Visualization of Large and Unstructured Data Sets

Applications in Geospatial Planning, Modeling and Engineering

Proceedings of IRTG 1131 Workshop
June 10-11, 2011, Kaiserslautern, Germany

Edited by
Christoph Garth
Ariane Middel
Hans Hagen

OASIcs – Vol. 27 – VLUDS'11 www.dagstuhl.de/oasics

Editors

Christoph Garth
Computational Topology Group
University of Kaiserslautern
garth@cs.uni-kl.de

Ariane Middel
Decision Center for a Desert City
Arizona State University
Ariane.Middel@asu.edu

Hans Hagen
Computer Graphics & HCI Group
University of Kaiserslautern
hagen@cs.uni-kl.de

ACM Classification 1998
I.3 Computer Graphics

ISBN 978-3-939897-46-0

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-939897-46-0.

Publication date
October, 2012

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

Digital Object Identifier: 10.4230/OASIcs.VLUDS.2011.i

ISBN 978-3-939897-46-0 **ISSN 2190-6807** **http://www.dagstuhl.de/oasics**

OASIcs – OpenAccess Series in Informatics

OASIcs aims at a suitable publication venue to publish peer-reviewed collections of papers emerging from a scientific event. OASIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 2190-6807

www.dagstuhl.de/oasics

Contents

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
IRTG 1131

Preface

The International Research and 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 Davis, Arizona State University and University of Utah. It is funded by the German research foundation (DFG) under grant DFG GK 1131/2, and is currently in the last of two 4.5-year stages.

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, our research integrates 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.

Until now, the state of the art has centered on the visualization of large and structured or small and unstructured data. Dataset that are both large and unstructured 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 produced either by data from field measurements like environmental observation stations, from experiments, and from simulation.

For example, 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, scaled-down earthquake laboratory experiments within a centrifuge improved sensor technology permit the measurement of an increased number of participants 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.

In this issue of OASIcs – OpenAccess Series in Informatics we present the results of the annual workshop of this IRTG held in Kaiserslautern on June 10–11, 2011. The aim of the workshop was to bring together all project partners, PhD students and advisors to report on the different research projects. After two days of presentations and discussions the graduates spent their time on writing papers that cover the outcome of the program and give surveys on related topics.

Kaiserslautern, April 2012

Christoph Garth
Ariane Middel
Hans Hagen

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
IRTG 1131

List of Authors

Jan C. Aurich
Institute for Manufacturing Technology and Production Systems (FBK)
University of Kaiserslautern
Gottlieb-Daimler-Straße 47
D-67663 Kaiserslautern, Germany
aurich@cpk.uni-kl.de

Hans-Jörg Bart
Chair of Separation Science and Technology
University of Kaiserslautern
Gottlieb-Daimler-Straße
D-67663 Kaiserslautern, Germany
bart@mv.uni-kl.de

Henning Barthel
Fraunhofer IESE
Fraunhofer-Platz 1
D-67663 Kaiserslautern, Germany
henning.barthel@iese.fraunhofer.de

Jens Bauer
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
j_bauer@cs.uni-kl.de

Christoph Bender
HTWG Konstanz
Computer Graphics Lab
Brauneggerstr. 55
D-78462 Konstanz, Germany
chbender@htwg-konstanz.de

Daniel Cernea
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
cernea@cs.uni-kl.de

Fang Chen
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
chen@informatik.uni-kl.de

Klaus Denker
HTWG Konstanz
Computer Graphics Lab
Brauneggerstr. 55
D-78462 Konstanz, Germany
kdenker@htwg-konstanz.de

Achim Ebert
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
ebert@cs.uni-kl.de

Daniel Engel
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
d_engel@cs.uni-kl.de

Markus Friedrich
HTWG Konstanz
Computer Graphics Lab
Brauneggerstr. 55
D-78462 Konstanz, Germany
mafriedr@htwg-konstanz.de

Christoph Garth
University of Kaiserslautern
Computational Topology Group
Postfach 3049
D-67653 Kaiserslautern, Germany
garth@cs.uni-kl.de

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Andreas Gerndt
German Aerospace Center (DLR)
Simulations- und Softwaretechnik
Lilienthalplatz 7
D-38108 Braunschweig, Germany
andreas.gerndt@dlr.de

