener positions throughout the room.

3.5 Listener-based Visualization

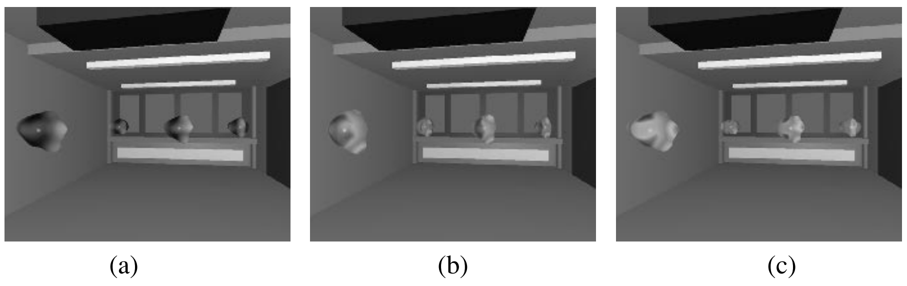


Figure 6: Deformed spheres representation at four listener positions. (a) color coded using the overall frequency spectrum (b) by 80 Hz and (c) by 1280 Hz.

The approaches described above visualize the phonon map considering the surfaces of the room and their acoustic properties. In this section we present a visualization method depicting the received energy at a listener position. With this approach we can detect from which direction the most energy reaches the listener and visualize the energy spectrum.
For this purpose we render a triangulated sphere deformed according to the weighted phonons received at the listener position. The phonons are collected using the collection step described in section 2. The color of the sphere points is calculated by use of the RGB (for overall frequency spectrum) or the HSV (certain frequency band) model as mentioned in the previous sections. For assessing purpose of the acoustic quality at a listener position it is important to know at which time the reflections arrive at the listener. For this reason we add the feasibility to determine the lower and upper time limits for phonon selection.
The following figures present the results of the introduced visualization approach. Figure 6 (a) show spheres at four positions in the considered room. The energy is collected over

the entire time interval. The spheres are deformed and color mapped by using the overall energy spectrum. As we can see that the most energy of low frequencies arrives at the listener from the bottom and the ceiling, since they do not absorb low frequencies. Whereas middle and high frequencies reach the listener from the walls. Considering the energy at 80 Hz (figure 6 (b)) and at 1280 Hz (figure 6 (c)) we can observe that most part of the energy at 80 Hz reflects at the bottom and ceiling, and at 1280 Hz most part is reflected at the walls.

4 Discussion

In this work we have presented various visualization approaches for analyzing acoustic behavior inside a room by use of the phonon map resulting from our phonon tracing algorithm. First of all we visualized the single phonons as color coded spheres. The advantages of this approach is the simplicity of the technique and the direct visibility of the material influence on the sound traversed from the source. A huge drawback is the lack of connectivity information between the phonons. This is overcome by our second approach, the visualization of wave fronts. Additionally this method is a natural representation of the propagation of sound.
Due to increasing fragmentation, this method can only be used for the first few reflections and there is only limited time dependency. To visualize the reverberations of higher order at the scene surfaces and to include the time dependency in the visualization, we used scattered data interpolation. At the moment, we restricted this method to the surface representation and neglect the direction of the particular phonons. In these three presented methods we disregarded the situation at certain listener positions. Therefore we introduced the listener-based visualization approach. This technique allows a time dependent view on the received energy on a certain listener position.
In total, these approaches give a general idea over the acoustic behavior inside the considered scene which can be derived from the phonon map. Further work has to be done in the area of interpolation to incorporate the outgoing direction of the phonons so that it can be used in the simulation process. For a better assessment of hearing experience it would be favorable to look at the different acoustic metrics at the listener positions, as well.

References

[BDM$^+$05] M. Bertram, E. Deines, J. Mohring, J. Jegorovs, and H. Hagen. Phonon Tracing for Auralization and Visualization of Sound. In *IEEE Visualization*, Minneapolis, MN, October 2005.

[Bos] Bose Corporation. Bose Modeler. *http://www.bose.com*.

[CAT] CATT. CATT-Acoustic. *http://www.catt.se*.

[CGA] CGAL. Computational Geometry Algorithms Library. *http://www.cgal.org*.

[DBM$^+$06] E. Deines, M. Bertram, J. Mohring, J. Jegorovs, F. Michel, H. Hagen, and G.M. Nielson. Comparative Visualization for Wave-based and Geometric Acoustics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5), Nov. 2006.

[DMB$^+$06] E. Deines, F. Michel, M. Bertram, H. Hagen, and G. Nielson. Visualizing the Phonon Map. In *Eurovis06*, Portugal, Spring 2006.

[FCE$^+$98] 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.

[Jen96] Henrik Wann Jensen. Global Illumination using Photon Maps. In *Rendering Techniques '96 (Proceedings of the 7th Eurographics Workshop on Rendering)*, pages 21–30, 1996.

[KFW98] 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.

[KJM04] 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.

[Kro68] U. Krockstadt. Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound and Vibrations*, 8(18), 1968.

[Kul84] U. Kulowski. Algorithmic Representation of the Ray Tracing Technique. *Applied Acoustics*, 18:449–469, 1984.

[ODE] ODEON. Room Acoustic Software. *http://www.odeon.dk*.

[SG89] 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.

[Vor89] 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.

Computational Modelling of Micromorphic Continua – Theory, Numerics, and Visualisation Challenges

C. Britta Hirschberger*, Ellen Kuhl, and Paul Steinmann
University of Kaiserslautern
Department of Mechanical Engineering, Chair of Applied Mechanics,
D-67653 Kaiserslautern, Germany
*bhirsch@rhrk.uni-kl.de

Abstract: In order to account for scale-dependence effects that are imposed by the microstructural constitution of certain materials, we employ a micromorphic continuum theory. Therein microcontinua are assumed to be attached to each physical point. They may experience arbitrary homogeneous deformation, which are kinematically independent from the macroscale. The additional kinematical quantities, which account for the micro-deformation, yield additional stresses and contributions to the balance of momentum. As a consequence we obtain both non-symmetric second-order as well as higher-order stress tensors. These among other non-symmetric tensors, which will be introduced, provide an additional challenge in visualisation compared to standard continua.

1 Introduction

Many engineering materials possess a distinct microstructure. Examples for such materials are: geomaterials, which are often a conglomerate or a gathering of innumerable single solid grains, secondly metals which exhibit a relatively small granular substructure consisting of polycrystals or furthermore biomaterials such as blood as a fluid containing solid particles (see Figure 1). The influence of these microstructures may be considered negligible if we consider specimens many orders larger than their dimension. However, at scales of observation approaching this microscale, their influence on the material behaviour will generally become significant. Particular *microcontinuum* theories have been developed to account for the effects of the substructure, especially the scale-dependence. Among such theories, the so-called micromorphic continuum theory, which is treated here, represents a rather general case with respect to the kinematics of the microcontinua, see e.g. [Eri99, KS05, HKS06]. The micromorphic continuum is defined such that a microcontinuum is assigned to each physical point of the body. These microcontinua may experience both stretch and rotation, which are homogeneous throughout the microcontinuum, nevertheless kinematically independent from the macrocontinuum deformation.

In continuum solid mechanics, usually deformation processes of a material body over time are viewed with respect to the initial or rather *material* configuration. Within this

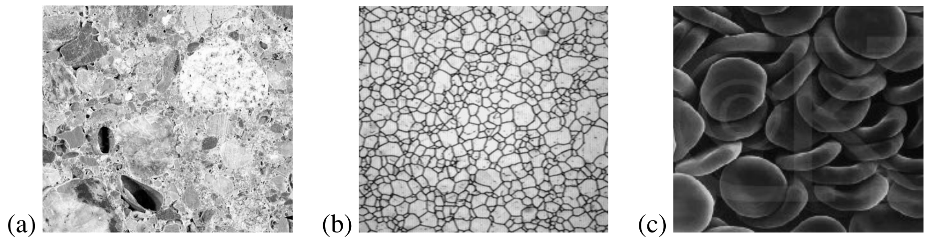


Figure 1: Examples for materials with microstructure: (a) conglomerate (geomaterial), (b) steel possessing a granular microstructure, c) blood with red blood cells

perspective, which we call the *spatial-motion problem*, the occurring quantities are parameterised with respect to material position. The opposite perspective called *material-motion problem*—wherein the spatial placement is fixed while the material position is the unknown quantity of observation and all quantities are parameterised with respect to spatial reference—can be utilised in order to obtain so-called *material forces*. These are anticipated to act as driving forces for the propagation of inhomogeneities such as cracks and voids in the literature [Mau93, Ste00, Gur00, MS05].

In our research we have applied both perspectives to the micromorphic continuum [HKS06]. In the current contribution we will present the essential continuum-mechanical and numerical framework in order to point out the key visualisation tasks and challenges.

2 Theory

As indicated before, in the description of our micromorphic continuum we distinguish between the macro- and the micro-scale. On the macroscale we consider the behaviour of a physical point $\mathcal{P}$ with the material and the spatial placement vector, $\boldsymbol{X}$ in $\mathcal{B}_0$ and $\boldsymbol{x}$ in $\mathcal{B}_t$, respectively. In the microcontinuum, which is attached to each physical (macro) point, a micro point $\bar{\mathcal{P}}$ occupies the material micro-position $\bar{\boldsymbol{X}}$ in $\bar{\mathcal{B}}_0$ and the spatial one $\bar{\boldsymbol{x}}$ in $\bar{\mathcal{B}}_t$, respectively. Herein, $(\bullet)_0$ represents a quantity at *material* time $t = 0$ and $(\bullet)_t$ at *spatial* time $t > 0$. Furthermore quantities of the microscale are denoted by $\bar{\bullet}$, see Figure 2(a) for illustration.

2.1 Kinematics

The kinematic relations, which characterise the deformation of both macro- and microcontinuum, are summarised in Table 1. Herein $\boldsymbol{\varphi}$ is a vector, whereas both $\boldsymbol{F}$ and $\bar{\boldsymbol{F}}$ represent second-order tensors, furthermore the gradient [1] of the latter quantity, $\bar{\boldsymbol{G}}$, represents a tensor of third order.

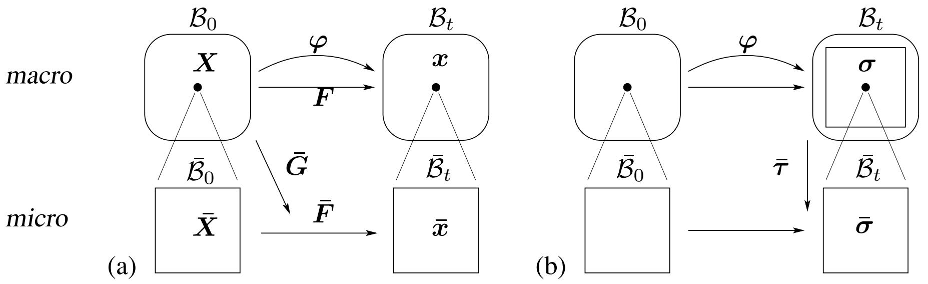


Figure 2: Configurational view: (a) Micromorphic deformation maps of the spatial-motion problem. (b) Cauchy-type stresses and their spatial character

		macro			*micro*	
deformation map	$\boldsymbol{x}$	$=$	$\varphi(\boldsymbol{X})$	$\bar{\boldsymbol{x}}$	$=$	$\bar{\boldsymbol{F}}(\boldsymbol{X}) \cdot \bar{\boldsymbol{X}}$
tangent map	$\mathrm{d}\boldsymbol{x}$	$=$	$\boldsymbol{F}(\boldsymbol{X}) \cdot \mathrm{d}\boldsymbol{X}$			
gradient	$\boldsymbol{F}(\boldsymbol{X})$	$:=$	$\nabla_{\boldsymbol{X}}\varphi(\boldsymbol{X})$	$\bar{\boldsymbol{G}}(\boldsymbol{X})$	$:=$	$\nabla_{\boldsymbol{X}}\bar{\boldsymbol{F}}(\boldsymbol{X})$

Table 1: Micromorphic kinematics: Deformation mappings and variables for the macro- and the microscale

2.2 Balance Relations for Energy and Momentum

For the derivation of the balance relations, we avail ourselves of the fact that for the present quasi-static case, the total potential energy must be stationary for the system to be in equilibrium. Upon the assumption of hyperelasticity we may thus introduce stresses as energetically conjugate quantities to the aforementioned deformation variables at fixed material placement: We define the macro-, the micro-, and the double-stress of Piola type as

$$\boldsymbol{P} := \mathsf{D}_{\boldsymbol{F}} U_0 \,, \qquad \bar{\boldsymbol{P}} := \mathsf{D}_{\bar{\boldsymbol{F}}} U_0 \,, \qquad \bar{\boldsymbol{Q}} := \mathsf{D}_{\bar{\boldsymbol{G}}} U_0 \,, \tag{1}$$

respectively. Herein the scalar U_0 represents the stored energy density per unit volume. The orders of these tensors follow those of their energetically conjugate partners. With these at hand, by considering the weak form and application of the Gauss theorem, we achieve the local form of the balance of momentum[2]as arranged in Table 2. It consists of a macro- and a micro-term and is supplemented by homogeneous natural boundary conditions. Since, due to their introduction in (1), the Piola-type stresses belong to both the spatial and the material configuration, they are problematic to visualise. Thus we perform a Piola transformation to obtain a purely spatial description: the alternative representation

[1]The derivative of any quantity $\circ$ with respect to any tensor $\bullet$ reads $\partial_\bullet \circ := \frac{\partial \circ}{\partial \bullet}$, while the gradient of $\circ$ with respect to material placement is denoted by $\nabla_{\boldsymbol{X}} \circ$. Furthermore $\mathsf{D}_\bullet \circ := \left.\frac{\partial \circ}{\partial \bullet}\right|_{\boldsymbol{X}}$ is used for the derivative of $\bullet$ with respect to a variable $\circ$ at fixed material placement $\boldsymbol{X}$.

Piola-type	balance of momentum	natural boundary condition
macro	$\mathrm{Div}\,\boldsymbol{P} = \boldsymbol{0}$ in $\mathcal{B}_0$	$\boldsymbol{P}\cdot\boldsymbol{N} = \boldsymbol{0}$ on $\partial\mathcal{B}_0^{T_P}$
micro	$\mathrm{Div}\,\bar{\boldsymbol{Q}} - \bar{\boldsymbol{P}} = \boldsymbol{0}$ in $\bar{\mathcal{B}}_0$	$\bar{\boldsymbol{Q}}\cdot\boldsymbol{N} = \boldsymbol{0}$ on $\partial\bar{\mathcal{B}}_0^{T_{\bar{Q}}}$

Cauchy-type	balance of momentum	natural boundary condition
macro	$\mathrm{div}\,\boldsymbol{\sigma} = \boldsymbol{0}$ in $\mathcal{B}_t$	$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \boldsymbol{0}$ on $\partial\mathcal{B}_t^{t_\sigma}$
micro	$\mathrm{div}\,\bar{\boldsymbol{\tau}} - \bar{\boldsymbol{\sigma}} = \boldsymbol{0}$ in $\bar{\mathcal{B}}_t$	$\bar{\boldsymbol{\tau}}\cdot\boldsymbol{n} = \boldsymbol{0}$ on $\partial\bar{\mathcal{B}}_t^{t_{\bar{\tau}}}$

Table 2: Local balance of momentum and according natural boundary conditions in terms of Piola-type and Cauchy-type stress tensors

of the balance of momentum in terms of the purely spatial Cauchy-type stresses is given in Table 2 and their character is illustrated in Figure 2(b). Particularly the macro-, micro- and double-stress of both stress types are related to each other by [3]

$$\boldsymbol{\sigma} := j\boldsymbol{P}\cdot\boldsymbol{F}^{\mathrm{t}}\,, \qquad \bar{\boldsymbol{\sigma}} := j[\bar{\boldsymbol{P}}\cdot\bar{\boldsymbol{F}}^{\mathrm{t}} + \bar{\boldsymbol{Q}} \overset{2,3}{:} \bar{\boldsymbol{G}}]\,, \qquad \bar{\boldsymbol{\tau}} := j\bar{\boldsymbol{Q}} : [\bar{\boldsymbol{F}}^{\mathrm{t}}\overline{\otimes}\boldsymbol{F}^{\mathrm{t}}]\,. \quad (2)$$

2.3 Constitutive Law

A constitutive assumption provides a relation between deformation and stress. For our assumption of hyperelasticity (compare (1)), we introduce the following straightforward formulation for the stored energy density U_0 (neglecting external potentials)

$$U_0 = \frac{1}{2}[\lambda \ln^2 J + \mu[\boldsymbol{F} : \boldsymbol{F} - n^{\mathrm{dim}} - 2\ln J] + \mu l^2 \bar{\boldsymbol{G}} \,\vdots\, \bar{\boldsymbol{G}} + p[\bar{\boldsymbol{F}} - \boldsymbol{F}] : [\bar{\boldsymbol{F}} - \boldsymbol{F}]]\,, \quad (3)$$

which consists of a term of neo-Hooke type for the macroscale, a simple quadratic formulation for the microscale and an additional scale-transition term. The material parameters incorporated are the Lamé constants λ and μ as well as the internal length l and the scale-transition parameter p, which plays the part of a penalty controlling the interaction between the macro- and the micro-deformation [KS05, HKS06]. With this formulation at hand, the different stress fields may be evaluated, utilising the definitions (1) and the relations (2).

[2]The divergence operators are identified as $\mathrm{Div}\,\bullet := \nabla_{\boldsymbol{X}}(\bullet) : \boldsymbol{I}$ and $\mathrm{div}\,\bullet := \nabla_{\boldsymbol{x}}(\bullet) : \boldsymbol{I}$.

[3]Between two third-order tensors the product $\boldsymbol{A} \overset{i,j}{:} \boldsymbol{B}$ denotes the contraction over the i-th and the j-th index and thus yields a second-order tensor. The modified dyadic product is defined through the relation $[\boldsymbol{A} \overline{\otimes} \boldsymbol{B}] : \boldsymbol{C} = \boldsymbol{A}\cdot\boldsymbol{C}\cdot\boldsymbol{B}^{\mathrm{t}}$, wherein $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are tensors of second order.

3 Numerics

For the numerical simulation of boundary-value problems, the finite-element method is used. According to the continuum theory presented in Section 2, a finite-element approximation scheme is presented. The primary unknown variables, any boundary-value problem is to be solved for, we have chosen to be the macro- and the micro-deformation map, $\boldsymbol{\varphi}$ and $\bar{\boldsymbol{F}}$. In a postprocessing step, the stress fields are computed and the material-force method is applied, which is anticipated to yield insight into the behaviour for instance at heterogeneities.

3.1 Finite-Element Approximation Scheme

The use of a standard Bubnov-Galerkin approach allows the same interpolation functions, N_{I}^{φ} and $N_{\mathrm{J}}^{\bar{F}}$, for the approximation of both trial and test functions. The discrete approximations or trial functions read

$$\boldsymbol{\varphi}^h = \sum_{\mathrm{L}=1}^{n_\varphi} N_{\mathrm{L}}^{\varphi} \boldsymbol{\varphi}_{\mathrm{L}} \in \mathcal{V}^h \qquad \text{and} \qquad \bar{\boldsymbol{F}}^h = \sum_{\mathrm{M}=1}^{n_{\bar{F}}} N_{\mathrm{M}}^{\bar{F}} \bar{\boldsymbol{F}}_{\mathrm{M}} \in \bar{\mathcal{V}}^h \tag{4}$$

Unlike the trial functions, the test functions are required to be zero on the essential boundaries, i.e. $\delta\boldsymbol{\varphi}^h \in \mathcal{V}_0^h$, $\delta\bar{\boldsymbol{F}} \in \bar{\mathcal{V}}_0^h$. The constitutive law (3) necessitates the use of approximations of different order within the finite elements. Particularly, we have chosen quadrilateral Lagrangian plane elements with a bi-quadratic approximation for the macro-deformation map $\boldsymbol{\varphi}$ and a bi-linear approximation for the micro-deformation map $\bar{\boldsymbol{F}}$; confer [Kou02] and references cited therein.

To obtain a system of equations, the weak from of the balance of momentum in Table 2 is discretised. From that we obtain the discrete residual vectors as

$$\begin{array}{lclcl} \mathbf{R}_{\mathrm{I}}^{\varphi} &=& \displaystyle\int_{\mathcal{B}_0} \boldsymbol{P} \cdot \nabla_{\boldsymbol{X}} N_{\mathrm{I}}^{\varphi} \mathrm{d}V & - \ \mathbf{F}_{\mathrm{I}}^{\varphi\,\mathrm{ext}} & \stackrel{!}{=} \mathbf{0} \\ \mathbf{R}_{\mathrm{J}}^{\bar{F}} &=& \displaystyle\int_{\mathcal{B}_0} \bar{\boldsymbol{Q}} \cdot \nabla_{\boldsymbol{X}} N_{\mathrm{J}}^{\bar{F}} \mathrm{d}V + \int_{\mathcal{B}_0} \bar{\boldsymbol{P}} N_{\mathrm{J}}^{\bar{F}} \mathrm{d}V & - \ \mathbf{F}_{\mathrm{J}}^{\bar{F}\,\mathrm{ext}} & \stackrel{!}{=} \mathbf{0} \end{array} \tag{5}$$

with the Piola-type stresses determined by (1) with (3). Consequently the following linearised coupled problem is to be solved for increments of $\boldsymbol{\varphi}_{\mathrm{L}}^h$ and $\bar{\boldsymbol{F}}_{\mathrm{M}}^h$:

$$\begin{bmatrix} \mathbf{K}_{\mathrm{IL}}^{\varphi\varphi} & \mathbf{K}_{\mathrm{IM}}^{\varphi\bar{F}} \\ \mathbf{K}_{\mathrm{JL}}^{\bar{F}\varphi} & \mathbf{K}_{\mathrm{JM}}^{\bar{F}\bar{F}} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\varphi}_{\mathrm{L}}^h \\ \Delta\bar{\boldsymbol{F}}_{\mathrm{M}}^h \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{\mathrm{I}}^{\varphi\,\mathrm{ext}} \\ \mathbf{F}_{\mathrm{J}}^{\bar{F}\,\mathrm{ext}} \end{bmatrix} - \begin{bmatrix} \mathbf{F}_{\mathrm{I}}^{\varphi\,\mathrm{int}} \\ \mathbf{F}_{\mathrm{J}}^{\bar{F}\,\mathrm{int}} \end{bmatrix} . \tag{6}$$

Herein the component matrices of the global tangent stiffness matrix are given by

$$\mathbf{K}_{\mathrm{IL}}^{\varphi\varphi} := \frac{\partial \mathbf{R}_{\mathrm{I}}^{\varphi}}{\partial \boldsymbol{\varphi}_{\mathrm{L}}}, \qquad \mathbf{K}_{\mathrm{IM}}^{\varphi\bar{F}} := \frac{\partial \mathbf{R}_{\mathrm{I}}^{\varphi}}{\partial \bar{\boldsymbol{F}}_{\mathrm{M}}}, \qquad \mathbf{K}_{\mathrm{JL}}^{\bar{F}\varphi} := \frac{\partial \mathbf{R}_{\mathrm{J}}^{\bar{F}}}{\partial \boldsymbol{\varphi}_{\mathrm{L}}}, \qquad \mathbf{K}_{\mathrm{JM}}^{\bar{F}\bar{F}} := \frac{\partial \mathbf{R}_{\mathrm{J}}^{\bar{F}}}{\partial \bar{\boldsymbol{F}}_{\mathrm{M}}} . \tag{7}$$

The actual computation of the stiffness matrix and the residual vector is done element-wise (utilising the isoparametric concept with Jacobian transformations), before these are assembled to a global system of equations, which is solved, as in standard finite-element procedures. For the numerical evaluation of the appearing integrals within the finite elements, we avail ourselves of the common Gauss quadrature.

3.2 Application of the Material-Force Method

We apply the material force method to the micromorphic continuum in the spirit of the work of our group, e.g. [Ste00, SAB01]. Therefore we employ the two equations of the balance of momentum (compare Table 2) in the spirit of the inverse problem, multiply them with the respective energetically conjugate variations, $\delta\boldsymbol{\Phi}$ or $\delta\bar{\boldsymbol{f}}$ respectively. After some manipulations and upon the assumption that the relations hold for arbitrary variations, we discretise the equations and define the nodal *material force* $\mathfrak{F}_\mathrm{L}$ of the macroscale on the one hand and the *nodal configurational double-force* $\bar{\mathfrak{m}}_\mathrm{J}$ on the microscale on the other hand as

$$\mathfrak{F}_\mathrm{L} := \int_{\mathcal{B}_t} \boldsymbol{p} \cdot \nabla_{\boldsymbol{x}} N_\mathrm{L}^{\Phi} \mathrm{d}v \,, \qquad \bar{\mathfrak{m}}_\mathrm{J} := \int_{\mathcal{B}_t} [\bar{\boldsymbol{q}} \cdot \nabla_{\boldsymbol{x}} N_\mathrm{J}^{\bar{f}} + \bar{\boldsymbol{p}} N_\mathrm{J}^{\bar{f}}] \mathrm{d}v \,. \tag{8}$$

According to their introduction, the vector of the material macro force, $\mathfrak{F}_\mathrm{L}$, is energetically conjugate to the material virtual node-point displacements $\delta\boldsymbol{\Phi}_\mathrm{L}$. Analogously the nodal configurational micro double-force is energetically conjugate to the micro-deformation map $\bar{\boldsymbol{f}}$ of inverse problem and it represents a nonsymmetric tensor of second order being present at the finite-element nodes. Note that herein—for the inverse problem—the approximation provided in (4) are analogously used and the Piola-type stress tensors $\boldsymbol{p}$, $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{q}}$ are determined analogously to (1).

4 Simulation and Visualisation

In order to present some features of the theory and point out open challenges in visualisation, we perform numerical simulations. Here we exemplarily show a rectangularly shaped specimen with a static crack under longitudinal uniaxial loading, which will lead to mode-I crack opening. The geometry is discretised with finite elements described before, and the boundary-value problem is solved.

Concerning visualisation, we may distinguish between two groups within the resulting quantities: The first of which consist of field quantities that are continuously distributed within element domains. Such quantities, for instance the stresses, are usually evaluated at the integration points and globally projected to the finite-element nodes. The other major group of data results from nodal equilibrium considerations and is thus exclusively evaluated at the nodal points of the finite-element mesh. Hereby the nodal material forces and configurational double forces as results from the material-force method need to mentioned.

4.1 Visualisation of Element-Wise Continuous Quantities

The visualisation of tensor fields is a non-trivial task in continuum solid mechanics, and here with the presence of tensors of higher order, it becomes even more involved. The data which is discretely evaluated at the integration point, is projected to the global element nodes by a global algorithm based on least-square fits and then plotted. Doing so, one obtains a smooth stress distribution of single tensor components with values at FE nodes. Figure 3 shows plots of all the occurring components of the Cauchy-type stresses. Note that in 2d, a second-order tensor is characterised by $2 \times 2 = 4$ components, while a third-order tensor has $2 \times 2 \times 2 = 8$ components. From the plots of single components—especially in the case of the third-order double-stress tensor—we may only arduously recognise any significant characteristics of the stress tensors, besides the non-symmetry. Note that for the micromorphic continuum, generally none of these stress tensors is symmetric; however certain symmetries in the shear components occur for special choices of parameters.

4.2 Visualisation of Discrete Nodal Quantities

The second challenge results from the quantities that we obtained from the material-force method: the nodal material macro force vectors and the nodal configurational micro double-force tensors are both evaluated at each node of the finite element mesh and require an appropriate visualisation. The first quantity, the material-force vectors can be easily visualised as arrows, as seen in Figure 4 for a variation of the internal length l. At boundary nodes with prescribed displacements these material forces represent the support reactions, while for instance at the crack tip the material force will always be directed opposite to a (potential) crack propagation direction.

In contrast to that, the visualisation of the nodal configurational double-force resulting from the microstructure, however, proves itself as a demanding issue, since this quantity is a second-order tensor to begin with, and furthermore it is nonsymmetric. In Figure 5 we have attempted various visualisation of the tensors, by solving an eigenvalue problem, spectral decomposition into a rotation and stretch, and the invariants plotted as if they were continuously distributed. Any of the eigenvectors are scaled by their corresponding eigenvalues, which facilitates to recognise a characteristic structure.

4.3 Visualisation Challenges

However, we still seek for a more elaborate visualisation technique which reveals the character of the occurring non-symmetric tensor quantities more clearly. In this context, we hope to mutually benefit from the interdisciplinary character of the DFG IRTG 1131.

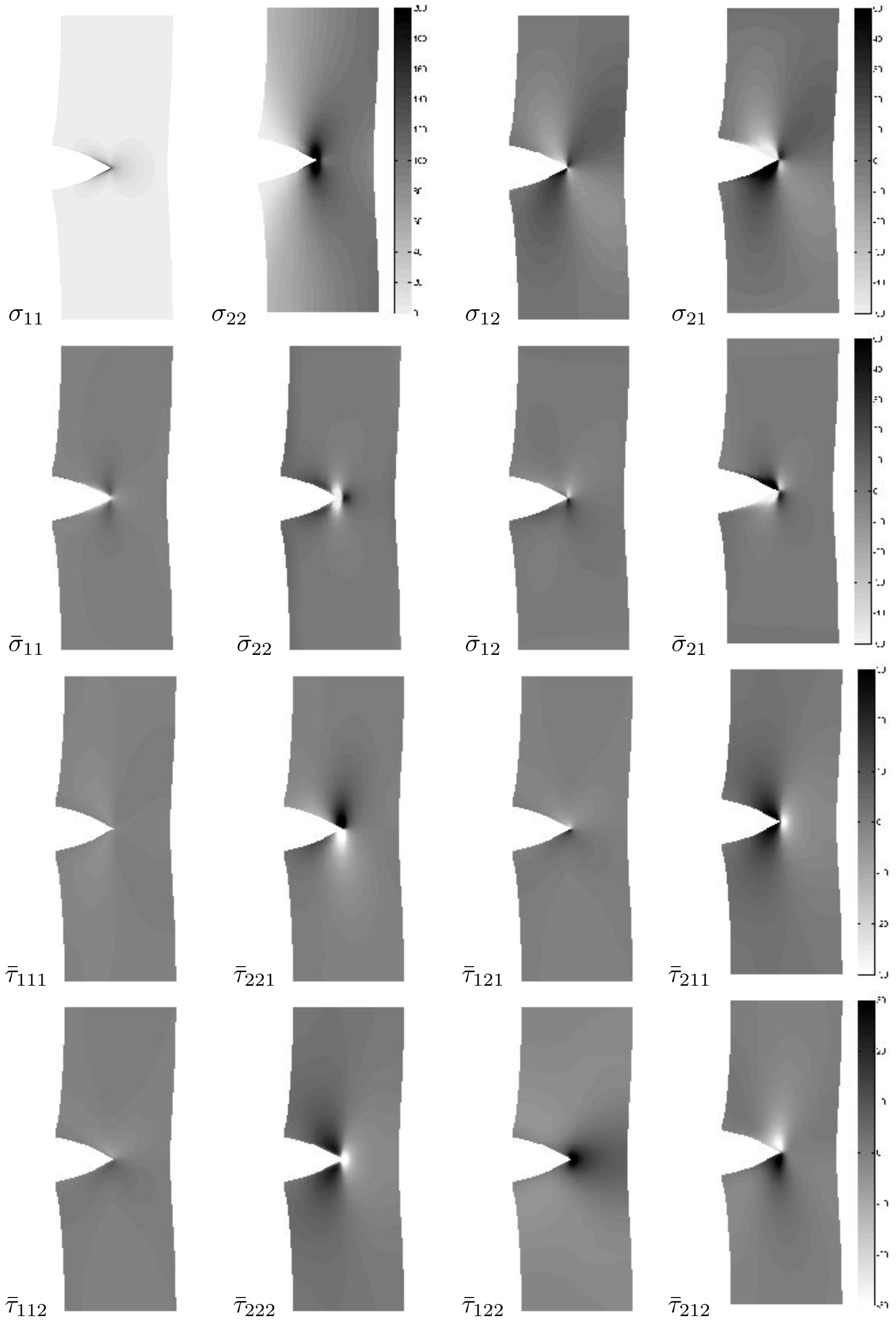


Figure 3: Component-wise plot of the stress fields of Cauchy type: macro-stress $\boldsymbol{\sigma}$, micro-stress $\bar{\boldsymbol{\sigma}}$, and double-stress $\bar{\boldsymbol{\tau}}$, all for $p = 20E$, $l = L_0/20$

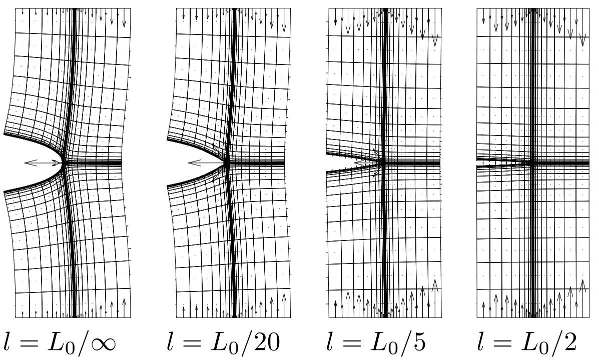


Figure 4: Material forces $\mathfrak{F}_{\mathrm{L}}$ at spatial mesh for variation of the internal length at $p = 20E$

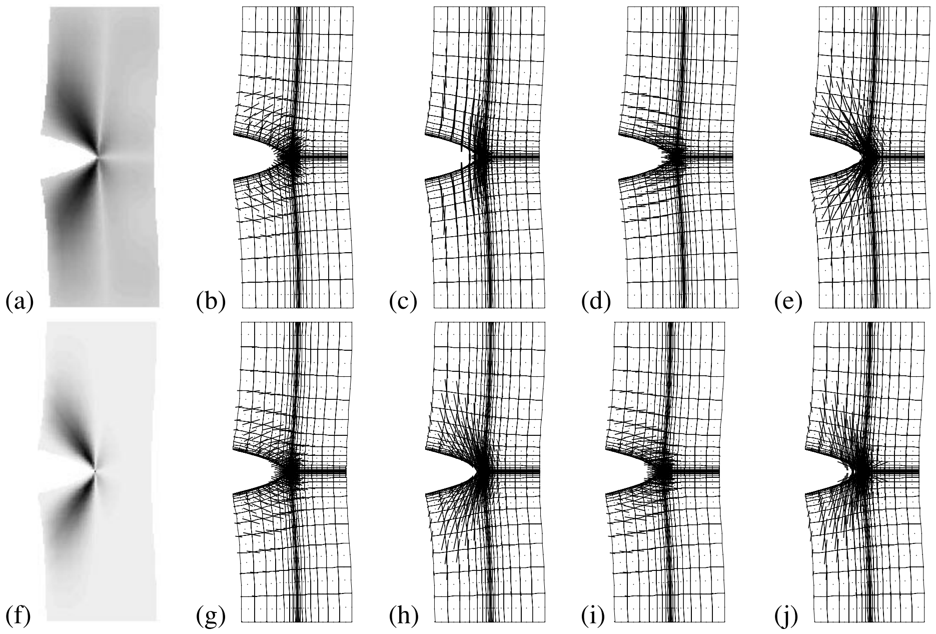


Figure 5: Configurational nodal micro double-forces $\bar{\mathfrak{m}}_{\mathrm{J}}$ ($l = L_0/20$, $p = 20E$): invariants: (a) trace, (f) determinant. (b)–(e) first and second left as well as first and second right eigenvectors of the tensor, scaled by length of eigenvalues. (g)–(h) material tensor from spectral decomposition: first, second scaled eigenvector. (g)–(h) spatial tensor from spectral decomposition: first, second scaled eigenvector

References

[Eri99] A. C. Eringen. *Microcontinuum Field Theories: I. Foundations and Solids*. Springer, New York, 1999.

[Gur00] M. E. Gurtin. *Configurational Forces as Basic Concepts of Continuum Physics*, volume 137 of *Applied Mathematical Sciences*. Springer, 2000.

[HKS06] C. B. Hirschberger, E. Kuhl, and P. Steinmann. On Deformational and Configurational Mechanics of Micromorphic Hyperelasticity. Theory and Computation. submitted for publication, 2006.

[Kou02] V. G. Kouznetsova. *Computational homogenization for the multiscale analysis of multi-phase materials*. PhD thesis, Technische Universiteit Eindhoven, 2002.

[KS05] N. Kirchner and P. Steinmann. A unifying treatise on variational principles for gradient and micro-morphic continua. *Phil. Mag.*, 85:3875–3895, 2005.

[Mau93] G. A. Maugin. *Material Inhomogeneities in Elasticity*. Chapman & Hall/CRC, 1993.

[MS05] G. A. Maugin and P. Steinmann. *Mechanics of Material Forces*, volume 11 of *Advances in Mechanics and Mathematics*. Springer, 2005.

[SAB01] P. Steinmann, D. Ackermann, and F. J. Barth. Application of material forces to hyperelastostatic fracture mechanics. II. Computational setting. *Int. J. Solid Struct.*, 38:5509–5526, 2001.

[Ste00] P. Steinmann. Application of material forces to hyperelastostatic fracture mechanics. I. Continuum mechanical setting. *Int. J. Solid Struct.*, 37:7371–7391, 2000.

On Discrete Modeling and Visualization of Granular Media

Holger A. Meier, Ellen Kuhl, and Paul Steinmann
University of Kaiserslautern
Chair of Applied Mechanics
D-67653 Kaiserslautern, Germany
homei@rhrk.uni-kl.de

Abstract: Granular media have been gaining much attention in the last few years. Several attempts have been made to model the behavior of granular media, including continuous and dis-continuous approaches. The dis-continuous structure of the particulate media calls for methods being able to map the discontinuities between the single grains, for example the discrete element method. Assuming periodicity inside the granular assembly, a periodic representative volume element is set up. The production of the periodic representative volume element is the focal point of our paper. We would like to draw the attention of the geo-mechanical community to a class of particle packing methods developed in computational chemistry. These packing methods are very efficient and their outcome is congruent with the expectations of a periodic representative volume element used in geo-mechanical multiscale methods. Properties which are intrinsically fulfilled include the geometric periodicity of the representative volume element boundary as well as an irregular highly dense packing of the periodic representative volume element itself. The presented algorithm allows the straight forward generation of polydisperse packings related to given sieve curves taken into account via prescribed growth rates.

1 Introduction

Analysis of granular media is accomplished in many ways. Next to the approach of assuming a continuum, a dis-continuum can be set up. This dis-continuum, consisting of discrete elements ([DM04]), is capable of capturing the behavior of granular media, validated by Cundall and Strack. Furthermore, Cundall and Strack introduced the discrete element method (dem)([CS78],[CS79b]), a method to analyze the behavior of granular media [CS79a]. Embedded on the microscale of a homogenization approach (see Zohdi, [ZW05]), the discrete element method can be used as a powerful and effective tool to model structures consisting of granular media. In contrast to methods based on a continuum approach, the discrete element method allows the single grains to form and break contacts. This behavior is distinguishing for particulate media and is directly related to the failure behavior known from granular assemblies. Due to the enormous number of grains inside a particulate medium it is not possible to simulate each grain. Hence, a simplification, assuming that the particulate medium is periodic, has to be made. The assumption of periodicity allows to set only the focal point on a limited number of particles (nop), called periodic volume element. Assuming that the number of grains inside the periodic volume element are capable of reproducing the behavior of the whole granular medium, the periodic volume element can be considered to be representative (rve). Introducing this kind of sample, the periodic rve, enables to restrict the computational effort to the smallest, still representative, material sample. Each rve consists of a limited number of particles and incorporates a geometric periodic boundary of particles. Although theoretically well understood, the efficient construction of an rve having a high volume fraction and at the same time being irregular regarding the particle setup is still part of current research. One outstanding method, developed to produce irregular highly packed particle assemblies having geometric periodic boundaries was developed by Stillinger and Lubachevsky [BDL91], [Lub91]. This method is based on a hard particle contact model and fulfills the needs of dense packing, irregularity inside the rve and incorporates geometric periodic boundaries. In the following, this method will be introduced and applied to produce irregular highly packed granular rves having geometric periodic boundaries. It is noted that only the two dimensional case is considered here, whereas any other dimension can be tackled with the same concept.

2 Generation of Representative Volume Elements

The generation of the rve is accomplished by employing an event driven scheme, advancing from event to event. In an event driven scheme, the event dictates the time step size. Here, an event is considered to be the discrete collision between two particles. Each event is considered individually and in serial, postulating that only one discrete event is taking place at one discrete time. This leads to the possibility to handle each event separately. The algorithm used to produce the rve is described in Subsection 2.3, where as the basic steps of finding and handling an event are specified in Subsection 2.1 and 2.2.

2.1 Time Step Calculation

To calculate the time step ${}^{n}\Delta t_{c}$ which is needed to advance the particle system from time ${}^{n}t$ (Fig. 1, left) to time ${}^{n+1}t$ (Fig. 1, right) the event, collision between two particles, has to be observed. Since we are using a hard contact model we do not allow any overlap of the particles. Entering at time ${}^{n}t$ we assume that the positions of the particle centers, the particle radii as well as the particle velocities are known. Furthermore, we postulate that the particle velocities are not zero and that particle i and j are currently not in contact, see Fig. 1. The particle radii r_i are assumed to increase as

$$\mathsf{D}_t\, r_i = g_i \quad \forall \quad i \in \{1, \ldots, \mathsf{nop}\}\,, \quad r_i\,, g_i \in \mathbb{R} \tag{1}$$

whereby we assume a constant growth rate $g_i = \mathsf{const.}$. The discrete counterpart of (1) can be constructed, e.g. with the help of a finite difference scheme, i.e. $\mathsf{D}_t\, r_i = \left[{}^{n+1}r_i - {}^{n}r_i\right]/\Delta t$, yielding the discrete update equation of the particle radii at time

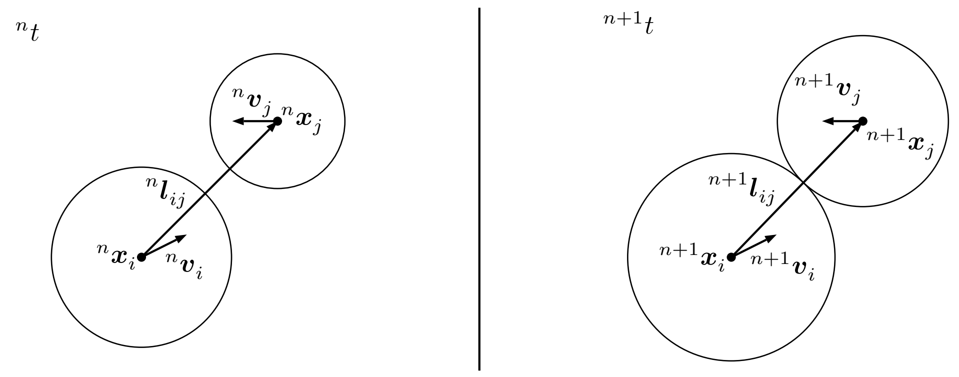


Figure 1: Configuration of particle i and j at time ^{n}t and ^{n+1}t. Left: Particles i and j at time ^{n}t with centers $^{n}\boldsymbol{x}_i$ and $^{n}\boldsymbol{x}_j$, having radii $^{n}r_i$ and $^{n}r_j$ and velocities $^{n}\boldsymbol{v}_i$ and $^{n}\boldsymbol{v}_j$. Right: Event position of particle i and j, with center positions $^{n+1}\boldsymbol{x}_i$ and $^{n+1}\boldsymbol{x}_j$, radii $^{n+1}r_i$ and $^{n+1}r_j$ and velocities $^{n+1}\boldsymbol{v}_i$ and $^{n+1}\boldsymbol{v}_j$. The branch vector $^{n+1}\,\boldsymbol{l}_{\,ij}$ connects the centers of the particles i and j with length equal to the sum of the radii of the particles i and j.