Andreas Kerren
Linnaeus University
Computer Science Department
ISOVIS Group
Vejdes Plats 7
SE-35195 Växjö, Sweden
andreas.kerren@lnu.se

Bernd Hamann
Institute for Data Analysis and Visualization
Department of Computer Science
University of California, Davis
Davis, CA 95616-8562, USA
hamann@cs.ucdavis.edu

Taimur Khan
University of Kaiserslautern
Computer Graphics and HCI Group
Postach 3049
D-67653 Kaiserslautern, Germany
tkhan@informatik.uni-kl.de

Kai Hirt
HTWG Konstanz
Computer Graphics Lab
Brauneggerstr. 55
D-78462 Konstanz, Germany
kahirt@htwg-konstanz.de

Peter Liggesmeyer
University of Kaiserslautern
Software Engineering: Dependability Group
Postfach 3049
D-67653 Kaiserslautern, Germany
liggesmeyer@cs.uni-kl.de

Mark W. Hlawitschka
University of Kaiserslautern
Chair of Separation Science and Technology
Gottlieb-Daimler-Straße
D-67663 Kaiserslautern, Germany
mark.hlawitschka@mv.uni-kl.de

Yarden Livnat
Scientific Computing and Imaging Institute
University of Utah
72 S. Central Campus Drive
Salt Lake City, UT 84112, USA
yarden@sci.utah.edu

Lars Hüttenberger
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
l_huette@cs.uni-kl.de

Harald Obermaier
Institute for Data Analysis and Visualization
Department of Computer Science
University of California, Davis
Davis, CA 95616-8562, USA
hobermaier@ucdavis.edu

Kenneth I. Joy
Institute for Data Analysis and Visualization
Department of Computer Science
University of California, Davis
Davis, CA 95616-8562, USA
joy@cs.ucdavis.edu

Peter-Scott Olech
University of Kaiserslautern
Computer Graphics & HCI Group
Postfach 3049
67653 Kaiserslautern, Germany
olech@cs.uni-kl.de

Patric Keller
University of Kaiserslautern
Software Engineering: Dependability Group
Postfach 3049
D-67653 Kaiserslautern, Germany
pkeller@cs.uni-kl.de

Gerd Reis
Augmented Vision Group
DFKI Gmbh
Trippstadter Straße 122
D-67663 Kaiserslautern, Germany
Gerd.Reis@dfki.de

Peter Salz
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
salz@rhrk.uni-kl.de

Simon Schröder
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
simon.schroeder@itwm.fraunhofer.de

Didier Stricker
Augmented Vision Group
DFKI Gmbh
Trippstadter Straße 122
D-67663 Kaiserslautern, Germany
Didier.Stricker@dfki.de

Raga'ad M. Tarawaneh
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
tarawneh@cs.uni-kl.de

Sebastian Thelen
University of Kaiserslautern
Computer Graphics and HCI Group
Postfach 3049
D-67653 Kaiserslautern, Germany
thelen@cs.uni-kl.de

Georg Umlauf
HTWG Konstanz
Computer Graphics Lab
Brauneggerstr. 55
D-78462 Konstanz, Germany
umlauf@htwg-konstanz.de

Rolf Westerteiger
German Aerospace Center (DLR)
Simulations- und Softwaretechnik
Lilienthalplatz 7
D-38108 Braunschweig, Germany
rolf.westerteiger@dlr.de

Xiang Yang
University of Kaiserslautern
Institute for Manufacturing Technology and
Production Systems (FBK)
Gottlieb-Daimler-Straße 47
D-67663 Kaiserslautern, Germany
yang@cpk.uni-kl.de

Yi Yang
Software Engineering: Dependability Group
University of Kaiserslautern
Postfach 3049
D-67653 Kaiserslautern, Germany
yang@cs.uni-kl.de

Virtual Reality supported Visualization and Evaluation of Noise Levels in Manufacturing Environments*

Xiang Yang[1], Bernd Hamann[2], and Jan C. Aurich[1]

1 **Institute for Manufacturing Technology and Production Systems (FBK) University of Kaiserslautern, Germany**
yang@cpk.uni-kl.de

2 **Institute for Data Analysis and Visualization & CS Department University of California, Davis, USA** hamann@cs.ucdavis.edu

Abstract

Virtual Reality (VR) provides users advanced visualization and interaction technology for designing, analyzing and exploring complex data. To address the issue of noise in manufacturing environments, we developed a VR-supported method allowing users to explore noise behavior. This method consists of an implementation of acoustic simulation and visualization for both desktop and Cave Automatic Virtual Environment (CAVE) based VR systems. It enables user-oriented, interactive analysis of simulated data, where there capability to immerse oneself in the data is especially valuable. In a real-world factory, the acoustic measurements obtained essential input data for simulation settings and validation data for simulation results. Furthermore, some political and legal aspects are addressed to enhance the evaluation of results and the visualization. By using the implemented software tool, users are able to understand and investigate the noise issue in manufacturing straightforwardly.

1998 ACM Subject Classification J.6 Computer-Aided Engineering

Keywords and phrases virtual reality, acoustic simulation, visualization, manufacturing

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.1

1 Introduction

1.1 Virtual Reality

The Virtual Reality (VR) is applied in this paper to support the investigation of noise issues in manufacturing industry. With a virtual environment, the acoustic simulation and visualization are implemented. The simulation results and enhanced analysis capabilities in VR provide a new point of view to fulfill this special requirement during factory planning. As a comprehensive and widely developed technology, VR is originally defined as: „a system that can display information to all senses of the user with an equal or bigger resolution than the one that can be achieved in a natural way so that the user cannot say that the artificial world is not real" [24].

In recent years the VR technology is improved significantly by increasing demand of industrial applications and shows following changes, such as increased hardware power, reduced acquisition costs, integrated methods with simulation and visualization. And on

* This work was partially supported by Eduard Deines, Simon Schröder and Tim Biedert.

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 1–12

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

implementation level, the Virtual Reality Modeling Language (VRML)/Extensible 3D (X3D) standard is widely used and further developed.

The VRML standard is originally developed as a modeling language for web applications and later accepted by the International Organization for Standardization (ISO) [2]. It is implemented for different applications, e.g. virtual assembly, machining processes simulation, visualization of facilities, and employee training parallel to the running production [17, 10]. Some web-based applications are also found in literature review. For example, Qiu et al. demonstrate an implementation of automation animation using VRML and identified fundamentally their advantages like assembling CAD-objects and their interaction [19]. And remarkable of Ranga and Gramoll [20] are the introduction and implementation of JavaScript into VRML and the performance in 3D FE-analysis. They realized a web-based virtual environment and included a customized user interface. As successor to VRML, X3D is also standardized by ISO and contains mostly all VRML features. For current applications X3D has no remarkable advantages comparing VRML. On the other hand, VRML has more developed Nodes, APIs and extensions. With Java or JavaScript, different programming solutions like mathematical descriptions can be implemented in VRML. The Nodes offer a customized link to different positioning and allocations between different objects. Therefore, VRML is used as modeling language and scene graph standard in this paper. Due to the internet nature of VRML, in this paper a web-server based concept is developed.

According to different user immersion levels, VR systems are classified into non-immersive and immersive systems. A non-immersive VR system, such as a desktop-based display system, is according to some VR definitions not understood as a real VR system. The full-immersive VR systems, such as the CAVE system, provide most costly and complex solution with unique benefits. Compared with non-immersive system an immersive VR system has higher sense of situational awareness, wider field of view, higher scale perception and sense of immersion. However, a non-immersive VR has advantages of lower costs, shorter development time, and better implementation conditions. For example, the desktop-based VRML viewer enables users implementing and viewing the developed virtual environment with simple configuration.

At the Institute for Manufacturing Technology and Production Systems (FBK), we use the both VR systems to satisfy different demands of research and industrial projects [8, 9]. The proposed concept in this paper is also first developed and tested in a desktop-based environment and further in CAVE system. The implementation and visualization in both systems are to be discussed in latter sections.