^{n+1}t to be equal to (2).

$$^{n+1}r_i = {^{n}r_i} + g_i\,{^{n}\Delta t} \quad \forall \quad i \in \{1,\ldots,\mathsf{nop}\} \tag{2}$$

In the present case, without periodic boundary conditions, the particle velocities and growth rates are selected to ensure a collision. Assuming periodic boundary conditions, a collision between the particles is guaranteed without restricting [1] the particle velocities or growth rates, see proof at the end of this Subsection. Postulating a constant velocity of particle i between the time nodes, the position of particle i at time ^{n+1}t is calculated by using the well known forward Euler formula.

$$^{n+1}\boldsymbol{x}_i = {^{n}\boldsymbol{x}_i} + {^{n}\Delta t}\,{^{n}\boldsymbol{v}_i} \quad \forall \quad i \in \{1,\ldots,\mathsf{nop}\}\,, \quad \boldsymbol{x}_i\,,\boldsymbol{v}_i \in \mathbb{R}^n \tag{3}$$

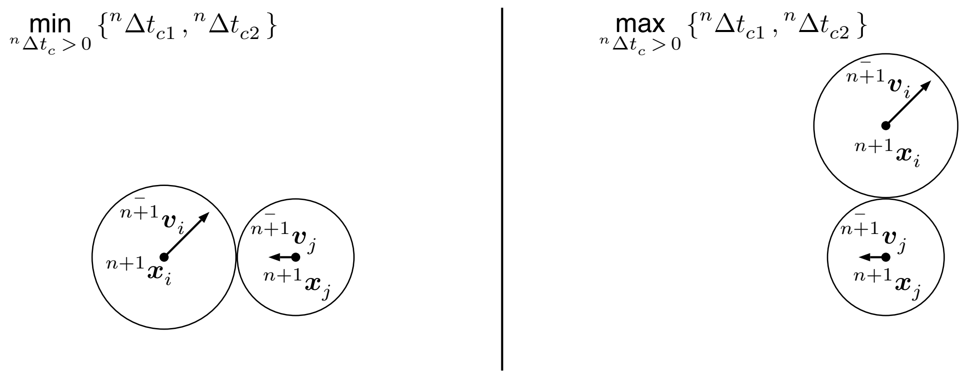
The increase of the volume fraction between the particles and the periodic boundary box is controlled by the growth rate g_i. If g_i is equal for all particles i, a monodisperse packing is constructed, while different growth rates g_i generate a polydisperse packing ([ARK02]). The growth rate, $g_i > 0$, is integrated in the particle radius evolution formula (2), increasing the radii depending on the time increment $^{n}\Delta t$. The branch vector $^{n+1}\boldsymbol{l}_{ij}$ is calculated by subtracting the position vectors of the particles.

$$^{n+1}\boldsymbol{l}_{ij} = {^{n+1}\boldsymbol{x}_j} - {^{n+1}\boldsymbol{x}_i} \quad \forall \quad i \neq j, \qquad i,j \in \{1,\ldots,\mathsf{nop}\} \tag{4}$$

The branch vector connects the centers of the particles i and j. Its absolute value measures the distance between the particle pair i and j. A contact condition, consisting of the absolute value of the branch vector as well as the particle radii can be set up.

$$\|{^{n+1}\boldsymbol{l}_{ij_c}}\| = {^{n+1}r_{i_c}} + {^{n+1}r_{j_c}} \tag{5}$$

[1]This still implies the postulation of zero overlapping between the particles which leads to a restriction of the largest radius, driven by the largest growth rate, to be less or equal to the half of the length of the rve, see Fig.3



$\min_{{}^n\Delta t_c > 0} \{{}^n\Delta t_{c1}, {}^n\Delta t_{c2}\}$
$\max_{{}^n\Delta t_c > 0} \{{}^n\Delta t_{c1}, {}^n\Delta t_{c2}\}$
${}^{n+1}\bar{\boldsymbol{v}}_i$
${}^{n+1}\boldsymbol{x}_i$
${}^{n+1}\bar{\boldsymbol{v}}_j$
${}^{n+1}\boldsymbol{x}_j$

Figure 2: Different settings for the collision configuration of particle i and j depending on the time step size ${}^n\Delta t_c$. Both configurations fulfill constraint (5). The growth of the particles is neglected. Left: ${}^n\Delta t_c$ is selected to be the minimum and positive, see (7). Right: ${}^n\Delta t_c$ is selected to be the maximum and positive. This leads to an inadmissible configuration and is not tolerated.

For the sum of the particle radii being equal to the length of the branch vector, particle i and j are in contact, see Fig. 1, right. Using (3),(2) and (5) leads up to two possible solutions for the time step ${}^n\Delta t_c$.

$$
{}^n\Delta t_{c_{1,2}} = \frac{\left[-\mathrm{v} \pm \sqrt{\mathrm{v}^2 - \mathrm{uw}}\right]}{\mathrm{u}} \tag{6}
$$

$$
\text{with} \quad \mathrm{u} = \Delta \boldsymbol{v}_{ij}{}^2 - \left[g_i + g_j\right]^2, \quad \mathrm{v} = {}^n\boldsymbol{l}_{ij} \bullet \Delta \boldsymbol{v}_{ij} - \left[{}^n r_i + {}^n r_j\right]\left[g_i + g_j\right],
$$
$$
\mathrm{w} = {}^n\boldsymbol{l}_{ij}{}^2 - \left[{}^n r_i + {}^n r_j\right]^2, \quad \Delta \boldsymbol{v}_{ij} = {}^n\boldsymbol{v}_j - {}^n\boldsymbol{v}_i
$$

We restrict our solution for ${}^n\Delta t_c$ to be greater than zero since only advancing in time is possible. Furthermore, we are interested in the minimum of the possible solutions.

$$
{}^n\Delta t_c = \min_{{}^n\Delta t_c > 0} \left\{{}^n\Delta t_{c_1}, {}^n\Delta t_{c_2}\right\} \tag{7}
$$

The minimum of the possible solutions, regarding the time step, is connected to the fulfillment of the contact condition between two particles prior overlap, while the maximum solution leads to the fulfillment of the contact condition posterior the particle overlap, see Fig. 2. Calculation of the time step ${}^n\Delta t_c$ enables the progress of the event driven scheme to the next event.

Proof. For two particles the upper time bound of a collision can be constructed from the equality between the sum of the particle diameters and the diagonal of the rve, having periodic boundary conditions:

$$
2\left[{}^{n+1}r_1 + {}^{n+1}r_2\right] = \sqrt{2}\,\mathsf{length}_{\mathsf{rve}}. \tag{8}
$$

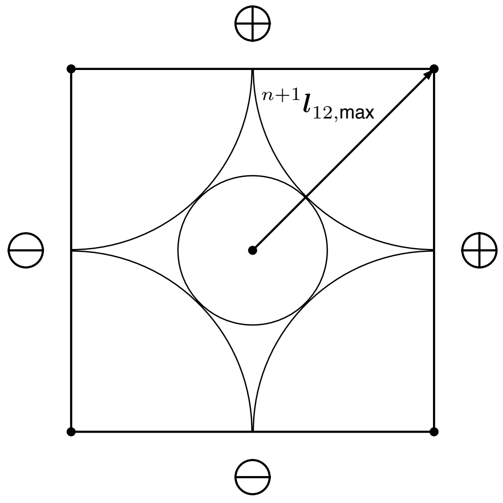


Figure 3: One possible configuration of two particles in a boundary box having periodic boundary conditions. The dimensions of the box are given by $\mathsf{length}_{\mathsf{rve}} \times \mathsf{length}_{\mathsf{rve}}$. Particle one is located at the center of the boundary box, while particle two and its images are positioned at the corners. The highest possible volume fraction ψ_{max} is reached. Moreover (11) is distinctly visible.

Using (2) leads to the maximum time step of

$$^{n}\Delta t_{c,\mathsf{max}} = \frac{\sqrt{2}\,\mathsf{length}_{\mathsf{rve}}}{2\left[{}^{n}r_1 + g_1 + {}^{n}r_2 + g_2\right]} \tag{9}$$

to reach the collision event. As can be seen, the maximum time step $^{n}\Delta t_{c,\mathsf{max}}$ to reach the next event does not depend on the particle velocities. This leads to the statement that under arbitrary particle velocities, the collision between the two particles will occur in less or equal time regarding the maximum time step. This is founded by the fact that the state reached by the maximum time step $^{n}\Delta t_{c,\mathsf{max}}$ is equivalent to the state possessing the highest possible volume fraction φ_{max}, restricting the total number of particles to be two.

$$\varphi_{\mathsf{max}} = \frac{\pi\left[\left[{}^{n}r_1 + g_1\,{}^{n}\Delta t_{c,\mathsf{max}}\right]^2 + \left[{}^{n}r_2 + g_2\,{}^{n}\Delta t_{c,\mathsf{max}}\right]^2\right]}{\mathsf{length}_{\mathsf{rve}}} \tag{10}$$

A larger number of particles will automatically lead into a smaller maximum time step $^{n}\Delta t_{c,\mathsf{max}}$. Additionally, it can be shown by (8) and (5) that the distance between the two particles in contact $\|^{n+1}\boldsymbol{l}_{12}\|$, assuming the highest reachable volume fraction φ_{max}, is constant for any arbitrary growth rates g_1 and g_2, illustrated in Fig. 3.

$$\|^{n+1}\boldsymbol{l}_{12}\| = \frac{\sqrt{2}\mathsf{length}_{\mathsf{rve}}}{2} \tag{11}$$

	$\overset{-}{{}^{n+1}v}_{\mathrm{n}_i} < \overset{-}{{}^{n+1}v}_{\mathrm{n}_j}$	$\overset{-}{{}^{n+1}v}_{\mathrm{n}_i} > \overset{-}{{}^{n+1}v}_{\mathrm{n}_j}$
$\overset{+}{{}^{n+1}v}_{\mathrm{n}_i}$	$\overset{-}{{}^{n+1}v}_{\mathrm{n}_i} - g_i$	$\overset{-}{{}^{n+1}v}_{\mathrm{n}_j} - g_i$
$\overset{+}{{}^{n+1}v}_{\mathrm{n}_j}$	$\overset{-}{{}^{n+1}v}_{\mathrm{n}_j} + g_j$	$\overset{-}{{}^{n+1}v}_{\mathrm{n}_i} + g_j$

Table 1: Posterior collision velocity increase and decrease regarding the enhanced momentum relation, depending on the prior collision velocity ratio.

For a number of n particles, collisions inside the periodic boundary box are guaranteed. This is due to the fact that the size of the rve boundary box does not change while the volume of the particles increase with time, trying to approach the maximum possible volume fraction. □

2.2 Event Handling

Being able to advance to the next event, the event itself has to be handled. For simplicity purposes, we select the mass m of particle i and j being equal to unity. The relation between the particle normal velocities directly before and right after the collision can be formulated by

$$\overset{+}{{}^{n+1}v}_{\mathrm{n}_i} = \overset{-}{{}^{n+1}v}_{\mathrm{n}_j}\,, \qquad \overset{+}{{}^{n+1}v}_{\mathrm{n}_j} = \overset{-}{{}^{n+1}v}_{\mathrm{n}_i}\,, \tag{12}$$

$$\text{with} \quad {}^{n+1}v_{\mathrm{n}} = {}^{n+1}\boldsymbol{v} \bullet {}^{n+1}\boldsymbol{n}_{ij}\,, \quad {}^{n+1}\boldsymbol{n}_{ij} = \frac{{}^{n+1}\boldsymbol{l}_{ij}}{\|{}^{n+1}\boldsymbol{l}_{ij}\|},$$

where $\overset{-}{(\bullet)}$ indicates quantities prior and $\overset{+}{(\bullet)}$ posterior to the collision. The assumption of the smoothness of the particles leaves the tangential particle velocities unchanged. Due to the previously introduced growth rates g_i it is not possible to use the well known simplified momentum relations given in (12). An enhancement, see Table 1, needs to be introduced to keep the particles from overlapping. This is done by increasing or decreasing the post collision normal velocities of the colliding particles by the growth rate g_i, see Table 1.

Using Table 1 to calculate the posterior collision velocities will finally lead to an energy blow up $\overset{+}{\mathrm{E}} > \overset{-}{\mathrm{E}}$. Since the construction of the rve does not underlie any physical meaning this effect can be neglected.

2.3 Algorithm

The here presented algorithm is based on the assumption of periodicity, laid out in Section 3. This presumption allows to simply focus on a periodic granular rve, having geometric periodic boundaries. The construction of such an rve starts with the definition of the periodic boundary box. For simplicity the rve is set to be a square with dimensions $\text{length}_{\text{rve}} \times \text{length}_{\text{rve}}$, however any reasonable shape is possible. The desired number of particles is randomly distributed inside the periodic boundary box, provided with random particle velocities. The initial radii of all particles are set to zero. Having the initial state set up, the event algorithm is started. Of interest is the next particle pair collision and its time. The time step calculation outlined in Subsection 2.1 is performed for each particle pair being able to collide. Different algorithms for fast collision detection can be found in the literature. We will assume the simplest possible collision search by comparing all particles, even though the contact search time is not attractive.

$$t_{\text{search time}} \propto \frac{\text{nop}\,[\text{nop} - 1]}{2} \tag{13}$$

The minimum time step of all possible collisions, calculated by (6), is selected to advance the event driven scheme. The post velocities of the colliding particle pair are further enhanced by using Table 1, followed by a new search for the next collision. Allowing the algorithm take its course, the particles float around inside the rve, collide and grow depending on the elapsed time. Postulation of a dropout criterion can be accomplished in many ways. We follow the simple approach of [BDL91] and select the time during collisions to be the variable of interest. With ${}^{n}\Delta t_c$ dropping under a certain threshold over a specified number of events fulfills our criterion. The complete algorithm is listed in Fig. 4.

```
INITIALIZATION
random positions ${}^{0}\boldsymbol{x}_i$, random velocities ${}^{0}\boldsymbol{v}_i$
radii ${}^{0}r_i = 0$, growth rates $g_i \in \mathbb{R}$
DO WHILE
    find collisions and collision times between all particles (5), (6)
    select minimum collision time of all possible collisions
    advance all particle positions regarding the minimum collision time (3)
    update velocity of colliding particle pair (Table 1)
```

Figure 4: Basic algorithm to produce dense particle packings by using a periodic boundary box. The dropout criterion, regarding the DO WHILE-loop, is based on the minimum collision time.

3 Concept of Periodicity

The assumption of periodicity inside the particulate medium leads to the use of a boundary box having periodic boundary conditions, see Fig. 5. Topologically, the periodic boundary box for two dimensional systems can be thought of as a torus. The torus is set up by connecting the opposite boundary box sides marked by $\ominus$ and $\oplus$. Assuming a torus, the particles can only move on its surface. A particle with its center being inside the boundary

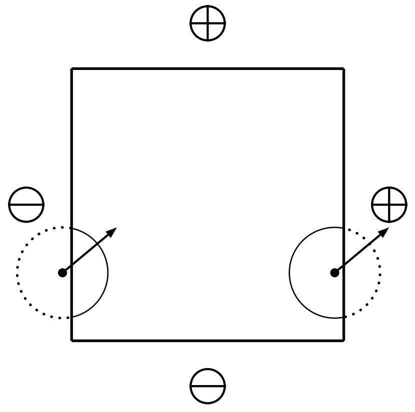


Figure 5: Periodic boundary box including one primary particle and its replica. Both particles share the same properties, excluding their center positions. The primary particle is found on the right side of the box, while the replicated particle is positioned on the left side. The distance between the particles is equal to the length of the boundary box. The replicated particle is entering the boundary box, while the primary particle is leaving.

box is considered to be a primary particle. If a primary particle intersects with a boundary of the periodic boundary box, a replica of this particle is positioned on the opposite side. All properties of the primary particle are projected onto the replicated particle. As soon as the center of the primary particle leaves the boundary box, the replicated particle center enters the boundary box and their states change. This leads to a constant number of primary particles inside the periodic boundary box. The periodic boundary box is used as a frame for the periodic rve.

4 Examples

A series of different stages of one rve during production is shown in Fig. 6. In the early stages of the evolution process, particles move freely from collision to collision. At this point no oscillation of the particles are noticed. At a later time the radii are increased and the space available for the particles to move is limited. This point is distinguished by a high oscillation of the particles. From this point on, only shifts of patterns or reorganization is possible. Due to the very close packing of the particles, times between collisions narrow down quickly. This results, due to (2), into a slow particle growth between the collisions. Close to the final state, only shifts inside the particle assembly enable a relocation of the particles. The quality of the final rve is in the present case dependent on a full boundary set $\mathscr{B}$.

$$\mathscr{B} := \left\{ i \in \{1, \ldots, \mathsf{nop}\} : \overline{B}_{R_i}\left({}^{0}\boldsymbol{x}_i\right) \cap \partial\mathsf{rve} \neq \emptyset \right\} \tag{14}$$

Set $\mathscr{B}$ is considered a full set if each gap between the particles of set $\mathscr{B}$ is less then the sum of the two particle radii observed and the smallest radius. Fulfilling the need of a full set $\mathscr{B}$ insures that under shear and compression no particles from the inside will escape the boundary box. Images of an evolution process can be seen in Fig. 6. The images shown in Fig. 7 are all based on different values of initial particle positions

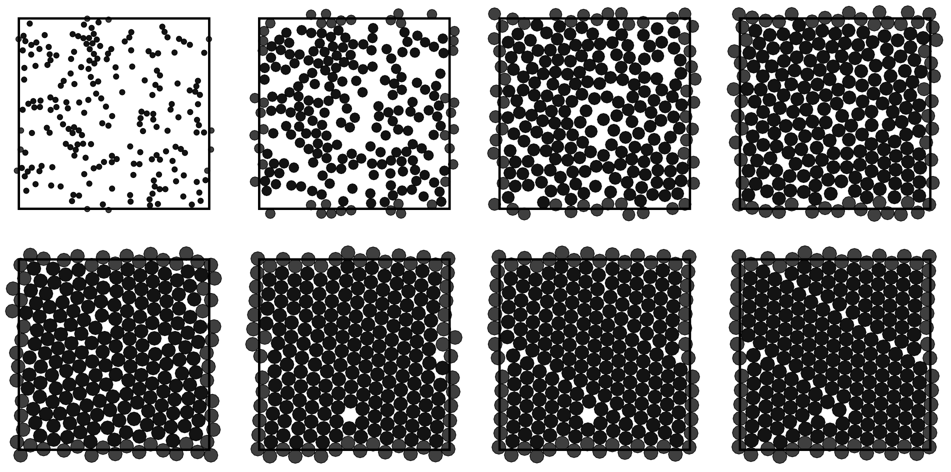


Figure 6: Evolution process of a monodisperse rve. Particles inside the rve are colored in blue, particles intersecting with the periodic boundary box are colored in red. The periodic boundary box is pictured by a black frame. The evolution process runs from the upper left image to the lower right image. The upper left image shows the rve after ten event cycles. The image at the lower right shows the final state of the rve. Areas of dense packing are clearly visible while at the same time irregularities can be observed.

and velocities. A fixed growth rate of 5 leads to the upper two images of Fig. 7. The highhandedness of the outcome is clearly visible. Even though the particle radii of the final state are nearly equivalent, patterns of highly packed particles are not comparable. Still, the way of how the rve is structured is equal in both images. Optimally packed particle clusters are separated from one another by edges. The edges are distinguished by the shifts and small turns of densely packed particle areas. The lower left images shows an rve constructed with different particle growth rates. For a snug fit of the particles and a high volume fraction of the rve, the growth rates were calculated by a power law. The power law is used to reconstruct a sieve curve known from natural granular materials. Using this growth rates renders a well packed and mixed polydisperse rve. The different particle sizes do not allow a clear statement regarding highly packed areas and their edges, found in monodisperse packings. A special kind of polydisperse packing is illustrated in the lower right image. Here, only two different growth rates are used. Similarities like densely packed particle clusters and their edges, known from the monodisperse packing, are of existence. Furthermore, this example shows the numerical stability of the here presented method. This leads to the conclusion that a meaningful growth rate relation does not effect the algorithm itself.