1.2 Noise in Manufacturing

The factory workers are exposed to any of occupational hazards every day, such as, chemical solvent, heat, noise, vibration, etc. Noise is becoming one of the most frequent occupational hazards in manufacturing. The noise in a factory could from machinery, powered tools or other activities, which influences employees' health and can even cause diseases. According to DIN 1320 and VDI 99, the noise is described as unwanted sound causing disorder, harassment and other health problems. As the noise exceeds specific limits, the risks of hearing loss or other sicknesses are increasing. According to the investigation of Federal Institute for Occupational Safety and Health (BAUA), about five million employees are exposed to noise in Germany. And this number in USA 1991 was over six million [12]. To protect health and safety of the employees, there are existing laws and guidelines to follow, e.g. the Federal Ministry of Labour and Social Affairs (BMAS) limits noise and vibration levels within Germany's Occupational Safety Law "Arbeitssicherheitsgesetz" (ASiG), German ordinance "Lärm- und Vibrations-Arbeitsschutzverordnung" (LärmVibrationsArbSchV) and

other additional guidelines. In the European Union the minimum requirements to protect the workers from noise is determined with directive 2002/44/EU.

According to the influence effects, the impacts of noise to the workers in manufacturing systems are classified from three main points of view: occupational safety, negative health impacts, and preservation of work performance.

- The occupational safety: According to regulation ISO 7731 [5], the loudness of a warning signal in industry must be 15 dB over the ambient noise. The warning signal will not be effective if the factory is too noisy. Also, communication among workers deteriorates [7] due to noisy working environment, which causes potential unsafe factors.
- The negative health impacts: the health damages can result from exposure to repeated or very loud noise at work, e.g. permanent hearing loss and heart disease. More noise related health risks such as physiological effects or the risk of accidents are listed by [26, 27, 14].
- The preservation of work performance: much research work has been done to determine the influence of noise on workers' performance. Different authors found a significantly poorer performance when employees are exposed to noise [22, 11, 16].

The laws and guidelines require the employers to worry about the noise, namely eliminate or reduce the risks from exposure to noise. The specific duties on employers are also placed when the average sound pressure level and peak sound pressure level reach certain limits. Different methods and instruments are deployed in practice [21]. According to the solution methods, these methods are classified into three categories [18]: reduction of noise emission, reduction of sound propagation, and reduction of sound pollution. In manufacturing, the latter two methods are often implemented. Important methods to reduce the sound propagation are changing the room shape, optimizing division of work areas, and using the sound absorbing building structures as well as materials. The methods to reduce the sound pollution are workstation related, such as arrangement of workstations and sound shielding around workstations.

1.3 Acoustic Simulation

Acoustic simulation enables the investigation of the behavior of sound propagation, which determines the influences of noise. Nowadays, many established methods for sound simulation and sound visualization exist, which are widely implemented for computer games, multi-sensory user interface, or acoustic prototyping. They are mostly wave-based or geometric methods. Such as the Finite Element Methods (FEM), Boundary Element Methods (BEM), and Finite Difference Time Domain (FDTD) are widely discussed and compared in [25, 23].

A geometric approach called Phonon Mapping is also developed in [13], which is implemented to achieve the object in this paper. Analogous to seeing light as particles called photons sound sources emit sound particles called phonons. With each reflection the phonon's sound pressure is decreased according to the material's absorption coefficient. At each reflection position the current sound pressure of the phonon is stored. Calculating the influence of all phonons to a particular listener position the information of all reflection positions is collected and weighted according to their distance to the listener. Tracing the pressure for different frequency bands and using a Dirac impulse as sound source this calculation provides the room's impulse response. Convoluting an anechoic signal with this we get the exact sound in the simulated room with specific source and listener positions. Furthermore in [13] acoustic properties of simulated rooms are extracted like understandability of speech or suitability for concerts.

In addition to room acoustic methods, an outdoor acoustic simulation and visualization method is developed in [15]. It enables the industry to make the assessment of industrial noise in the neighborhood. However, the outdoor noise issue is not concerned in this paper.

Adjusting or rescheduling a manufacturing system is expensive and time consuming. It is necessary to take the noise issue into account both during the planning and design stage of the manufacturing. In this paper, the simulation is used to calculate the sound level at arbitrary listener positions. Then the simulation and visualization are implemented in a virtual environment using VR, which enhances the understanding of noise influence in a factory.

The rest of this paper is organized as follows. Section 2 gives an overview of the implementation concept and involved tasks.The experimental acoustic measurements are shown in section 3. In section 4, an interactive sound simulation and visualization approach is presented. To implement this approach in VR, a web/client structure is introduced. Further, the simulation results is visualized using a CAVE system. The final section concludes the paper and gives an outlook of future research.

2 Method and Workflow

The VR-based concept presented in this paper includes several subdivided objectives. They are design and implementation of acoustic measurements, an investigation of legal regulations, 3D modeling of a virtual environment, modeling of a sound source, parameter setting and simulation implementation, design of a graphic user interface (GUI), development of a web server as interface between VR and simulation etc. Figure 1 shows these tasks within a basic workflow.

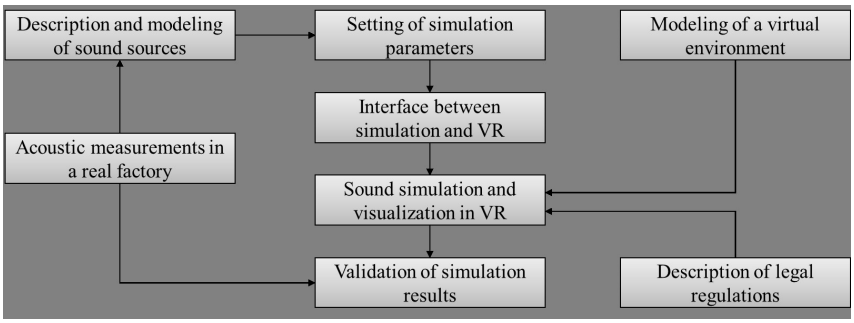


Figure 1 Workflow of a VR-based Method.

An acoustic measurement consists of sound level measurement and sound intensity measurement, which provides input and validation data for simulation. Before simulation start the parameters are specified according to the acknowledgement from measurement. However, the validation part will not be included in this paper due to current research status. At the same time, a mechanical laboratory is rebuilt into a 3D model, which provides a virtual environment for simulation and visualization. In this process several modeling software are used, such as SolidWorks and 3ds Max. An interface is made to enable users to change the simulation settings. An additional interactive 3D user interface is created to navigate and to control the simulations in a virtual environment. Combined with investigation of legal regulations for manufacturing, the visualization of simulation results facilitates further the analysis of noise issue in a factory.

Several requirements are considered, such as, the efficient and safe data transfer and file

converting, a straightforward user interface, and extendable software structure with function modules. In this paper not all tasks are discussed, the focuses are acoustic measurements, acoustic simulation and visualization, and the implementation in VR. These three main issues ensure the usability and efficiency of intended approach. The design and implementation of both sound pressure measurement and sound power measurement are firstly shown in the next section.

3 Acoustic Measurement

The acoustic measurements are classified into sound pressure level measuring and sound power measuring. The sound pressure level indicates the effective sound pressure relative to a reference value and the sound power shows the total energy of a sound wave per unit time and measured in watts. The former allows a sound source description which is input data for the simulation, and the latter provides the data basis for simulation result validation. The measurement environment is a mechanical laboratory at the University of Kaiserslautern. In this laboratory, the employees usually work with two manually operated lathes, a CNC (Computerized Numerical Control) cutting machine, a CNC tool grinding machine, a universal milling, a CNC drilling center, and some other small equipment. The layout of this laboratory is shown latter in Figure 3a.

3.1 Sound pressure level measurement

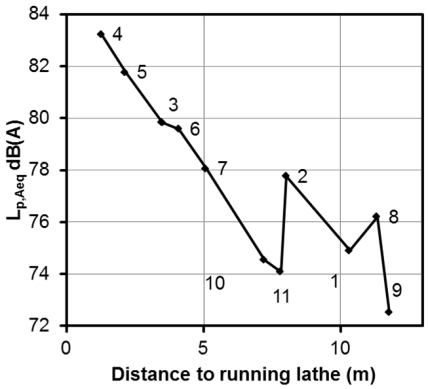
(a) Measuring parameter setting **(b)** Microphone calibration **(c)** Measurement configuration

Figure 2 Sound pressure level measurement.