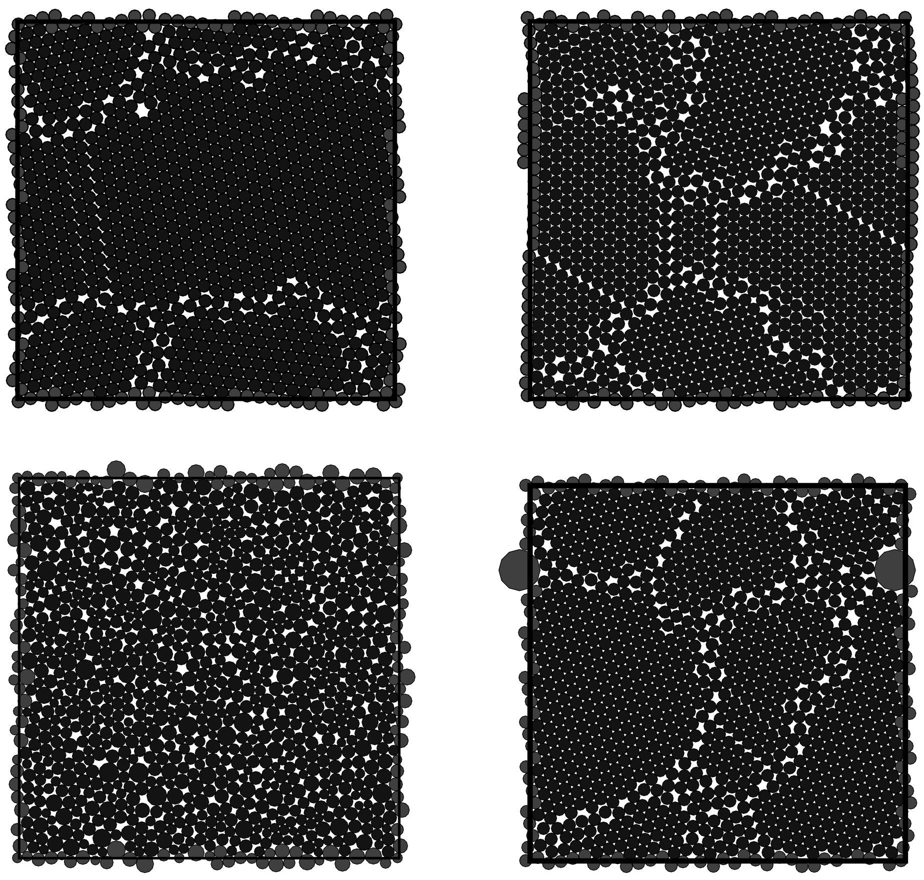


Figure 7: Four final state rves, started with 1000 particles having different initial positions as well as velocities. The two upper images show monodisperse particle packings. Highly packed patterns as well as pattern distortions are visible. The two lower images represent polydisperse particle packings. The lower right image shows a particle packing having growth rates related to the power law. Dense patterns are not clearly visible. The lower right image shows an almost monodisperse packing including one larger particle. Relations to the monodisperse packing exist.

5 Acknowledgment

The authors thank the German Research Foundation (DFG) for financial support within the DFG International Research Training Group 1131, ”Visualization of Large and Unstructured Data Sets Applications in Geospatial Planning, Modeling and Engineering” at the University of Kaiserslautern. Furthermore, we would like to acknowledge Prof. T. I. Zohdi of the University of California, Berkeley for his outstanding cooperation.

References

[ARK02] F. H. Stillinger A. R. Kansal, S. Torquato. Computer generation of dense polydisperse sphere packings. *J. Chem. Phys.*, 117(18):8212–8218, 2002.

[BDL91] E. N. Pinson B. D. Lubachevsky, F. H. Stillinger. Disks vs. Spheres: Contrasting Properties of Random Packings. *J. Stat. Phys.*, 64:501–524, 1991.

[CS78] P. A. Cundall and O. D. L. Strack. The Distinct Element Method as a Tool for Research in Granular Media. *Report to the National Science Foundation Concerning NSF Grant ENG76-20711, PART I*, 1978.

[CS79a] P. A. Cundall and O. D. L. Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29:47–65, 1979.

[CS79b] P. A. Cundall and O. D. L. Strack. The Distinct Element Method as a Tool for Research in Granular Media. *Report to the National Science Foundation Concerning NSF Grant ENG76-20711, PART II*, 1979.

[DM04] J. Dettmar and C. Miehe. A framework for micro-macro transitions in periodic particle aggregates of granular materials. *Comput. Methods Appl. Mech. Engrg.*, 192:225–256, 2004.

[Lub91] B. D. Lubachevsky. How to Simulate Billiards and Similar Systems. *J. Comput. Phys.*, 94:255–283, 1991.

[Zoh04] T. I. Zohdi. Modeling and direct simulation of near-field granular flows. *Int. J. Solid Struct.*, 42(2):539–564, 2004.

[Zoh05] T. I. Zohdi. Charge-induced clustering in multifield particulate flow. *Int. J. Num. Meth. Eng.*, 62(7):870–898, 2005.

[ZW05] T. I. Zohdi and P. Wriggers. *Introduction to computational micromechanics*. Springer-Verlag, 2005.

Visualization of Multidimensional Phase Space Portraits in Structural Dynamics

Patrick R. Schmitt and Paul Steinmann
University of Kaiserslautern
Chair of Applied Mechanics
Department of Mechanical and Process Engineering
D-67653 Kaiserslautern, Germany
prschmit@rhrk.uni-kl.de

Abstract: Within the Hamiltonian setting of classical mechanics the behavior of a dynamical system is typically visualized in terms of the canonical phase space coordinates, i.e. the position and the canonical momentum in a so called phase space portrait. The investigation of elastodynamic systems modelled by finite element simulations leads to high dimensionality of phase spaces, i.e. twice the number of nodal degrees of freedom (DOFs). In order to study the overall phase space structure a reduction of the problem's dimensionality seems absolutely necessary which leads us to the theory of pseudo-rigid bodies. A pseudo-rigid body is a continuum capable of undergoing only spatially homogeneous deformation, which enables us to interpret its configuration manifold as a Lie group. In a Hamiltonian setting the phase space is just the cotangent bundle of the configuration manifold, which has a natural symplectic structure. We will give an overview of a class of geometric numerical integration schemes which should be well suited to conserve the problem's inherent structure.

1 Introduction

In order to study the behaviour of dynamical systems one usually makes use of the theory of Hamiltonian systems and the phase-space concept. Within Hamilton's framework a dynamical system with n degrees of freedom (DOFs) is described by generalized coordinates $q_k \in \mathbb{R}$ and corresponding canonical momenta $p_k \in \mathbb{R}$ for $k = 1, \ldots, n$. The scalar Hamiltonian $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$ is sufficient to describe the dynamics of the system completely. The equations of motion are Hamilton's equations:

$$\dot{p}_k = -\frac{\partial H(q,p,t)}{\partial q_k}, \qquad \dot{q}_k = \frac{\partial H(q,p,t)}{\partial p_k}, \quad k = 1, \ldots, n$$

Solutions to this system of $2n$ ordinary differential equations are curves $(q(t), p(t)) \in \mathbb{R}^{2n}$ in *phase-space*.

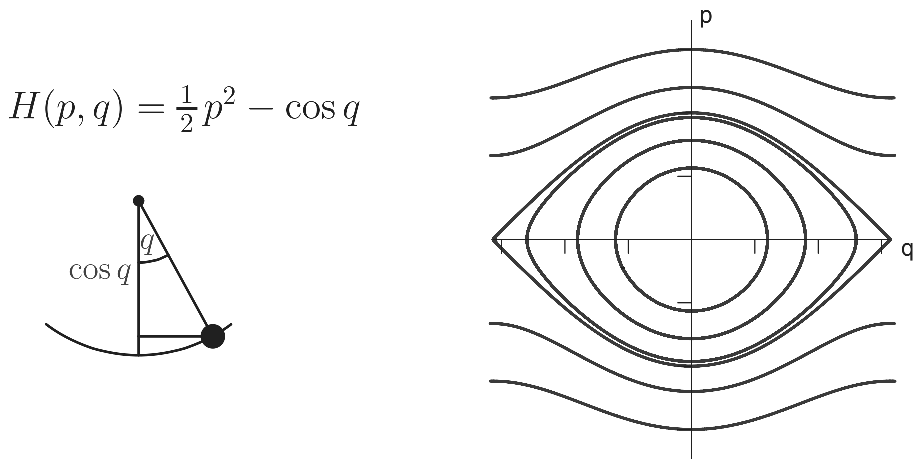


Figure 1: 2-dimensional mathematical pendulum in ”natural” units, gravity acting downwards (left) and its phase-space portrait (right)

To illustrate the phase-space concept figure 1 depicts one of the most simple dynamical systems, i.e. the 2-dimensional mathematical pendulum. Solution curves are clearly divided by a separatrix into two regions of vastly different behaviour. In the neighorhood of the origin of the coordinate system the pendulum is in normal back-and-forth oscillatory motion (closed, ellipsoidal solution curves). The outer region describes clockwise and counterclockwise spinning motion.

The dynamics of engineering structures are usually modelled by finite element simulations which leads to high dimensionality of phase spaces, i.e. twice the number of nodal degrees of freedom. Even small academic examples like the rotor blade shown in the upper left quadrant of figure 2 have a huge number of DOFs, in this case 716. The time-evolution of this freely flying rotor blade was determined by solving Hamilton’s equations with an enhanced Galerkin method [Gro04]. The three quadrants labeled ”phase-space projection” show different views of the $\{q_1, p_1, p_2\}$-projection for an arbitrarily chosen node. Obviously projection to an only 3-dimensional subspace of a vastly higher dimensional phase-space can not yield insight into the overall phase space structure.

2 Pseudo-rigid Bodies

As we have seen in the last section it is absolutely necessary to reduce the dimensonality of the considered system if insight into phase-space structures is to be gained by means of visualization. Considering the theory of elasto-dynamics, the obvious simplification of the full problem is the restriction of possible modes of deformation. A suitable and well studied [CM88, SP99, KP04, Pap01, Cas04] choice for this restriction is the so called pseudo-rigid body. On the one hand it generalizes a rigid body by admitting deformations

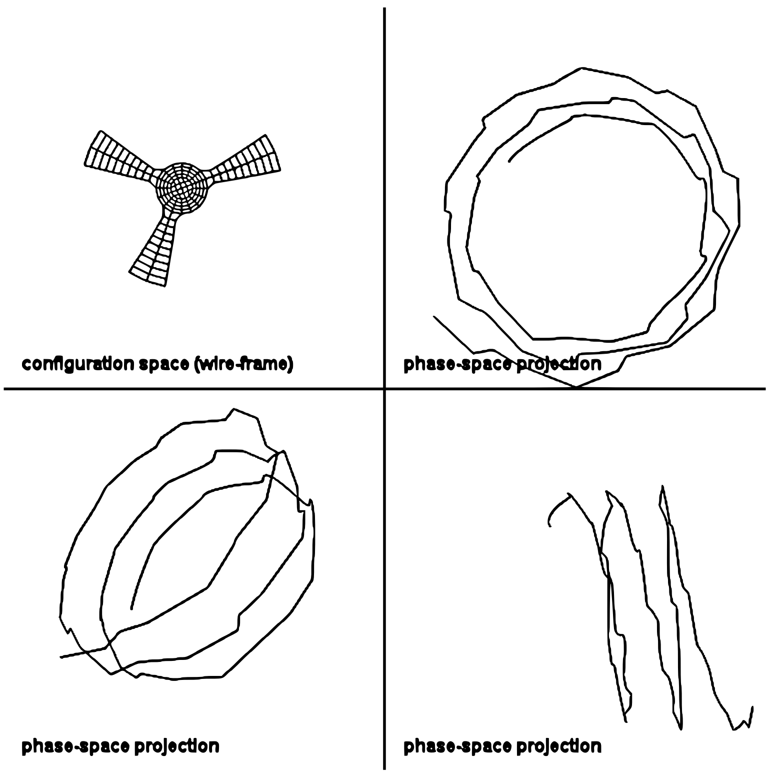


Figure 2: freely flying 2-dimensional hyper-elastic rotor blade, 140 4-node elements $\rightarrow$ 716 DOFs, different views of the $\{q_1, p_1, p_2\}$-projection for a single node

of the body via affine transformations, on the other hand it can be viewed as a restriction of an elastic body.

We will start by describing the pseudo-rigid body in a standard continuum mechanical setting. Let $\mathcal{B}_0 \subset \mathbb{R}^3$ denote the reference configuration of a pseudo-rigid body with smooth boundary $\partial\mathcal{B}_0$ and reference mass density $\varrho_0 : \mathcal{B}_0 \to \mathbb{R}^+$. The non-linear deformation map $\boldsymbol{\varphi} : \mathcal{B}_0 \to \mathcal{B}_t$ describes the motion of the pseudo-rigid body. A Taylor-expansion of the non-linear deformation map

$$\boldsymbol{\varphi}(\boldsymbol{X}, t) = \bar{\boldsymbol{x}}(t) + \boldsymbol{F}^{(0)}(t)\left(\boldsymbol{X} - \bar{\boldsymbol{X}}\right) + \frac{1}{2}\boldsymbol{F}^{(1)}(t) : \left[\left(\boldsymbol{X} - \bar{\boldsymbol{X}}\right) \otimes \left(\boldsymbol{X} - \bar{\boldsymbol{X}}\right)\right] + \dots$$

can be used to illustrate the restriction on the full elasto-dynamic case imposed by pseudo-rigidity:

$$\text{pseudo-rigid} \Leftrightarrow \boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{X}, t) = \bar{\boldsymbol{x}}(t) + \boldsymbol{F}(t)\left(\boldsymbol{X} - \bar{\boldsymbol{X}}\right). \tag{1}$$

This mapping is affine, i.e. $\boldsymbol{F}$ is a second order tensor with $\det \boldsymbol{F} > 0$, and allows us to state the equations of motion of a pseudo-rigid body as follows.

Starting from the *balance of linear momentum* and the definition of centre of mass $\bar{\boldsymbol{X}}$

$$\operatorname{Div}\boldsymbol{P} + \varrho_0\boldsymbol{b} = \varrho_0\ddot{\boldsymbol{x}}, \quad m\bar{\boldsymbol{X}} = \int_{\mathcal{B}_0} \varrho_0\boldsymbol{X}\,dV \tag{2}$$

we obtain the equations of motion for independent variables $\bar{\boldsymbol{x}}$ and $\boldsymbol{F}$ by integrating eqn. $(2)_1$ and the tensor product of eqn. $(2)_1$ with $\left(\boldsymbol{X} - \bar{\boldsymbol{X}}\right)$ over $\mathcal{B}_0$:

$$\begin{aligned} m\ddot{\bar{\boldsymbol{x}}} &= \int_{\mathcal{B}_0} \varrho_0\boldsymbol{b}\,dV + \int_{\partial\mathcal{B}_0} \boldsymbol{P}\cdot\boldsymbol{N}\,dA & (3)\\ V_0\bar{\boldsymbol{P}} + \ddot{\boldsymbol{F}}\cdot\boldsymbol{E} &= \int_{\mathcal{B}_0} \varrho_0\boldsymbol{b}\otimes\left(\boldsymbol{X}-\bar{\boldsymbol{X}}\right)dV + \int_{\partial\mathcal{B}_0} \left(\boldsymbol{P}\cdot\boldsymbol{N}\right)\otimes\left(\boldsymbol{X}-\bar{\boldsymbol{X}}\right)dA, & (4)\end{aligned}$$

where $\boldsymbol{E}$ is the Euler tensor and $\bar{\boldsymbol{P}}$ is the mean first Piola-Kirchhoff stress defined by

$$\boldsymbol{E} = \int_{\mathcal{B}_0} \varrho_0\,\left(\boldsymbol{X}-\bar{\boldsymbol{X}}\right)\otimes\left(\boldsymbol{X}-\bar{\boldsymbol{X}}\right)\,dV, \quad \bar{\boldsymbol{P}} = \frac{1}{V_0}\int_{\mathcal{B}_0} \boldsymbol{P}\,dV,$$

with $m = \int_{\mathcal{B}_0} \varrho_0\,dV$ and $V_0 = \int_{\mathcal{B}_0} dV$.

Following the ideas of [SK03, Sou02] we formalize the pseudo-rigid body to study it in a general Hamiltonian setting. The configuration manifold Q for our system is the semi-direct product of the group of non-degenerate 3×3 matrices with positive determinant with $\mathbb{R}^3$

$$Q = GL^+(3) \ltimes \mathbb{R}^3.$$

Therefore the configuration manifold becomes a Lie-group with elements $(\boldsymbol{A},\boldsymbol{x}),\ \boldsymbol{A}\in GL^+(3), \boldsymbol{x}\in\mathbb{R}^3$ and multiplication law

$$(\boldsymbol{B},\boldsymbol{y})\circ(\boldsymbol{A},\boldsymbol{x}) = (\boldsymbol{BA},\boldsymbol{Bx}+\boldsymbol{y}).$$

The phase-space of this general Hamiltonian system is its cotangent bundle

$$T^*Q = T^*[GL^+(3)\ltimes\mathbb{R}^3] \cong [GL^+(3)\ltimes\mathbb{R}^3]\times[\mathfrak{gl}(3)^*\ltimes\mathbb{R}^3],$$

which is itself also a Lie-group with a natural symplectic structure that should be preserved by numerical integration schemes. The kinetic energy of the pseudo-rigid body can be described by a function $K : T^*Q \to \mathbb{R}$ as

$$K = \frac{1}{2}\int_{\mathcal{B}_t} \varrho_t\dot{\boldsymbol{x}}\cdot\dot{\boldsymbol{x}}\,dv = \frac{1}{2}\left[\operatorname{tr}\left(\dot{\boldsymbol{F}}\cdot\boldsymbol{E}\cdot\dot{\boldsymbol{F}}\right) + m\dot{\bar{\boldsymbol{x}}}\cdot\dot{\bar{\boldsymbol{x}}}\right]$$

using eqn. (1) where mixed terms cancel due to eqn. $(2)_2$.

3 Geometric Integration (Lie-group Methods)

In order to exploit the problem's geometric sructure to the fullest extent, we have to consider integration schemes that generate solutions which remain in the cotangent bundle of the configuration manifold T^*Q. In recent years a powerful machinery of geometric integration (especially Lie-group integrators) has been developed, see for example [IMNZ00, HLW02]. We will present its basic ingredients in the following subsections.

3.1 Lie-groups and Lie-algebras

We will start by looking at Lie-groups. A Lie-group is a group with a smooth (differentiable) group operation and smooth inversion. Therefore it can also be viewed as a differentiable manifold. Several well-known examples are listed below:

$$\begin{aligned} GL(n,\mathbb{R}) &= \{g \in Mat(n \times n, \mathbb{R}) | \det g \neq 0\} \\ SL(n,\mathbb{R}) &= \{g \in GL(n,\mathbb{R}) | \det g = 1\} \\ O(n,\mathbb{R}) &= \left\{g \in GL(n,\mathbb{R}) | g^T g = e\right\} \\ SO(n,\mathbb{R}) &= \{g \in O(n,\mathbb{R}) | \det g = 1\} \end{aligned}$$

The Lie-algebra $\mathfrak{g}$ corresponding to a Lie-group G can be defined as the tangent space to the group at the identity element $e \in G$

$$\mathfrak{g} = T_e G.$$

$\mathfrak{g}$ is a vector space equipped with a skew-symmetric, bilinear product called Lie-bracket

$$[\,.\,,\,.\,] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$$

which satisfies Jacobi's identity: let $a, b, c \in \mathfrak{g}$

$$[a,[b,c]] + [b,[c,a]] + [c,[a,b]] = 0.$$

Because of this tangent space construction there are several choices of coordinate charts that map from the Lie-algebra back to the Lie-group. One such coordinate chart is the exponential map $\exp : \mathfrak{g} \to G$ which maps the whole Lie-algebra to a neighbourhood of $e \in G$. Because of the smooth group operation required above, this map can be extended to the whole group G via left- or right-translation. For matrix groups the exponential map is just the standard matrix exponential.

3.2 Lie-group and Lie-algebra Actions

Lie-group and Lie-algebra actions describe the effect of group and algebra elements on points of a manifold, think of rotations $R \in SO(3)$ acting on points of $\mathbb{R}^3$ for example. A (left) *Lie-group action* is a map

$$\Lambda : G \times \mathcal{M} \to \mathcal{M}.$$

It induces a corresponding *Lie-algebra action*

$$\lambda : \mathfrak{g} \times \mathcal{M} \to \mathcal{M}$$

via

$$\lambda(v,p) = \Lambda(\exp(v), p) \qquad v \in \mathfrak{g},\ p \in \mathcal{M}.$$

Next we define a map $\lambda_* : \mathfrak{g} \to \mathcal{X}(\mathcal{M})$ from the Lie-algebra to the set of vector-fields on $\mathcal{M}$ pointwise as

$$(\lambda_* v)(p) = \left.\frac{d}{dt}\right|_{t=0} \lambda(tv, p)$$

A differential equation on the manifold can now be written as

$$y' = \mathcal{F}_\lambda(t, y) = (\lambda_* f(t, y))\,(y), \qquad y(0) = p \in \mathcal{M} \tag{5}$$

where $f : \mathbb{R} \times \mathcal{M} \to \mathfrak{g}$.

These definitions allow us to state the main paradigm of Lie-group methods as follows: The solution of the differential equation (5) for small $t \in \mathbb{R}^+$ is

$$y(t) = \lambda(u(t), p)$$

where $u(t) \in \mathfrak{g}$ satisfies

$$u' = \tilde{f}(u) = d\exp_u^{-1}\left(f(t, \lambda(u, p))\right), \qquad u(0) = 0 \in \mathfrak{g}. \tag{6}$$

This statement follows by the commutativity of the following diagram

$$\begin{tikzcd}
T\mathfrak{g} \arrow[r, "\lambda_p'"] & T\mathcal{M} \\
\mathfrak{g} \arrow[u, "\tilde{f}"] \arrow[r, "\lambda_p"'] & \mathcal{M} \arrow[u, "\mathcal{F}"']
\end{tikzcd}$$

One problem we have not yet considered is the non-commutativity of general Lie-groups and Lie-algebras. Let $u_1, u_2 \in \mathfrak{g}$, then

$$\begin{aligned}
\Lambda(\exp(u_1), \Lambda(\exp(u_2), p)) &= \Lambda(\exp(u_1) \cdot \exp(u_2), p) \\
&= \Lambda(\exp(\mathcal{B}(u_1, u_2), p) \\
\Rightarrow \qquad \lambda(u_1, \lambda(u_2, p)) &= \lambda(\mathcal{B}(u_1, u_2), p)
\end{aligned}$$

The symbol $\mathcal{B}(.,.)$ is the Baker-Campbell-Hausdorff formula which has the following explicit form in terms of iterated commutators

$$\begin{aligned}
\mathcal{B}(u_1, u_2) &= u_1 + u_2 + \frac{1}{2}[u_1, u_2] \\
&\quad + \frac{1}{12}[u_1, [u_1, u_2]] - \frac{1}{12}[u_2, [u_1, u_2]] + \ldots .
\end{aligned}$$

In order to solve differential equations on manifolds it is sufficient to solve corresponding equations in the Lie-algebra which is a linear space. Therefore standard integration schemes can be applied on the algebra-level and the solution mapped back to the nonlinear manifold via appropriate coordinate mappings. Suitable combinations of these have been developed in recent years, foremost the *Runge-Kutta-Munthe-Kaas* methods which usually require many iterated commutator evaluations and *Crouch-Grossman* methods which evaluate vector field flows via matrix exponentials.

4 Conclusions

In this contribution we have laid out the problems in understanding phase-space structures of engineering structures caused by the high dimensionality of their respective phase-spaces. In order to reduce this high dimensionality a simpler model namely the pseudo-rigid body has been presented. The theory of pseudo-rigid bodies has then been cast into a Lie-algebraic framework which makes its dynamics approachable by new geometric numerical integration schemes.

Acknowledgements:
Financial support by the DFG within the IRTG 1131 is gratefully acknowledged.

References

[Cas04] J. Casey. Pseudo-Rigid Continua: Basic Theory and a Geometrical Derivation of Lagrange's Equations. *Proc. Roy. Soc. London A*, 460:2021–2049, 2004.

[CM88] H. Cohen and R. G. Muncaster. *The Theory of Pseudo-Rigid Bodies*. Springer-Verlag, New York, 1988.

[Gro04] M. Groß. *Conserving Time Integrators for Nonlinear Elastodynamics*. Dissertation, Lehrstuhl für Technische Mechanik, Technische Universität Kaiserlautern, 03 2004.

[HLW02] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31 of *Springer series in computational mathematics*. Springer-Verlag, New York, 2002.

[IMNZ00] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna. Lie-group methods. *Acta Numerica*, 9:215–365, 2000.

[KP04] E. Kanso and P. Papadopoulos. Dynamics of Pseudo-rigid Ball Impact on Rigid Foundation. *Int. J. Nonl. Mech.*, 39:299–309, 2004.

[Pap01] P. Papadopoulos. On a Class of Higher-order Pseudo-rigid Bodies. *Math. Mech. Sol.*, 6:631–640, 2001.

[SK03] J. J. Sławianowski and V. Kovalchuk. Invariant Geodetic Problems on the Affine Group and Related Hamiltonian Systems. *Rep. Math. Phys.*, 51(2/3):371–379, 2003.

[Sou02] M. E. Sousa Dias. Pseudo-rigid bodies: A geometric Lagrangian approach. *Acta Appl. Math.*, 70:209–230, 2002.

[SP99] J.M. Solberg and P. Papadopoulos. A Simple Finite Element-based Framework for the Analysis of Elastic Pseudo-rigid Bodies. *Int. J. Numer. Meth. Engng.*, 45:1297–1314, 1999.

Galerkin-Based Time Integrators for Geometrically Nonlinear Elasto-Plastodynamics – Challenges in Modeling & Visualization

Rouven Mohr, Andreas Menzel, and Paul Steinmann
University of Kaiserslautern
Chair of Applied Mechanics
Department of Mechanical and Process Engineering
D-67653 Kaiserslautern, Germany
rmohr@rhrk.uni-kl.de

Abstract: Computational modeling often demands the incorporation of dynamic effects. The consideration of computational continuum dynamics requires in particular advanced numerical techniques to satisfy the classical balance laws like balance of linear and angular momentum or the laws of thermodynamics. Energy and momentum conserving time integrators for purely elastic behaviour are well-established in computational dynamics [2, 3, 4, 11, 12, 13]. The involvement of dissipation effects in the dynamic modeling poses further challenges. There are only a handful approaches that deal with dynamic plasticity or dissipative systems and the related conservation properties [1, 5, 7, 8, 9]. Most of them are based on specific order-restricted time integration schemes. Some of the conserving schemes were constructed on the basis of a particulary designed algorithmic stress tensor, see [3, 4].
In this contribution we deal with the enhancement of these concepts to finite plasticity based on a general fully discrete Finite Element Galerkin method combined with an 'elastic-enhanced algorithmic' stress tensor. First, we apply concepts of geometrically nonlinear continuum mechanics and Finite Elements in space to receive a semidiscrete system of equations of motion based on a Hamiltonian-type formalism. The formulation of finite plasticity adopts a multiplicative decomposition of the deformation gradient in an elastic and a plastic part, see [6, 10]. For the temporal approximation of the semidiscrete system also Finite Elements are used, compare [2, 4].
The resulting mechanically and thermodynamically consistent time-stepping scheme for geometrically nonlinear elasto-plastodynamics is analysed regarding its conservation properties influenced by the applied nonstandard quadrature rule. It will be shown that these algorithmic conservation properties require for nonstandard visualization methods to further insight on the performance of the resulting time integrator.

1 Constitutive Formulation

We start with some basic concepts of geometrically nonlinear continuum mechanics. First, the nonlinear deformation map $\boldsymbol{\varphi}(\boldsymbol{X}, t) : \mathcal{B}_0 \times [0, T] \mapsto \mathcal{B}_t$ shall be introduced as a mapping from the reference to the spatial configuration. In the context of finite plasticity, the resulting deformation gradient $\boldsymbol{F} := \nabla_{\boldsymbol{X}}\varphi(\boldsymbol{X},t)$ is assumed to be multiplicative decomposed into an elastic and a plastic part, see [6]:

$$\boldsymbol{F} \doteq \boldsymbol{F}_e \cdot \boldsymbol{F}_p \tag{1}$$

This multiplicative decomposition naturally implies three configurations. In contrast to the modeling of elasticity, additional internal variables $\boldsymbol{q}^{int}$ have to been included in the Helmholtz energy density Ψ_0 for the plastic case to model the loading history. These are the plastic part of the deformation gradient $\boldsymbol{F}_p$ and other variables to incorporate e.g. hardening or damage effects. By assuming an additive structure of $\Psi_0(\boldsymbol{F},\boldsymbol{q}^{int}) = \Psi_0^{mac}(\boldsymbol{F},\boldsymbol{q}^{int}) + \Psi_0^{mic}(\boldsymbol{q}^{int})$ with the macroscopic Ψ_0^{mac} and the microscopic part Ψ_0^{mic} and by restriction on isotropy, the Helmholtz energy density can be written in eigenvalues of the elastic right Cauchy-Green strain tensor $\widehat{\boldsymbol{C}_e} := \boldsymbol{F}_e^t \cdot \boldsymbol{F}_e$ of the intermediate configuration $\widehat{\mathcal{B}}$:

$$\Psi_0 = \Psi_0^{mac}({}^1\lambda_{\widehat{\boldsymbol{C}}_e}, {}^2\lambda_{\widehat{\boldsymbol{C}}_e}, {}^3\lambda_{\widehat{\boldsymbol{C}}_e}) + \Psi_0^{mic}(\boldsymbol{q}^{int}) \tag{2}$$

For thermodynamical aspects it is accepted to introduce the so-called conjugated thermodynamical forces $\boldsymbol{\alpha}^{int} := -\nabla_{\boldsymbol{q}^{int}}\Psi_0$. To differ between elastic and plastic deformation states the yield function Φ is introduced which defines the elastic range $\mathbb{E}_{\boldsymbol{\alpha}^{int}} := \{\boldsymbol{\alpha}^{int} \,|\, \Phi(\boldsymbol{\alpha}^{int};\boldsymbol{q}^{int}) < 0\}$, see [10]. For thermodynamically consistent modeling the dissipation inequality – corresponding to the second law of thermodynamics – should be fulfiled, namely:

$$\mathcal{D} = \left\langle \boldsymbol{\alpha}^{int}, \mathrm{D}_t\boldsymbol{q}^{int} \right\rangle \geq 0 \tag{3}$$

By accounting additionally the postulate of maximum dissipation, the resulting local evolution of the internal variables – well-known as associated evolution equations, see e.g. [10] – can be written as:

$$\mathrm{D}_t\boldsymbol{q}^{int} = \mathrm{D}_t\lambda \left[\nabla_{\boldsymbol{\alpha}^{int}}\Phi(\boldsymbol{\alpha}^{int};\boldsymbol{q}^{int})\right] \tag{4}$$

The plastic multiplier $\mathrm{D}_t\lambda$ and the yield function Φ must satisfy the consistency and the Kuhn-Tucker conditions, compare [10].

2 Semidiscrete Dynamics

We begin with a standard Finite Element discretisation in space for the reference configuration of a solid continuum body: $\mathcal{B}_0 = \bigcup_{e=1}^{n_{el}} \mathcal{B}_0^e$. With the spatial approximations of the nonlinear deformation map $\boldsymbol{\varphi} : \mathcal{B}_0 \times [0,T] \mapsto \mathbb{R}^{n_{dim}}$ the semidiscrete map can be written by means of the spatial shape functions $N_A(\boldsymbol{X})$ in the form: $\boldsymbol{\varphi}(\boldsymbol{X},t) = \sum_{A=1}^{n_{node}} \boldsymbol{q}_A(t)\, N_A(\boldsymbol{X})$. Consequently the spatial approximations of the spatial velocity $\mathrm{D}_t\boldsymbol{\varphi} = \sum_{A=1}^{n_{node}} \mathrm{D}_t\boldsymbol{q}_A\, N_A$, the deformation gradient $\boldsymbol{F} = \sum_{A=1}^{n_{node}} \boldsymbol{q}_A \otimes \nabla N_A$ and the right Cauchy-Green strain tensor $\boldsymbol{C} := \boldsymbol{F}^t \cdot \boldsymbol{F} = \sum_{A,B=1}^{n_{node}} \boldsymbol{q}_A \cdot \boldsymbol{q}_B\, \nabla N_A \otimes \nabla N_B$ of the reference configuration can be computed straightforwardly. To obtain a semidiscrete system of equations of motion a Hamiltonian-type formalism with the generalised coordinates $\boldsymbol{q} = [\boldsymbol{q}_1, ..., \boldsymbol{q}_{n_{node}}]^t$ and the generalised momenta $\boldsymbol{p} = [\boldsymbol{p}_1, ..., \boldsymbol{p}_{n_{node}}]^t$ – the phase-space variables $\boldsymbol{z} := [\boldsymbol{q}, \boldsymbol{p}]^t$ – is applied. The Hamiltonian is defined as the sum of the kinetic energy $T(\boldsymbol{p}) = \frac{1}{2}\, \boldsymbol{p} \cdot \mathbb{M}^{-1} \cdot \boldsymbol{p}$, the internal energy $V^{int} = \int_{\mathcal{B}_0} \Psi_0 \, \mathrm{d}V$ and possibly an external potential V^{ext}, see e.g. [2, 4]:

$$H(\boldsymbol{q}, \boldsymbol{p}) = T(\boldsymbol{p}) + V^{int}(\boldsymbol{q}) + V^{ext} \tag{5}$$

With these definitions and the symplectic matrix $\mathbb{J}$ the resulting canonical equations can be written in the compact form:

$$\mathrm{D}_t \boldsymbol{z}(t) = \mathbb{J} \cdot \nabla H(\boldsymbol{z}(t)) \tag{6}$$

More precisely the gradient of the Hamiltonian ∇H can be formulated in the following vector representation:

$$\nabla H = \begin{bmatrix} \boldsymbol{F}_{int} - \boldsymbol{F}_{ext} \\ \mathbb{M}^{-1} \cdot \boldsymbol{p} \end{bmatrix} \tag{7}$$

At this stage we have used the definiton of the internal load vector $\boldsymbol{F}_{int}$ which is given as a function of the second Piola-Kirchhoff stress tensor $\boldsymbol{S} = 2\nabla_{\boldsymbol{C}} \Psi_0$:

$$\boldsymbol{F}_{int}(\boldsymbol{S}) = \nabla_{\boldsymbol{q}} \int_{\mathcal{B}_0} \Psi_0 \, \mathrm{d}V \qquad \text{on nodal level} \qquad \boldsymbol{F}^{A}_{int}(\boldsymbol{S}) = \nabla_{\boldsymbol{q}_A} \int_{\mathcal{B}_0} \Psi_0 \, \mathrm{d}V \tag{8}$$

It will be demonstrated in the next section that especially this internal load vector $\boldsymbol{F}^{int}$ plays a decisive role for the temporal discretisation and in consequence for the resulting conservation properties of the time-stepping scheme.

3 Temporal Discretisation

In the following, a decomposition of the time interval of interest $[0, T] = \bigcup_{n=0}^{N} [t_n, t_{n+1}]$, a map of each time interval to the reference time interval $[0, 1]$ via the function $\alpha(t) = \frac{t - t_n}{h_n}$ and the time-step size $h_n = t_{n+1} - t_n$ are used. For the temporal approximation again a Finite Element method – more specific a continuous Galerkin method (short: a cG(k)-method) – shall be applied. Therefore, the temporal approximations for the unknown function $\boldsymbol{z}^h = \sum_{J=1}^{k+1} M_J(\alpha)\, \boldsymbol{z}_J$ and the test function $\delta \boldsymbol{z}^h = \sum_{I=1}^{k} \widetilde{M}_I(\alpha)\, \delta \boldsymbol{z}_I$ are introduced[1]. The resulting weak form in time can be written in the compact representation, compare [2, 4]:

$$\int_0^1 \left[\mathbb{J} \cdot \delta \boldsymbol{z}^h\right] \cdot \left[\mathrm{D}_\alpha \boldsymbol{z}^h - h_n \, \mathbb{J} \cdot \nabla H(\boldsymbol{z}^h)\right] \mathrm{d}\alpha = 0 \qquad \forall \;\; \delta \boldsymbol{z}^h \in \mathcal{P}^{k-1} \tag{9}$$

[1]It is important to underline that the temporal shape functions $M_I \in \mathcal{P}^k$ are polynomials of degree k and the shape functions $\widetilde{M}_I \in \mathcal{P}^{k-1}$ are only of degree $k - 1$.

In (9) a time-integrated internal load vector can be found which shall be refered to $\bar{\boldsymbol{F}}_{int}$. As mentioned before, the crucial aspect for the conservation properties of the resulting time-stepping scheme is the approximation of this time integral. To guarantee the conservation of the total energy in the elastic case, the following nonstandard quadrature rule with the Gauss points ζ_l and the Gauss weights w_l is used:

$$\bar{\boldsymbol{F}}_{int}^{A} \approx \sum_{l=1}^{n_{gp_t}} \sum_{B=1}^{node} w_l \, \widetilde{M}_I(\zeta_l) \, \boldsymbol{q}_B^h(\zeta_l) \left[\int_{\mathcal{B}_0} \nabla N_A \otimes \nabla N_B : {}^{el}\boldsymbol{S}^{alg} \, \mathrm{d}V \right] \Bigg|_{\zeta_l}$$

In this approximation the *elastic-enhanced algorithmic* stress tensor ${}^{el}\boldsymbol{S}^{alg}$ has been applied based on the enhanced Galerkin method – or short: eG(k)-method, see [4]:

$$ {}^{el}\boldsymbol{S}^{alg}(\boldsymbol{S}) = \boldsymbol{S} + 2 \left[\frac{\Psi_0^{\alpha=1} - \Psi_0^{\alpha=0} - \int_0^1 \boldsymbol{S} : \frac{1}{2} \mathrm{D}_\alpha \boldsymbol{C}^h \; \mathrm{d}\alpha}{\int_0^1 || \, \mathrm{D}_\alpha \boldsymbol{C}^h ||^2 \; \mathrm{d}\alpha} \right] \mathrm{D}_\alpha \boldsymbol{C}^h \tag{10}$$

With the above discretisations at hand we obtain a completely discrete system which offers the following conservation properties. By vanishing external loads the presented time-stepping scheme allows the conservation of both momentum maps: the conservation of linear momentum $\boldsymbol{I}$ as well as the conservation of angular momentum $\boldsymbol{L}$:

$$\boldsymbol{I}_{\alpha=1} - \boldsymbol{I}_{\alpha=0} = \boldsymbol{0} \qquad\qquad \boldsymbol{L}_{\alpha=1} - \boldsymbol{L}_{\alpha=0} = \boldsymbol{0} \tag{11}$$

By means of the elastic-enhanced algorithmic stress tensor ${}^{el}\boldsymbol{S}^{alg}$ this time integrator offers also a conserved total energy in the elastic case without external loading:

$$H_{\alpha=1} - H_{\alpha=0} = 0 \qquad\qquad D^{int} = 0 \tag{12}$$

And additionally the dissipation inequality (3) is fulfiled in the plastic case:

$$H_{\alpha=1} - H_{\alpha=0} < 0 \qquad\qquad D^{int} > 0 \tag{13}$$

Recapitulating, it can be retained that the presented time-stepping scheme enables the mechanically and thermodynamically consistent modeling of geometrically nonlinear elasto-plastodynamics.

4 Numerical Examples

Now the performance of the proposed time integrator shall be demonstrated with two representative numerical examples. In this context we investigate not only the elastic but also the plastic case to evaluate the featured algorithmic conservation properties. For all computations the elastic-enhanced algorithmic stress tensor (10) in combination with linear Finite Elements in time – a cG(1)-method – have been applied. The constitutive law relies on a Helmholtz energy density Ψ_0 formulated in principal stretches and a ‘v. Mises’-type yield function Φ.

4.1 Elastic Motion

The first example shows the elastic motion of a quasi-rigid ‘Flying-L’. For the spatial discretisation 36 4-node Finite Elements have been used, whereby the following parameters have been utilised: time-step size $h_n = 0.2$, elastic constants $\lambda = 100000$, $\mu = 50000$ and yield stress $Y_0 = 4000$. To start the motion we have applied an initial velocity $||\boldsymbol{v}_0|| = 5$ and a piecewise linear function with a loading period of 3 for the external loads which is shown in Figure 1 (a).

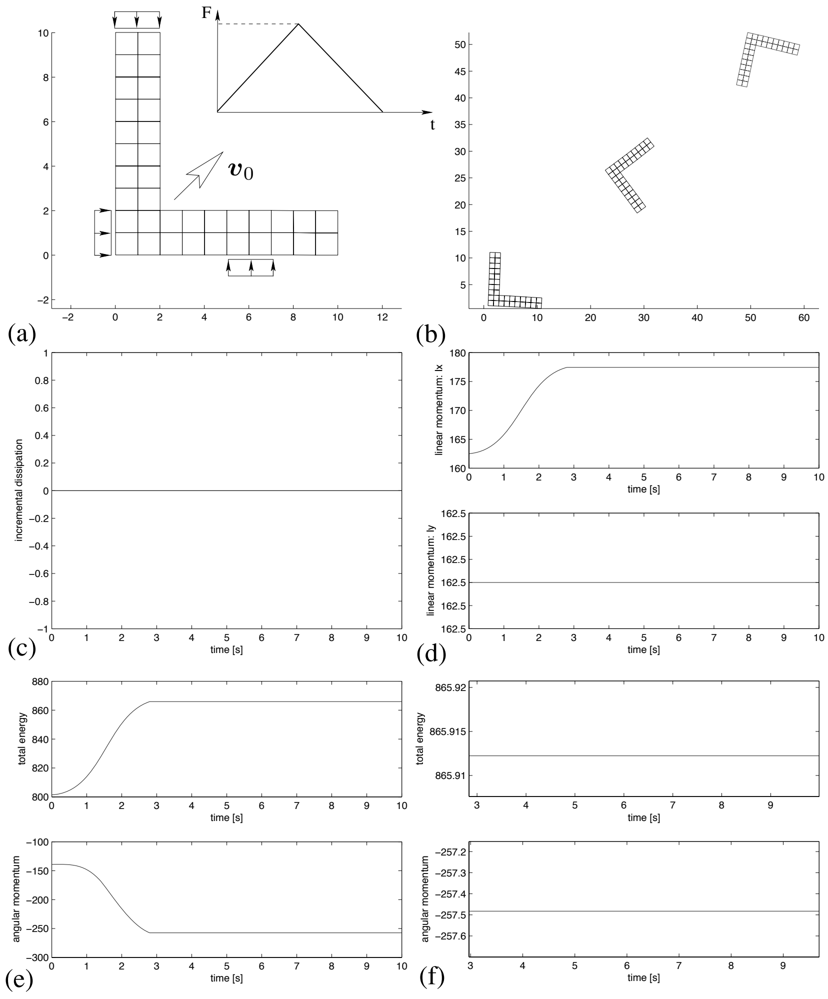


Figure 1: (a) external loading, (b) motion & deformation, (c) incremental dissipation, (d) linear momentum, (e) total energy & angular momentum, (f) zoom: total energy & angular momentum

In Figure 1 (b) we see a sequence of configurations of the nearly unstretched body after several time steps. As claimed in (12) the dissipation is in the elastic case equal to zero, see Figure 1 (c). Important to notice is that the presented time-stepping scheme enables the conservation of the two components of the linear momentum as well as the conservation of the third component L_z of the angular momentum after vanishing external loads. This properties are shown in Figure 1 (d) and Figure 1 (e), (f). Additionally the conservation of the total energy is guaranteed in the elastic case, compare Figure 1 (e). So that – as mentioned before – one important condition for thermodynamical consistency of the time integrator is fulfiled. This conservation property is also satisfied when zooming the plot of the total energy, see Figure 1 (f).