Sound pressure is estimated by the difference between local pressure and atmospheric pressure caused by a sound wave. This index is often used to identify the impact of sound to the human. The sound pressure level is measured by using the sound level meter. A portable computer allows users to make measurement settings, to store the measurement results, and to view visualized results (shown in Figure 2a). According to regulation ISO 11202 [1], the acoustic pressure level is described by continuous sound pressure level, sound from foreign sources, and maximal sound pressure level. A combined measurement design is based on DIN series 45635, in which the measuring requirements for different manufacturing processes are defined. For example, the measurement for a lathe is based on DIN 43635-1 [3]. A calibration shown in Figure 2b is done before measuring. For each single sound source

the measurements are made at 11 different positions (shown in Figure 3a) and at 3 typical machining procedures: idling, normal machining and high speed machining. During the measuring, the microphone is positioned using a tripod with extension arm. The microphone is adjusted 10 cm behind the head position at a height of 1.55m. In Figure 2c, the microphone is for example placed facing the control panel.

(a) Measuring points in a real factory



(b) Measurement results

Figure 3 Position-based sound pressure level measurement.

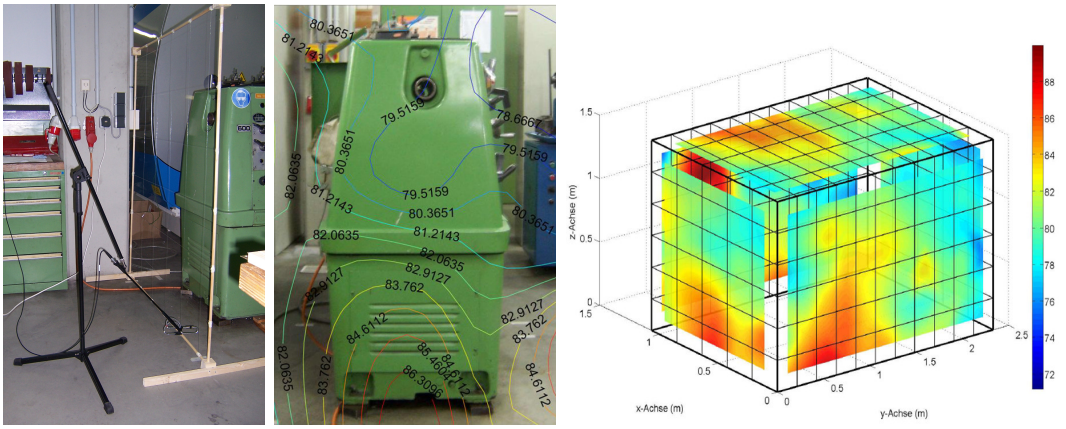
Figure 3b shows a measurement example of a running lathe (machine "DB1" in Figure 3a) at normal machining speed. Position 4 is directly in front of the lathe and shows the highest continuous sound pressure level. The sound pressure levels at different positions are basically proportional to the distance to the running lathe. Using the same analysis method, the measured maximal sound pressure level is summarized as well. The sound from foreign sources has no essential influence in this case and therefore not investigated. A validation between these results and simulation will be made in the future.

3.2 Sound power measurement

A sound power investigation enables the description of a sound source. To describe the sound power, the sound intensity description is usually implemented. In [4] the measuring equipment is defined as intensity probe. Prior to measuring, a calibration of the sound intensity probe is necessary, which is implemented by using a piston phone.

The intensity probe is placed 0.5m from the measured machine surface. The average during 30 seconds measuring time is determined as measurement result. Based on ISO 9614-1[6], it is necessary to repeat the measurement at least once per square meter. Therefore, a grid is used to fix the measurement points on the side of a machine (shown in Figure 4a). This grid is made by several wood frames and thin cords, which defines the sub-surfaces for measuring and helps the operator to find the right position to place the intensity probe.

The intensity measurement is repeated in the middle of each sub-surface for all five machine sides, except the bottom side. Based on the measurement data, a visualization of power distribution around the sound source is made, which facilitates the understanding of sound propagation in the vicinity of the sound source (Figure 4b and 4c) and further enhance the modeling of sound source in a virtual environment.



(a) Measurement setting **(b)** Power distribution **(c)** 3D visualized sound power distribution

Figure 4 Visualization of sound intensity distribution.