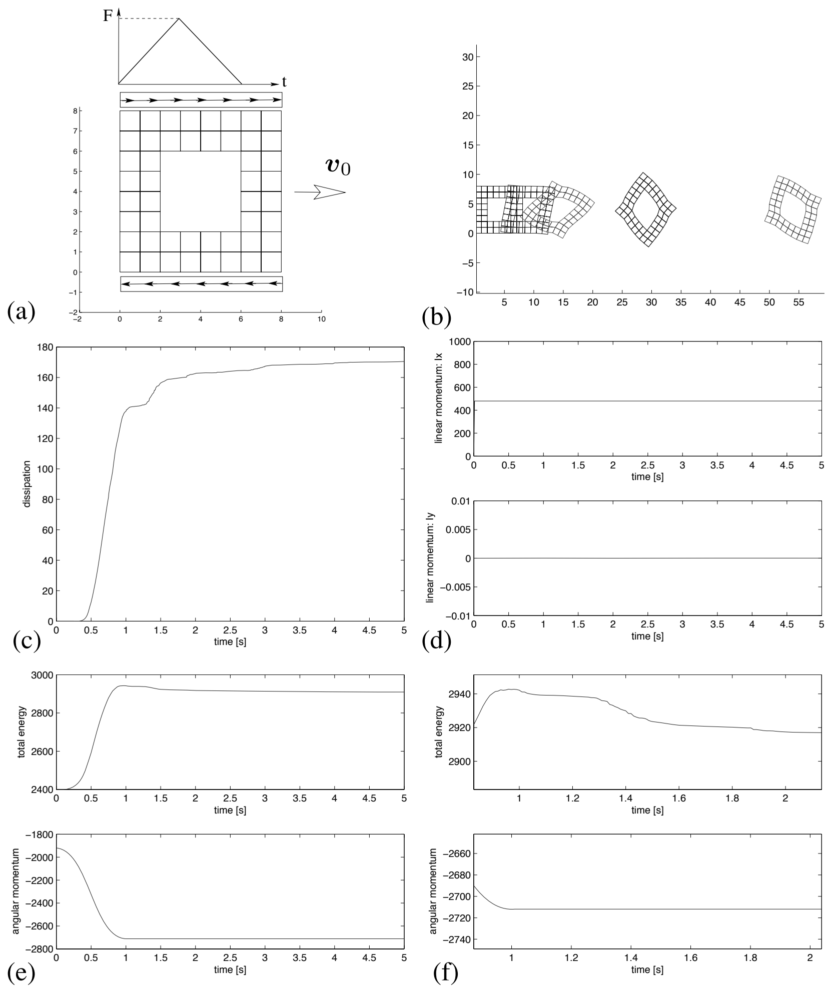


Figure 2: (a) external loading, (b) motion & deformation, (c) accumulated dissipation, (d) linear momentum, (e) total energy & angular momentum, (f) zoom: total energy & angular momentum

4.2 Plastic Motion

The second example deals with the motion of a 'Flying-Frame' with large plastic deformations. The continuous body has been discretised with 48 4-node Finite Elements in space. For this computation the yield function has been supplemented with a nonlinear isotropic hardening part. The incorporated parameters for this simulation are: time-step size $h_n = 0.01$, elastic constants $\lambda = 3000$, $\mu = 750$, initial yield stress $Y_0 = 40$, saturating yield stress $Y_i = 80$ and hardening parameters $H = 10$, $c_p = 36$. The initial velocity has been set to $||\boldsymbol{v}_0|| = 10$ and the time dependent course of the external loads is illustrated in Figure 2 (a).
With the chosen external loading pretty large deformations occur which are contained in Figure 2 (b). In Figure 2 (c) we see that in the plastic case the dissipation inequality (3) is fulfiled by the time-stepping scheme which enables again the conservation of both components of linear momentum and the conservation of L_z, diagrammed in Figure 2 (d) respectively in Figure 2 (e). A further interesting aspect – which can be seen in Figure 2 (e), (f) – is the decrease of the total energy caused by the plastic dissipation corresponding to the strictly positive dissipation.

5 Visualization – Thermodynamical Consistency

So far, the challenges in mechanically and thermodynamically consistent modeling of geometrically nonlinear elasto-plastodynamics have been demonstrated. In this section it will be shown that these special conservation properties of the time integrator open also the way for nonstandard visualization aspects related to one of the characteristic features in view of thermodynamical consistency: the conservation of the total energy H in the elastic case, see (12). The appropriate tool in this context has been the introduction of a nonstandard quadrature rule based on the elastic-enhanced algorithmic stress tensor ${}^{el}\boldsymbol{S}^{alg}$, which has been defined as a function of the standard second Piola-Kirchhoff stress tensor $\boldsymbol{S}$, compare (10). In general, these two symmetric second-order tensors of $\mathcal{B}_0$ have different eigenvalues and also different principal directions. Based on a spectral decomposition we obtain:

$$ {}^{el}\boldsymbol{S}^{alg} = \sum_{i=1}^{n_{dim}} {}^{\boldsymbol{S}}\lambda_i^{alg} \boldsymbol{N}_i^{alg} \otimes \boldsymbol{N}_i^{alg} \qquad \boldsymbol{S} = \sum_{i=1}^{n_{dim}} {}^{\boldsymbol{S}}\lambda_i \boldsymbol{N}_i \otimes \boldsymbol{N}_i \tag{14} $$

This deviation of the different eigenvalues or eigenvectors provides access to a new nonstandard analysis of the thermodynamical consistency of the applied time-stepping scheme in the elastic case, e.g. the differences between the principal directions of the second Piola-Kirchhoff stress tensor $\boldsymbol{S}$ and the elastic-enhanced algorithmic stress tensor ${}^{el}\boldsymbol{S}^{alg}$ are a local measure for the algorithmic behaviour with respect to the desired conservation properties. To visualize this local statement, the eigenvectors of both stress tenors are plotted after several time steps at the integration points of the undeformed reference configuration, see Figure 3 (a), (b). For conceptual clarity the eigenvectors have not been scaled with the corresponding eigenvalue, which, nevertheless, would provide additional information and should be included in a more sophisticated visualization method. In Figure 3 (c) the differences between the principal directions can be seen more clearly. It is important to underline that only with an application of the principal directions and eigenvalues of the algorithmic stresses ${}^{el}\boldsymbol{S}^{alg}$ a thermodynamically consistent modeling of finite elasto-plastodynamics can be offered. With this nonstandard visualization aspect at hand we can investigate two

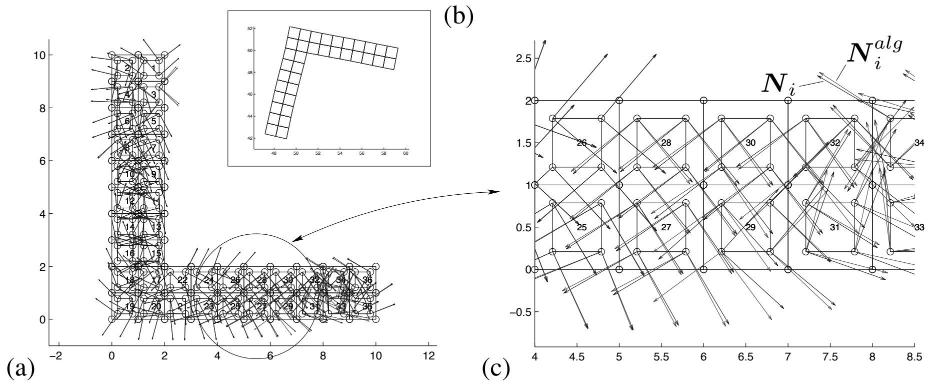


Figure 3: (a) reference configuration $\mathcal{B}_0$ with the eigenvectors $[\boldsymbol{N}_i^{alg}, \boldsymbol{N}_i]$ of the elastic-enhanced algorithmic stress tensor ${}^{el}\boldsymbol{S}^{alg}$ & the second Piola-Kirchhoff stress tensor $\boldsymbol{S}$, (b) deformed configuration $\mathcal{B}_t$ after $10s$, (c) zoom of the principal directions $[\boldsymbol{N}_i^{alg}, \boldsymbol{N}_i]$

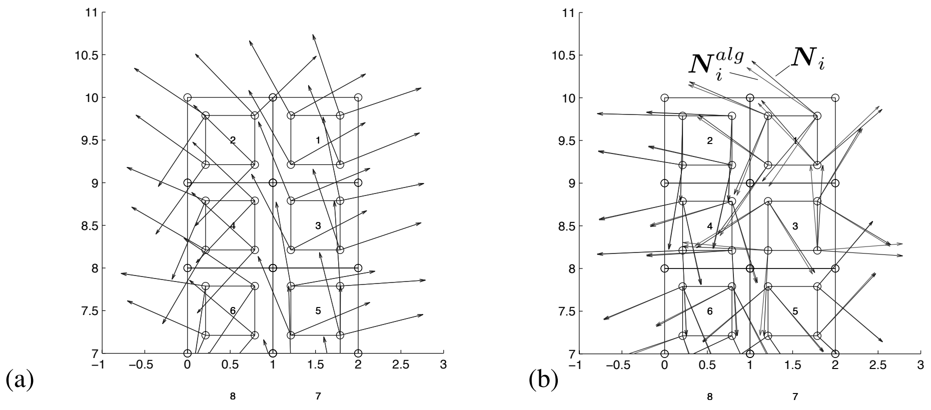
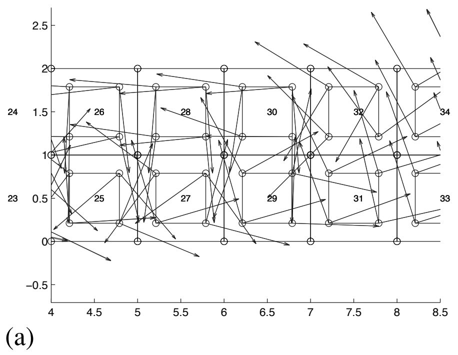


Figure 4: principal directions $[\boldsymbol{N}_i^{alg}, \boldsymbol{N}_i]$ of ${}^{el}\boldsymbol{S}^{alg}$ & $\boldsymbol{S}$: (a) stiff case $\lambda = 100000$, $\mu = 50000$, (b) nonstiff case $\lambda = 1000$, $\mu = 500$

important influences on the local thermodynamical consistency performance of the time-stepping scheme. Once more we focus on the elastic motion of the 'Flying-L' after $10s$.

First, the influence of the stiffness on the algorithmic properties in view of the consistency shall be demonstrated. For the computation the following parameters have been applied: time-step size $h_n = 0.2$, stiff case: elastic constants $\lambda = 100000$, $\mu = 50000$ and nonstiff case: elastic constants $\lambda = 1000$, $\mu = 500$. Considering the zoom of one part of the 'Flying-L' in Figure 4, it can be clearly seen that in the stiff case, plotted in Figure 4 (a), the differences between the principal directions of ${}^{el}\boldsymbol{S}^{alg}$ and $\boldsymbol{S}$ are not as significant as in the case with a reduced stiffness, compare Figure 4 (b).
Another important parameter for the thermodynamically consistent behaviour of the time integrator is the time-step size. This influence shall be illustrated again by means of the elastic motion of the 'Flying-L'. For this example the following parameters have been used: elastic constants $\lambda = 10000$, $\mu = 5000$, a small time step $h_n = 0.02$ and a large time step $h_n = 0.2$. Regarding again the zoom of a particular part of the 'Flying-L', it can be seen that in the case of a rather small time step there are nearly no differences between the eigenvectors of ${}^{el}\boldsymbol{S}^{alg}$ and $\boldsymbol{S}$, see Figure 5 (a). However, we observe a rather high deviation if for the same computation a larger time step is used, compare Figure 5 (b). Additionally we can realise the local distribution of the differences and therewith the local algorithmic achievement of the presented time-stepping scheme. This information cannot be found in plots of the total energy H over the time.



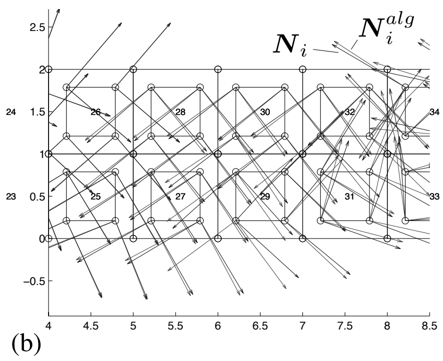


Figure 5: principal directions $[\boldsymbol{N}_i^{alg}, \boldsymbol{N}_i]$ of ${}^{el}\boldsymbol{S}^{alg}$ & $\boldsymbol{S}$: (a) small time step $h_n = 0.02$, (b) large time step $h_n = 0.2$

6 Conclusions

In this contribution we have elaborated a mechanically and thermodynamically consistent Galerkin-based time-stepping scheme for geometrically nonlinear elasto-plastodynamics based on a Hamiltonian-type formulation. This scheme enables the conservation of the linear momentum as well as the conservation of the angular momentum. In addition, the presented time integrator guarantees the conservation of the total energy for the elastic case and strictly positive dissipation for the plastic case which enables a stable integration of the resulting equations. Subsequently, we have demonstrated that these special conservation properties of the algorithm in general require new visualization tasks. In particular the local thermodynamical consistency-behaviour of the time-stepping scheme for the elastic case and the influence of the stiffness and the time-step size on this performance have been worked out.

Acknowledgements:
Financial support by the DFG within the IRTG 1131 is gratefully acknowledged.

References

[1] Armero, F. (2006): Energy-dissipative momentum-conserving time-stepping algorithms for finite strain multiplicative plasticity, *Comput. Methods Appl. Mech. Engrg.*, in press

[2] P. Betsch and P. Steinmann (2001): Conservation properties of a time FE method. Part II: Time-stepping schemes for nonlinear elastodynamics, *Int. J. Numer. Meth. Engrg.*, Vol. 50, pp. 1931-1955.

[3] O. Gonzalez (2000): Exact energy and momentum conserving algorithms for general models in nonlinear elasticity, *Comput. Methods Appl. Mech. Engrg.*, Vol. 190, pp. 1763-1783.

[4] M. Gross (2004): Conserving time integrators for nonlinear elastodynamics, *UKL/LTM T 04-01*, University of Kaiserslautern.

[5] C. Kane, J.E. Marsden, M. Ortiz, and M. West (2000): Variational integrators and the Newmark's algorithm for conservative and dissipative mechanical systems, *Int. J. Numer. Meth. Engrg.*, Vol. 49, pp. 1295-1325

[6] E.H. Lee (1969): Elastic-plastic deformation at finite strains, *Journal of Applied Mechanics*, Vol. 36, pp. 1-6.

[7] X.N. Meng and T.A. Laursen (2002): Energy consistent algorithms for dynamic finite deformation plasticity, *Comput. Methods Appl. Mech. Engrg.*, Vol. 191, pp. 1639-1675

[8] L. Noels, L. Stainier, and J.-P. Ponthot (2005): An energy momentum conserving algorithm using the variational formulation of visco-plastic updates, *Int. J. Numer. Meth. Engrg.*, in press

[9] J.C. Simo (1992): Algorithms for static and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory, *Comput. Methods Appl. Mech. Engrg.*, Vol. 99, pp. 61-112.

[10] J.C. Simo (1998): Numerical analysis and simulation of plasticity, *Handbook of Numerical Analysis*, Vol. 6, pp. 183-499

[11] J.C. Simo, N. Tarnow, and K.K. Wong (1992): Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics, *Comput. Methods Appl. Mech. Engrg.*, Vol. 100, pp. 63-116

[12] J.C. Simo and N. Tarnow (1992): "The discrete energy-momentum method. Conserving algorithms for nonlinear elastodynamics", *Zeitschrift fuer Angew. Mathematik und Physik (ZAMP)*, Vol. 43, pp. 757-792

[13] M. West (2004): Variational Integrators, *PhD-Thesis*, California Institute of Technology, Pasadena

Interaction Technologies for Large Displays - An Overview

Torsten Bierz
University of Kaiserslautern
Computer Science Department
D-67653 Kaiserslautern, Germany
bierz@informatik.uni-kl.de

Abstract: Large Displays and visualizations on such display systems are becoming more and more popular. In order to interact with those systems new devices and interaction techniques must be developed in order to fit the specific needs. In this paper, an overview of the common state of the art devices and techniques will be presented and the pro and cons discussed.

1 Introduction

Nowadays, the size of the displays and their resolution is rising extremely fast. Therefore, the task of efficient and simple interacting is quite challenging. Basic devices, *e.g.* the mouse, suffer first of all from their degree of freedom (DOF). This can easily be shown, when a user is asked to reposition an object in virtual environment [WJ88]. Even if some devices, *e.g.* the SpaceMouse or the SpacePilot, provide the sufficient degree of freedom, the aspect of the stationary usage of these devices is definitely a drawback. In order to be able to interact properly with these devices in front of a large display *e.g.* a table or a desk must be used. This is due to the fact, that most of the devices have been developed for desktop purposes. Consequently, while trying to focus on easy and also intuitive interaction, other devices and technologies are becoming more and more popular.

So, the following chapter will provide an overview of the basic interaction metaphors, which are essential while dealing with large and immersive displays. After that, the most common devices and their corresponding techniques are presented. The paper is finalized with the conclusions and remarks.

2 Interaction Metaphors

Interaction is one of the most important aspects when dealing with virtual environments. It can be separated into the basic tasks of navigation and manipulation [Bow99], whereas the selection metaphor can be combined with the manipulation. The following section will give a short overview of these basic metaphors.

2.1 Navigation

This is one of the basic and most common interaction techniques in a virtual environment. Every navigation technique consists of two basics. The first is the traveling, which controls the motion of the user's viewpoint in the environment. The second is the way finding, where the user determines the path by the knowledge of the environment or visual information, *e.g.* signs. Other supporting aids are maps, top–views, the well known *World in Miniature* metaphor [SCP95], or the *voodoo dolls* technique [PSP99]. So, when building up an efficient virtual environment, the navigation should be considered as an essential technique. However, most of the time traveling is just a basic task in order to perform a more important task for interaction further with the virtual environment, *e.g.* grabbing or moving of an object.

2.2 Selection and Manipulation

After the efficient user navigation to the desired object or target, the selection of the object is usually the next step. The selection can be performed by different purposes, *e.g.* by a command, or mapping. After the selection is finished, manipulation, *e.g.* translation or rotation of the object, can be performed. Based on trying to focus on interacting with the virtual environment in a most natural way, natural mapping simply maps the location and the scale of the user's physical hand to the virtual hand. So, when the user tries to grab an object in the virtual world, it is automatically selected and manipulated.

Although this interaction is very intuitive and is easily adapted to most users, a big disadvantage for the interaction is the physical limit of the users arm. Avoiding this issue, the *Arm–Extension Technique* solves this problem. This approach allows the extension of the virtual hand and interaction with far away objects. Within the threshold distance, the user interacts in a normal way. Outside the threshold, the virtual arm follows a nonlinear function relative to the distance of the physical arm from the user's body, *e.g.* the Go-Go technique in [PBWI96]. So, this mapping function is the important improvement based on this approach. However, the precision *e.g.* by position lacks in the far away distance. The *World in Miniature* technique [SCP95] as well as the *voodoo dolls* technique [PSP99] can also be used for selection and manipulation. Furthermore, the miniature representation of the objects can also be manipulated with both hands.

In order to select objects, the ray–casting techniques are commonly used, where the mouse or more generally a pointer is used to select an object. The three dimensional approach is comparable to the well known computer graphics ray casting technique. So, the object is selected by pointing a virtual ray into the scene. When the interaction is performed with a users hand and not a pointing device, the orientation of the hand is mostly used in order to specify the direction of the virtual ray. If an object intersects the ray, it will be selected and ready for further manipulations. Based on this technique is also the image plane technique, where the selection is performed by covering the desired object with the virtual hand [PFC$^+$97].

Dealing with the selection task in front of large scale devices, two main problems affect navigation beyond the reaching problem: the *bezel interference* and the *cursor tracking*. The navigation or selection sometimes becomes difficult due to the fact that a physical barrier might exist. These are areas of the screen, where the virtual (screen) space is not visible. Several approaches exist which provide a suitable solution to the problem, see *e.g.* [BCHG04]. Furthermore, the user can easily lose sight of the cursor's location due to the large number of potential distractors, or lose the position while moving. Various solutions for this problems exist, *e.g.* a high-density cursor [BCR03], or lost cursors [RCB$^+$05, KMFK05].

3 Large Display Interaction

Nowadays all different kinds of large displays exist. The most common ones are CAVE ®, PowerWalls, all different kinds of curved screens or tiled display systems, *e.g.* the HIPerWall by Küster *et al.*. The usage of basic devices or even haptic devices for interaction is very uncomfortable, unnatural, and often downright impossible. In order to simplify the interaction and provide a more suitable interface in front of this huge display systems, new devices have been developed. In the following sections, the main focus is on devices and techniques, which can be used for interaction with large displays.

3.1 Laser Pointer

Nowadays, there have been many approaches using laser pointer as interaction devices, *e.g.* by Haman *et al.* [ATK$^+$05]. Usually a camera captures the position of the red laser dot on the large display. The resulting position is used as a cursor for interaction purposes. However, the pointing task is often not accurate, because of the action tremor of the user, or the unsteadiness of the users hand. This effect even becomes worse by incrementing the distance from the user to the large display. Cheng and Pulo proposed in [CP03] the replacement by circulating gestures, in order to avoid this issue. However, these devices are really efficient for two dimensional interaction tasks, but they lack when dealing with three dimensional interaction tasks.

3.2 Wands

Associated with pointing devices like laser pointer are wand devices, *e.g.* the *Magic Wand* by Ciger *et al.* [CGVT03] or the *Visual Wand* [CB03]. These approaches are using tracked, sometimes color coded wands. The color coding is used, in order to solve the problem of the orientation of the wand. However, there resists still one missing degree of freedom, because the wand itself represents a rotation axis. This axis is not well suitable for performing efficient rotations. This can easily be demonstrated, when holding a pen and trying to rotate it in the hand.

3.3 Tracking Technologies

In order to control the virtual environment, tracking of alternative interaction devices or even the tracking of the position and gestures of the user becomes very common. In such cases, tracking a device's position and often its orientation is needed in order to make meaningful measurements on the user's intentions.

The two mainly interesting technologies, which provide sufficient update rates, accuracy and mobility of the user are the magnetic and optical tracking systems. These systems and their technology will be shortly presented in the following section.

Magnetic Tracking One of the most common tracking technologies is the magnetic tracking. By using a magnetic field, the position of the receiver element can be calculated without being dependent on a direct line of sight. There are two different types of magnetic trackers: AC and DC trackers. AC trackers use an alternating field of 7-14 kHz and DC tracker use pulsed fields. Nixon *et al.* [NMFP98] showed that the trackers can be influenced or distorted by ferromagnetic metals or copper. The metallic distortion for DC fields is significantly less sensitive than for AC fields. However, the DC tracking systems interfere with magnetic fields generated by ferromagnetic objects, *e.g.* loudspeakers, or monitors. Furthermore, the error rate is increasing dependent on the distance between transmitter and receiver. This influence was also noted by Fikkert [Fik06], who tried to obtain ground-truth for passively obtained user head orientation estimations. The driving simulator in which these measurements should be obtained contained great pieces of metal, influencing measured orientations greatly.

Optical Tracking Optical trackers go beyond magnetic trackers; the most often used tracking systems. Optical sensors (cameras) are used, in order to determine the position and the orientation of an object. The position can then be calculated using trigonometrical mathematical equations. In order to simplify the tracking, marker can be used. The trackable marker can be represented as pattern or balls in every size, where pattern recognition techniques are used in order to transmit more information. However, being dependent on a marker and the position of the marker on the interacting person is not really intuitive and cumbersome.

Nowadays, researchers are focusing on markerless tracking *e.g.* [FM05, CMZ05]. In markerless tracking the position of the users hand and head is detected and used for interaction purposes [MSG01]. Carranza *et al.* [CTMS03] build up a markerless tracking system, which uses common computer vision techniques in order to estimate the user's position and gestures in the real world. The data are then mapped to a virtual model. This interaction is more intuitive. Research is also done on real time tracking of human eyes [Han03, Duc03, ZFJ02] or faces without any markers [ZCPR03]. Not being dependent on any further devices markerless tracking and its corresponding interaction has become one of the most challenging topics in computer vision research. Moeslund and Granum [MG01] provided an overview of computer vision based human motion capturing, which currently is the main focus of the computer vision research area.

One big advantage of optical tracking is the immunity to ferromagnetic materials and the low latency time, which make them very attractive for researchers. However, optical tracking systems are also dependent on the line of sight. In order to solve this issue, different filter techniques, *e.g.* Kalman filters [MB89, WB01] or Wiener filters [Wie64], are used. The filter try to estimate the position or orientation of the target depending on the previously measured position or orientation. Furthermore, by using of high speed and high resolution cameras, a high accuracy can be obtained.

3.4 Gaze Tracking

One of the application areas for optical tracking is the so called gaze tracking or eye–tracking. This tracking technology measures the eye positions and the eye movements. Withal other method for pointing, the eye motion is the fastest [WM87]. However, when being used as input method, the eye motion conflicts with the primary usage for visualization purposes [Zha03]. Furthermore, the task of selecting of objects smaller than one degree of visual angle can not be achieved sufficiently [WM87]. Recently, some research is being done in order to solve this problem [vM05].

Some researchers are trying to focus on the combination of gaze tracking technology and other input devices, *e.g.* keyboard or mouse, in order to enhance pointing tasks. This can be achieved by setting the initial point via eye tracking and adjusting the result with an external device [ZMIC99]. This approach can be extended to collaborative environments [Zha03].

In order to detect a user's focus, gaze detection can be used an appropriate possibility. Therefore, the gathered information is used as a basis for gaze-contingent rendering [BDDG03]. As the displays are getting larger, it is becoming more and more popular. However, the massive communication bandwidth needed to deliver gigapixel or higher graphics at satisfying refresh rates is one of the rizing challenges in visualization [WL05].

3.5 Gesture Interfaces

Finding a natural way of interaction is a huge effort. One possibility is the usage of gesture-recognizing interfaces, that provide more accuracy than tracking devices and are very intuitive. Nowadays, most devices used for this interaction technique are gloves with embedded sensors. According to the sensor values, the position of the fingers can be calculated. For gesture recognition, these positions are compared to a stored set of defined gestures. The gesture is identified and used for the interaction task for the virtual environment. Another possibility is the interaction with virtual objects in the virtual environment. So, while interacting the user has the advantage of a multi finger interaction fulfilling different tasks, which provides a higher flexibility and definitely simplifies many tasks. The data gloves can be used not only for grabbing issues but also for the navigation or other tasks specified by the gesture library.

In order get the correct position in space, the data glove has to be connected to a tracking system. Otherwise the data glove is not really efficient. The well known state-of-the-art devices are the CyberGlove by Immersion, the 5DT Data Glove by Fifth Technology, the P5 Cyberglove by EssentialReality and the Pinch® Glove by Fakespace. Not every device has the same properties. The main differences amongst these state-of-the-art devices are the different sensor types, the sampling rates of the glove, and the interface connection.

A clear disadvantage of using datagloves is the varying size of the human hand between users. This results in different sensor locations for each user that needs to be accounted for. Typically, for every new user the data gloves have to be reconfigured in order to efficiently match the user specific configuration of the fingers and size of the hand. The Pinch Glove needs not be calibrated in this manner. It measures the contacts and the contact time of the fingertip, the finger back and palm electrodes. Consequently, there's no possibility of a specific readout of the finger configuration, which are typically the angles of the fingers.

3.6 User Tracking

Considering that every person has the ability to perceive the current orientation and the position of the body or parts of body, this perception can be used for navigation [MFPBS97]. The body tracking itself can be decomposed into finger, gesture/hand, head, and full body tracking. Most recent applications of these tracking methods use vision-based tracking. The different body tracking parts will be described in this section.

Finger tracking can be achieved with touch-based input; gestures constrained to the plane can also be detected this way. At arms length, however, additional technology is needed. The previously described gesture interfaces have been used extensively in virtual reality applications to track fingers and gestures. However, tracking resolutions have limited their applicability for fine-detail tracking until recently [VB05]. Many research groups are focusing on vision-based methods, which can segment the fingers and hand. These detected gestures can either be captured on a surface [Wil04, MRB05] or in the air [VB05, CC02]. Finger and gesture tracking is ideal for selection, though the wrist is not well suited to six degree-of-freedom tasks [Zha98].

Head tracking is often used to identify a user's location or the direction of their gaze; head tracking can be combined with eye tracking for this latter application. Head tracking can be done in a tethered (magnetic/inertial-based) or untethered (vision-based) fashion. One example of head tracking is its use to augment the accuracy of finger/gesture tracking [NS02]. A novel application is using gaze and facial expression (such as frowning) as input [SII04]; for example, one could consider using frowning as negative feedback to a semi-automatic visualization technique. However, except for its use in collaboration (indicating gaze), head tracking is not an ideal input for visualization applications which require fine motor control.

Full body tracking determines the body's location and pose. In order to navigate in 3D, body pose/position tracking has been used with partial success [JJLFKZ01]. However, body pose tracking is less fine-grained than even head tracking [VB04]. Concerning this fact, it can be efficient for macro-scale or rough initial interaction. However, for fine-grain manipulations and selections it is not a very satisfying way of interaction.

4 Conclusion

So, in this paper, an overview of the current state of the art interaction technologies for large displays has been presented. It based on the book chapter "Interacting with Visualizations" in [KEM07], which provides a more detailed view of the topic.

Concluding, it has been shown that many devices and many technologies exist, all having different advantages and disadvantages. So, when dealing with large displays, researches should try to focus on more natural ways of interacting *e.g.* the tracking or gesture recognition. This form of interaction can easily be learned and adapted. Furthermore, it is similar to the natural behavior of humans. Keep in mind, that some interaction devices are still heavy, and can lead to fatigue or have stationary usage *e.g.* the SpaceMouse.

The multi user interaction or collaborative interaction is still one of the hot topics today. Suitable solutions and new metaphors have to be found in order to simplify the interaction purpose.

5 Acknowledgment

I would like to thank the members of the Computer Graphics and Visualization Research Groups in Kaiserslautern and Irvine, as well as the members of the International Research Training Group (IRTG) for their cooperation, especially Achim Ebert and Jörg Meyer. The IRTG is supported by the German Research Foundation (DFG) under contract DFG GK 1131.

References

[ATK$^+$05] Benjamin A. Ahlborn, David Thompson, Oliver Kreylos, Bernd Hamann, and Oliver G. Staadt. A practical system for laser pointer interaction on large displays. In *VRST: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 106–109, 2005.

[BCHG04] Patrick Baudisch, Edward Cutrell, Ken Hinckley, and Robert Gruen. Mouse ether: Accelerating the acquisition of targets across multi-monitor displays. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1379–1382, New York, NY, USA, 2004. ACM Press.

[BCR03] Patrick Baudisch, Edward Cutrell, and George Robertson. High-Density Cursor: A Visualization Technique that Helps Users Keep Track of Fast-moving Mouse Cursors. In *Proceedings of IFIP INTERACT'03: Human-Computer Interaction*, 2: Display I/O, page 236, 2003.

[BDDG03] Patrick Baudisch, Doug DeCarlo, Andrew T. Duchowski, and Wilson S. Geisler. Focusing on the essential: considering attention in display design. *Communications of the ACM ACM*, 46(3):60–66, 2003.

[Bow99] D. Bowman. Interaction Techniques for Common Tasks in Immersive Virtual Environments, 1999.

[CB03] Xiang Cao and Ravin Balakrishnan. VisionWand: interaction techniques for large displays using a passive wand tracked in 3D. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 173–182, New York, NY, USA, 2003. ACM Press.

[CC02] A. Corradini and P. Cohen. Multimodal speech-gesture interface for hands-free painting on virtual paper using partial recurrent neural networks for gesture recognition. In *Proceedings of the International Joint Conference on Neural Networks*, volume III, pages 2293–2298, 2002.

[CGVT03] Jan Ciger, Mario Gutierrez, Frederic Vexo, and Daniel Thalmann. The magic wand. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics*, pages 119–124, New York, NY, USA, 2003. ACM Press.

[CMZ05] Marcio C. Cabral, Carlos H. Morimoto, and Marcelo K. Zuffo. On the usability of gesture interfaces in virtual reality environments. In *CLIHC '05: Proceedings of the 2005 Latin American conference on Human-computer interaction*, pages 100–108, New York, NY, USA, 2005. ACM Press.

[CP03] Kelvin Cheng and Kevin Pulo. Direct interaction with large-scale display systems using infrared laser tracking devices. In *CRPITS '24: Proceedings of the Australian symposium on Information visualisation*, pages 67–74, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[CTMS03] Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 22(3):569–577, 2003.

[Duc03] Andrew T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[Fik06] F.W. Fikkert. Estimating the Gaze Point of a Student in a Driving Simulator. *International Conference on Advanced Learning Technologies - Advanced Technologies for Life-Long Learning*, 6, July 2006. to appear.

[FM05] James Fung and Steve Mann. OpenVIDIA: parallel GPU computer vision. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 849–852, New York, NY, USA, 2005. ACM Press.

[Han03] D.W. Hansen. *Committing Eye Tracking*. PhD thesis, IT University of Copenhagen, July 2003.

[JJLFKZ01] Jr. Joseph J. LaViola, Daniel Acevedo Feliz, Daniel F. Keefe, and Robert C. Zeleznik. Hands-free multi-scale navigation in virtual environments. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 9–15, New York, NY, USA, 2001. ACM Press.

[KEM07] Andreas Kerren, Achim Ebert, and Jörg Meyer, editors. *Human-Centered Visualization Environments*. LNCS Tutorial. Springer-Verlag, to be published 2007.

[KMFK05] Azam Khan, Justin Matejka, George Fitzmaurice, and Gordon Kurtenbach. Spotlight: Directing users' attention on large displays. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 791–798, New York, NY, USA, 2005. ACM Press.

[MB89] G.V. Moustakides and J.-L. Botto. Stabilizing the fast Kalman algorithms. *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on*, 37:1342–1348, 1989.

[MFPBS97] Mark R. Mine, Jr. Frederick P. Brooks, and Carlo H. Sequin. Moving objects in space: exploiting proprioception in virtual-environment interaction. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 19–26, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[MG01] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Comput. Vis. Image Underst.*, 81(3):231–268, 2001.

[MRB05] Shahzad Malik, Abhishek Ranjan, and Ravin Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 43–52, New York, NY, USA, 2005. ACM Press.

[MSG01] Thomas B. Moeslund, Moritz Storring, and Erik Granum. A Natural Interface to a Virtual Environment through Computer Vision-Estimated Pointing Gestures. In *Gesture Workshop*, pages 59–63, 2001.

[NMFP98] Mark A. Nixon, Bruce C. McCalum, W. Richard Fright, and N. Brent Price. The Effects of Metals and Interfering Fields on Electromagnetic Trackers. *Presence*, 7(2):204–218, 1998.

[NS02] K. Nickle and R. Stiefelhagen. Pointing gesture recognition based on 3D-tracking of face, hands, and head-orientation. In *Proceedings of the International Conference on Multimodal Interfaces*, pages 140–146, 2002.

[PBWI96] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80, New York, NY, USA, 1996. ACM Press.

[PFC$^+$97] Jeffrey S. Pierce, Andrew S. Forsberg, Matthew J. Conway, Seung Hong, Robert C. Zeleznik, and Mark R. Mine. Image plane interaction techniques in 3D immersive environments. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 39–ff., New York, NY, USA, 1997. ACM Press.

[PSP99] Jeffrey S. Pierce, Brian C. Stearns, and Randy Pausch. Voodoo dolls: seamless interaction at multiple scales in virtual environments. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 141–145, New York, NY, USA, 1999. ACM Press.

[RCB$^+$05] George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan. The Large-Display User Experience. *IEEE Computer Graphics and Applications*, 25(4):44–51, jul/aug 2005.

[SCP95] Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual reality on a WIM: interactive worlds in miniature. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[SII04] Veikko Surakka, Marko Illi, and Poika Isokoski. Gazing and frowning as a new human-computer interaction technique. *ACM Transactions on Applied Perception*, 1(1):40–56, 2004.

[VB04] Daniel Vogel and Ravin Balakrishnan. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 137–146, New York, NY, USA, 2004. ACM Press.

[VB05] Daniel Vogel and Ravin Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 33–42, New York, NY, USA, 2005. ACM Press.

[vM05] Oleg Špakov and Darius Miniotas. Gaze-based selection of standard-size menu items. In *ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*, pages 124–128, New York, NY, USA, 2005. ACM Press.

[WB01] G. Welch and G. Bishop. An introduction to the kalman filter, SIGGRAPH 2001 course 8. In *In Computer Graphics, Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press Addison-Wesley Publishing Company, August 2001.

[Wie64] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[Wil04] Andrew D. Wilson. TouchLight: An imaging touch screen and display for gesture-based interaction. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 69–76, New York, NY, USA, 2004. ACM Press.

[WJ88] Colin Ware and Danny R. Jessome. Using the Bat: A Six-Dimensional Mouse for Object Placement. *IEEE Comput. Graph. Appl.*, 8(6):65–70, 1988.

[WL05] Benjamin Watson and David P. Luebke. The Ultimate Display: Where Will All the Pixels Come From? *IEEE Computer*, 38(8):54–61, 2005.

[WM87] Colin Ware and Harutune H. Mikaelian. An evaluation of an eye tracker as a device for computer input. In *Graphics Interface '87 (CHI+GI '87)*, pages 183–188, April 1987.

[ZCPR03] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458, 2003.

[ZFJ02] Zhiwei Zhu, Kikuo Fujimura, and Qiang Ji. Real-time eye detection and tracking under various light conditions. In *ETRA '02: Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 139–144, New York, NY, USA, 2002. ACM Press.

[Zha98] Shumin Zhai. User performance in relation to 3D input device design. *Computer Graphics*, 32(4):50–54, 1998.

[Zha03] Shumin Zhai. What's in the eyes for attentive input. *Communications of the ACM*, 46(3):34–39, 2003.

[ZMIC99] Shumin Zhai, Carlos Morimoto, Steven Ihde, and Research Center. Manual and Gaze Input Cascaded (MAGIC) Pointing. In *Proceedings of ACM CHI 99 Conference on Human Factors in Computing Systems*, volume 1 of *Gaze and Purpose*, pages 246–253, 1999.

GPGPU: General Purpose Computing on Graphics Hardware

Pushpak Karnick
Arizona State University
Tempe, Arizona 85287 USA
pushpak@asu.edu

Abstract: The graphics processor (GPU) on current personal computers has developed into a rich and stable platform for developing scientific applications which do not fall under the category of traditional computer graphics software. The application of the GPU as a SIMD processor for scientific computing has proven to be a very powerful tool to an extent that various graphics card manufacturers have started taking GPGPU into consideration when designing and developing the next generation hardware. The GPUs are a magnitude faster than the present generation CPUs due to high-bandwidth memory availability; and a more elegant parallel-programming platform due to true multi-threading units in hardware. We present a brief introduction to this rapidly growing field, discussing the basics of stream programming and then applying this generic framework to the GPU programming model.We present two examples from the wide spectrum of GPGPU applications, which illustrate the underlying concepts.

1 Introduction

The Graphics Processing Unit (GPU) has evolved into a mature platform for performing fast vector operations suited for the graphics pipeline. A short look at the latest hardware aptly demonstrates the tremendous rate of development in the manufacture of faster GPUs, a trend which overtakes similar development on traditional processors (CPUs). The main driving force behind this development is the gaming market, providing a steady user base and an annual turnover running into millions of US dollars. As a result, the GPUs today are more powerful than the conventional CPUs, delivering up to 45-50 GFLOPS of throughput and packing more transistors than a state-of-art dual core CPU [NVI06, ATI06, AMD06]. The GPU can therefore be thought of as a computing co-processor, for delegating computationally intensive tasks, where the same operation is applied on millions of data elements. Applications such as calculation of generalized 2D/3D Voronoi diagrams [KEHKL$^+$99] have already proven that the GPU is capable of operating beyond the graphics pipeline. While scientific applications reap the most benefit of having a second, much faster processor to offload computation [BFGS03], traditionally non-scientific applications like database query processing [GLW$^+$04] have also been shown to benefit from the additional power of the GPU. In this paper, we will introduce the reader to the process of porting general purpose scientific applications to the GPU.

The next section provides a brief introduction of the stream programming model. We provide definitions of the basic concepts, and highlight important characteristics of such a programming model. Section 3 introduces the GPU as an efficient stream processor, mapping the concepts introduced in the previous section to their counterpart implementations on the GPU. Sections 4 and 5 present example methods which use the GPU for scientific computing. We conclude with our remarks on the future of GPGPU and provide directions for the curious reader.

2 Stream Programming

Stream programming is a programming model developed for processors that can process multiple data records/packets at once viz., the so-called SIMD (Single Instruction Multiple Data) and MIMD (Multiple Instruction Multiple Data) processors. It was originally developed for vector processors [Ble90]. The processor is modeled as a “black box” that takes as input, one or more data packets and produces one or more output packets. Owens [Owe05] presents an excellent overview of recent trends in streaming architectures. We define the following terms wrt. such a processing paradigm:

- **Stream :** An ordered sequence of data packets / records, where the packets belonging to a stream are of the same type.
- **Kernel :** An algorithm which takes an input stream and produces corresponding output stream. The kernel operates upon the input stream and produces the corresponding output stream.

An important characteristic of a kernel is that it is defined and operates on an *entire stream*, and not on individual packets. This is the most useful property for us, because it corresponds directly to the GPU programming model.

The main bottlenecks of stream processing are communication latency and data dependency. Communication latency can be minimized if we can cache all partial results locally instead of accessing the main memory for each access. Data dependency is a domain-specific issue, and can at most be minimized, but not eliminated entirely for grid based simulations. We will show how mapping the input data to a 2D texture on a GPU solves this problem. We will now see how these characteristics can help us port our applications to the GPU.

3 GPU Programming As Stream Processing

3.1 GPU Architecture

The GPU programming model can be thought as a special case of the generic stream processing model developed for vector processors. The GPU provides two levels of programmable processors viz., the vertex processor and the fragment processor. The vertex processor takes as input, a stream of vertex positions and associated per-vertex attributes, and generates a stream of output vertices which are slated for rendering. These vertices are assembled into triangles, and then pass through the rasterizer, where a stream of fragments (proto-pixels) is generated and fed to the fragment processors. Typically, hundreds of fragments may be generated for a handful of vertices. Hence, there are more parallel pathways for the fragment data (24 in NVIDIA GTX 7800) than for vertices (8 in NVIDIA GTX 7800). We will exploit this property when we implement our "kernels" – implementing a kernel as a fragment program allows us a higher degree of parallelism than as a vertex program. Also, it is more efficient to access memory from the fragment processor, than it is from the vertex processor. In some cases, we cannot access random memory from the vertex processor at all. Hence, the fragment processor is the suitable candidate to implement our computation kernels.

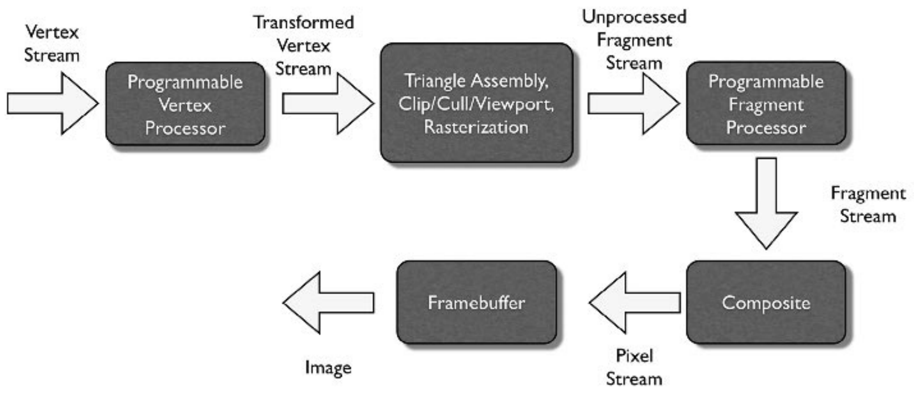

Vertex Stream
Programmable Vertex Processor
Transformed Vertex Stream
Triangle Assembly, Clip/Cull/Viewport, Rasterization
Unprocessed Fragment Stream
Programmable Fragment Processor
Fragment Stream
Composite
Pixel Stream
Framebuffer
Image

Figure 1: The GPU pipeline as a stream model. The GPU processes an input stream ($vertices$ / $fragments$) to produce a corresponding output stream ($fragments$ / $pixels$).

As seen in figure 1, the GPU is hardwired to efficiently process 3/4-component vector data ($RGBA$ or $xyzw$). Hence, the first issue at hand is to pack input data into a format which can be efficiently processed by the GPU pipeline. Data that can be put into 3D or 4D datatypes is mostly passed in either as vertex data, or as texture data. Memory reads on a GPU correspond to a texture fetch in the fragment shader, and memory writes can be achieved by setting the pipeline state to render-to-texture. Note that since the fragment program cannot change the location of the pixel, the only way to write to a random memory location is via output to textures. The fragment program typically executes the "kernel" on a per-fragment basis. The program is invoked by drawing a full screen quad with at least as many pixels rendered as are the number of data points. Changing the vertex locations helps in deciding which fragments will be processed, and thus can help in a distributing the computational range over multiple passes of the same pipeline. The fragment program can be considered as the "inner loop" in a typical CPU based implementation, the *algorithm step* that is executed for each data element. Feedback is also implemented using render-to-texture functionality. The output of the previous iteration is a texture which can be accessed in the fragment program for the next iteration.

3.2 GPGPU Programming Issues

Several fundamental changes need to be made when deciding to port an application from the CPU to a modern GPU. Foremost amongst them is the question whether the additional overhead of packing real-world data into 3/4-component vector format optimized for the graphics pipeline offsets the gain in computation speed. Memory transfers are a major bottleneck in any PC-based architectures, and the primary objective in such an exercise would be to reduce memory transfers to and from the main memory to the memory on the graphics card. Tools like ShaderPerf (NVIDIA), Shark and OpenGL Profiler (Apple) help in analyzing and optimizing memory transfers to/from the GPU.

Once an optimal configuration has been found, the next step involves exploiting data parallelism on the GPU itself. Scalar data is most often used as an input to many scientific applications, and the limited real-estate on the graphics memory makes data compaction a necessary task to process large volumes of data. Figure 2 describes two ways in which such a compaction could be achieved.

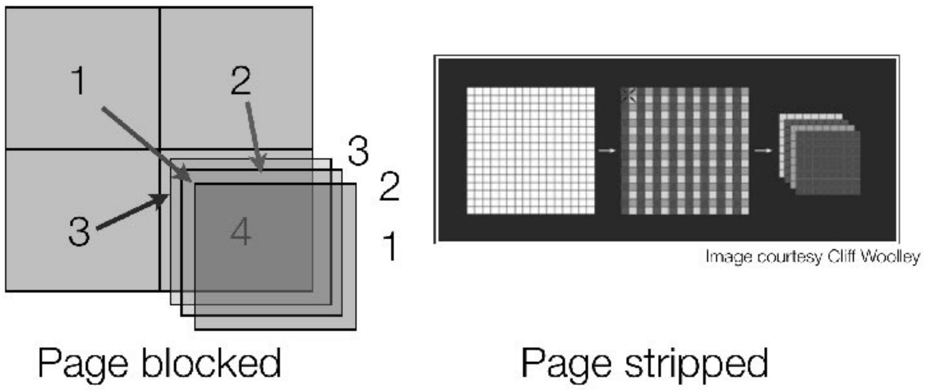


Figure 2: *Page blocked* and *Page stripped (with stride)* memory layouts for achieving data parallelism.

In the first case, individual blocks of memory are “paged” into the RGBA channels of a texture map. This mechanism can drastically reduce the memory requirement to a one-fourth of the actual data stored, with no additional overhead. This method is similar to the *Page blocked* configuration in main memory. The second example shows adjacent data elements being packed into a single *4-tuple*, thus giving access to all neighboring values in one texture fetch. We call this method *Page stripping*.

4 Examples: *N-body Simulation*

In this section, we study the GPU version of the classical *N-body Simulation* as described in [LHK$^+$04]. The N-body simulation is a general problem with applications in fields as diverse as astrophysics to molecular interactions. The problem can be stated as follows:

Given: N physical bodies $\{N_1, N_2, ..., N_n\}$ in general position, with mass $\{m_1, m_2, ...,$ $m_n\}$ respectively.

To Find: The total force acting on each of the N massive bodies.

Formula: The formula to calculate the force acting on a body N_i due to a body N_j is given by

$$F_j^i = G * \frac{m_i * m_j}{d^2}, i \neq j$$

where, G = Universal constant of gravity, and d = Euclidean distance between the two bodies.

In 3D, the positions of the bodies are given as a vector of three values (one for each dimension). These values can be readily stored on the graphics card as a floating point texture with the RGB channels holding the xyz values respectively. The index for accessing the position of a the i^{th} body is now a pair of $(row, column)$ values derived from a linear index n. The masses of the bodies are scalar quantities, and may be compacted to reside into a RGBA texture of size $N/4$. The force F_j^i is the output of the fragment program which executes for every pair of bodies in the input, and can be similarly calculated and stored in a texture.

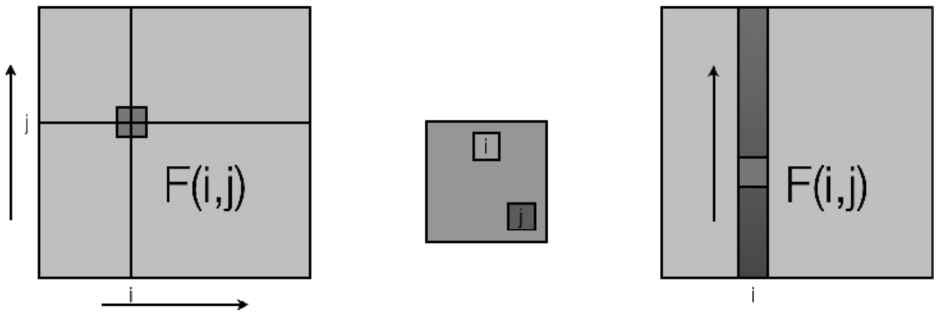


Figure 3: The Force F_j^i can be stored in a texture with the i^{th} column storing the forces acting on body N_i (left). The indices i and j are dereferenced into a pair of (row,column) values (middle) to retrieve the positions of the bodies N_i and N_j respectively. The output texture stores all forces acting upon the body i in the i^{th} column (right). A sum of all the values in one column gives us the total force F^i acting on N_i.

The arrangement is shown in the Figure 3. The "kernel" (fragment program) is invoked for each pixel on the screen by drawing a full screen quad. The fragment program reads the position values from the position texture based on the fragment's row and column position on screen and calculates the force using the above formula. The calculated value is written out to the output texture. At the end of this pass, every texel (i, j) of the output texture gives us the force on body i due to body j. We are interested in the $total$ force acting upon a body due to the other N-1 bodies. Observe that the column i holds all the necessary values to compute the same. We only need to sum up the row values for every column to find the total force acting upon the body. Since there is no dependency amongst the columns, this operation can be done in parallel. Moreover, using a clever trick, we can also reduce the time taken for this addition from $O(N)$ to $O(log_2(N))$. We repeatedly sum the lower and upper halves of the texture (and reset the active lower/upper bounds accordingly), till we are left with one row. This is the well known *parallel reduction* method (see Figure 4).

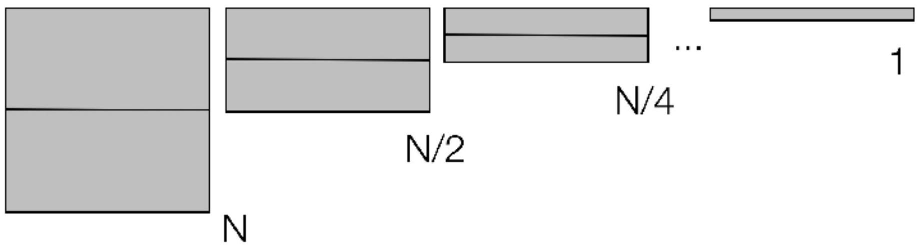


Figure 4: Repeated addition between lower and upper halves of *active* texture bounds till we are left with only one row sums N columns in parallel in $log_2(N)$ steps.

The first row of the output texture now holds the resultant force acting on the body. A simple pixel shader calculates the new positions of the bodies based on the present velocities as given by the formula

$$x(i, t + \delta t) = x(i, t) + u(i, t) * dt.$$

To calculate the new velocities of the bodies based on the above force calculation, we apply the force F^i to the present velocity of the body i. Hence,

$$u(i, t + \delta t) = u(i, t) + F^i * dt.$$

Another pixel shader can accomplish this task by reading the values off the output texture of the previous pass, and writing the results into a new velocity texture. The values of the new positions / velocities may overwrite the older values in-$place$ to save space. We repeat the whole process again, till a specified number of steps have elapsed, or an equilibrium under some threshold has been reached.

The *N-body Simulation* is a simple example which demonstrates the important steps necessary to transform a scientific simulation from CPU based procedural code, to a parallel implementation on the GPU, viz.,

- **Data Transformation:** The input data (positions of the bodies) is transformed into a three component floating point texture to facilitate random access read/write on the GPU.

- **Index Transformation:** The positions and velocities of the bodies are now indexed by 2D array indices which are obtained from linear index in the domain.
- **Output/Feedback Mechanism:** The output of every step is written into a texture, which is then read in the next "pass" by the corresponding fragment program. The final output is also rendered to a texture, which is then read into the main memory to obtain the final results.

We now turn to a more complex implementation - Linear Algebra operators [KW03].

5 Example: *Linear Algebra Toolset*

In this section, we take a look at the GPU implementation of linear algebra operators as described in [KW03]. Earlier works include faster matrix multiplication on GPUs [THO02], sparse matrix solvers [BFGS03], and numerical solutions for least squares problems [HMG03]. As seen in the previous section, 2D textures were used to represent vector data. However, as we saw, obtaining final result from such a representation requires post-processing steps (like parallel reduction). Also, texture access happens through a mapping mechanism which transforms linear indices into *(row, column)* index pairs. To solve a majority of performance issues related with such representations, the authors choose to represent the *diagonals* of the matrix as 2D texture maps instead of the columns. See figure 5. Lower diagonal elements are appended to upper diagonals such that the length of each 1D vector is N. The 2D textures are padded with zero entries if the vectors do not fill the texture completely. One of the most important characteristic of such a representation is that for matrix-vector multiplication, the result can be calculated $in\text{-}place$ and does not need further post-processing. Also, this layout is a natural representation for banded diagonal matrices, which occur most frequently in numerical simulation techniques. However, it should be noted that for a dense matrix, a columnwise splitting into 2D textures can also serve the purpose, which results in a more efficient matrix-vector multiplication. In the new layout, a matrix transpose operation is now reduced to merely reordering the diagonal rows. The result of each operation is bound to a 2D texture in graphics memory, and can serve as an input for further operations, especially in cases where we want to perform operations of the type ($Ax\ op\ y$), where A is a matrix, x and y are vectors, and op is an operation like sum, $product$ or $subtract$.

A matrix vector operation is now transformed into multiplication of each diagonal with a vector. This operation can be performed in multiple passes. The main diagonal of the matrix is multiplied with the vector to obtain the first partial product. For each subsequent diagonal, the multiplier is shifted by one and the same operation is carried out. This process is illustrated in figures 6 and 7. The authors also implement a reduction operator which is similar to the parallel reduction seen in the previous section, but now uses the *page-stripped* approach that we have described earlier.

For representing sparse matrices, vertex arrays are used to store exactly those locations which have non-zero values. The matrix value associated with that location is stored as

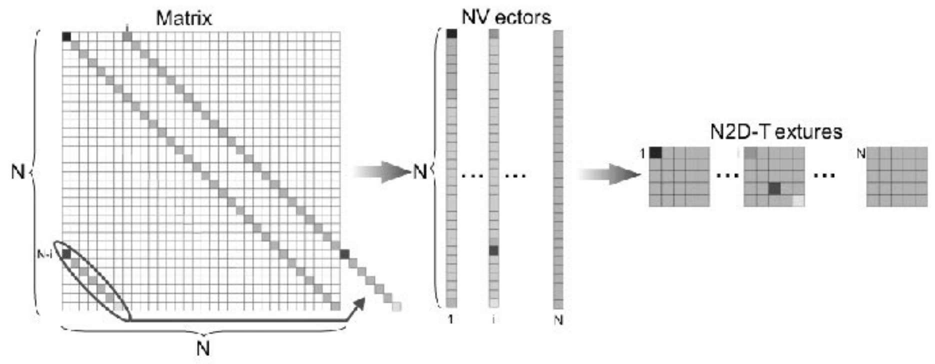


Figure 5: Diagonal matrix representation on a GPU. The diagonal entries are mapped to a 1D vector, and then, to a 2D Texture. *Figure courtesy Jens Krüger.*

colors associated with this vertex value. Multiplication is carried out between the color values and the vector b, and the result is stored into an output texture, from where it can be accessed for the next iteration.

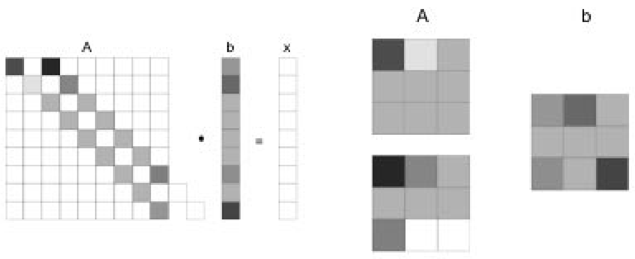


Figure 6: Matrix multiplication in two passes: The layout of the main diagonal and the second upper diagonal of the matrix. On the right, we can see the vectors packed as 2D textures. *Figure courtesy Jens Krüger.*

An important feature of this implementation is that the user is not aware of the different packing layouts for dense, banded or sparse matrices. User application interfaces with high-level C++ classes which encapsulate the lower level details. The authors demonstrate the efficiency of their method with the help of a realtime simulation of 2D water surfaces, which can be found on their website[1].

[1]See http://wwwcg.in.tum.de/Research/Publications/LinAlg

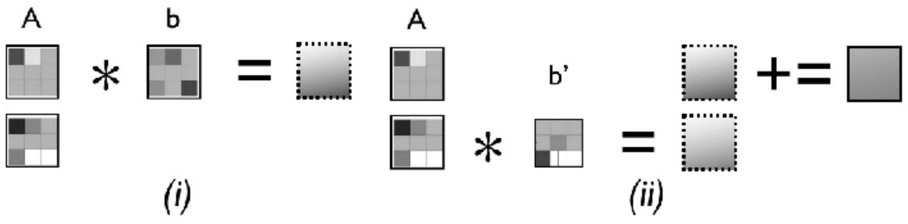


Figure 7: Matrix multiplication in two passes: In the first pass (i), the vector *b* is multiplied with the main diagonal. For the subsequent passes, the 1D vector b is shifted by the distance of the current column from the main diagonal (in this case, 2) and multiplied with the corresponding column (ii). The result is stored in the output texture itself (in-place). Dotted lines indicate that the values in the texture are a partial result.

6 Concluding Remarks

The GPU is indeed a very useful tool for numerical simulation and scientific computing. Recent projects have gone so far as to use a cluster of GPUs to perform numerically intensive simulations [FQKYS04]. The GPGPU website (www.gpgpu.org) maintains the state-of-the-art information about leading edge GPGPU methods. Dominik Göddeke's website[2] provides an excellent tutorial. With the introduction of multi-core CPUs, there has been a definite increase in available computing power for simulations, but the GPU far outperforms traditional CPUs in terms of GFLOPS that it can deliver. A GPU is multi-threaded in hardware, without the overhead of context switching or interrupts. In addition, from the user perspective, the GPU provides a more elegant platform for writing parallel code, by defining a fragment program per computational kernel. Hence, we feel that for achieving true parallelism, the GPU is most certainly a more prudent choice considering present hardware. However, the GPU is limited in memory, and the 1GB graphics cards are as expensive as the high-end CPUs, which do not make them an attractive investment. Technologies supporting multiple graphics cards have arrived in production systems, but the cards do not support a unified memory architecture. With a growing user base and technological support in terms of hardware and software [BFH$^+$04, LSK$^+$06], GPGPU is fast maturing into a discipline in itself. In this paper, we have demonstrated two simple applications which employ GPGPU techniques. We hope that the reader has gained sufficient insight to explore this emerging field and also contribute to its growth.

7 Acknowledgements

The author wishes to thank Dr. Gerald Farin (ASU) and Dr. Hans Hagen (TU Kaiserslautern) for their support in preparing this paper.

[2]See http://www.mathematik.uni-dortmund.de/~goeddeke/gpgpu/tutorial.html for details

References

[AMD06] AMD. AMD Athlon 64 X2 Dual-Core Processor Model Number and Feature Comparisons, 2006.

[ATI06] ATI. ATI(R) Radeon(R) X1900 Technical Specifications, 2006.

[BFGS03] Jeff Bolz, Ian Farmer, Eitan Grinspun, and Peter Schröder. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. *ACM Trans. Graph.*, 22(3):917–924, 2003.

[BFH+04] Ian Buck, Tim Foley, Daniel Horn, Jeremy Sugerman, Kayvon Fatahalian, Mike Houston, and Pat Hanrahan. Brook for GPUs: stream computing on graphics hardware. *ACM Trans. Graph.*, 23(3):777–786, 2004.

[Ble90] Guy E. Blelloch. *Vector models for data-parallel computing*. MIT Press, Cambridge, MA, USA, 1990.

[FQKYS04] Zhe Fan, Feng Qiu, Arie Kaufman, and Suzanne Yoakum-Stover. GPU Cluster for High Performance Computing. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 47, Washington, DC, USA, 2004. IEEE Computer Society.

[GLW+04] Naga K. Govindaraju, Brandon Lloyd, Wei Wang, Ming Lin, and Dinesh Manocha. Fast computation of database operations using graphics processors. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 215–226, New York, NY, USA, 2004. ACM Press.

[HMG03] Karl E. Hillesland, Sergey Molinov, and Radek Grzeszczuk. Nonlinear optimization framework for image-based modeling on programmable graphics hardware. *ACM Trans. Graph.*, 22(3):925–934, 2003.

[KEHKL+99] III Kenneth E. Hoff, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[KW03] Jens Krüger and Rüdiger Westermann. Linear algebra operators for GPU implementation of numerical algorithms. *ACM Trans. Graph.*, 22(3):908–916, 2003.

[LHK+04] David Luebke, Mark Harris, Jens Krüger, Tim Purcell, Naga Govindaraju, Ian Buck, Cliff Woolley, and Aaron Lefohn. GPGPU: general purpose computation on graphics hardware. In *GRAPH '04: Proceedings of the conference on SIGGRAPH 2004 course notes*, page 33, New York, NY, USA, 2004. ACM Press.

[LSK+06] Aaron E. Lefohn, Shubhabrata Sengupta, Joe Kniss, Robert Strzodka, and John D. Owens. Glift: Generic, efficient, random-access GPU data structures. *ACM Trans. Graph.*, 25(1):60–99, 2006.

[NVI06] NVIDIA. NVIDIA(R) GeForce(R) 7950 GX2 Technical Specifications, 2006.

[Owe05] John Owens. *GPU Gems 2*, chapter Streaming Architectures and Technology Trends, pages 15–28. Addison Wesley, 2005.

[THO02] Chris J. Thompson, Sahngyun Hahn, and Mark Oskin. Using Modern Graphics Architectures for General-Purpose Computing: A Framework and Analysis. *micro*, 00:306, 2002.