4 Sound Simulation and Visualization in VR

4.1 Server/Client structure

The objective of in section 2 discussed concept is to have an application for interactive noise investigation in a virtual environment. To do this, VRML is used as a front-end to users. It provides capabilities to render complex scene graph, to animate object motion, or to interact with the user. VRML a well established standard which supported by any important platform and operating system. Although VRML provides programming interfaces through JavaScript and Java. As a server/client concept is used, an existing C++ implementation of the acoustic simulation can be used and extended for the purposes. Figure 5 shows the method for interactive acoustic simulation and visualization. The geometric data is transferred firstly from modeling software to VRML editor, and then loaded into a web server generated viewer application. All the objects are built with 3ds Max and then exported using the VRML standard. Using VrmlPad as VRML editor, the Sensors, Events or other interaction Nodes are constructed in a VRML file. At the same time, Java and JavaScript descriptions are embedded into VRML directly. It enables interactive user interface, simulation launching, and data converting. After these steps, the data for the application is prepared. The simulation application acts as a server, which loads the model of a virtual factory and adds user interface elements. The resulted VRML code is delivered to VRML compliant VR platform via HTTP.

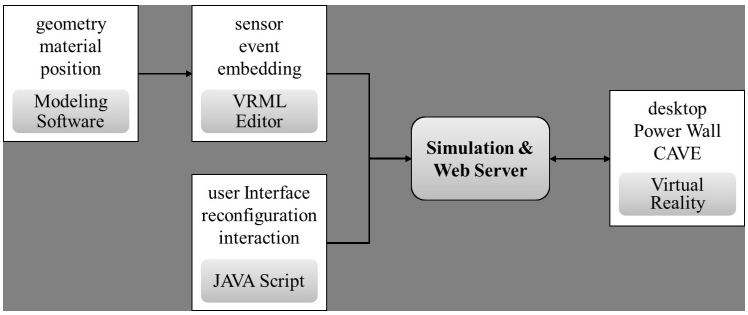


Figure 5 A web server method for acoustic simulation and visualization.

4.2 Simulation using web server

As the simulation is started, the server loads the VR model (VRML file) and adds additional VRML script describing basic user interface elements for interaction within the scene. Buttons and sliders are implemented using simple VRML geometric codes and JavaScript. This geometry is connected to a TouchSensor for buttons and to a PlaneSensor for sliders. These sensors emit events which are routed to Script Nodes containing simple interaction logic written with JavaScript. Commands are sent back then to the server by loading a special URL which encodes the action. The server computers the simulation and delivers the data to the VRML viewer again. These communications are done via HTTP connections. The viewer opens a new connection using an HTTP request asking for a file encoding commands in the filename. The server does its calculation and answers with a new VRML file delivered by this existing HTTP connection.

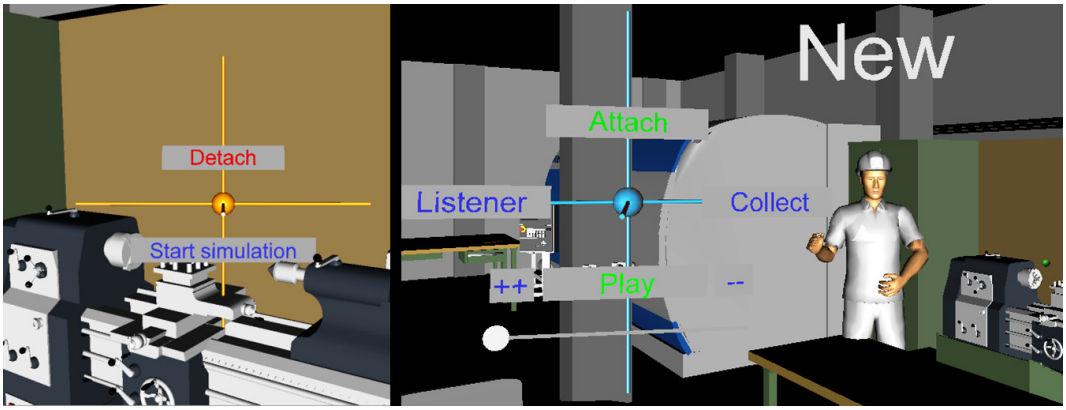


Figure 6 Simulation user interface.

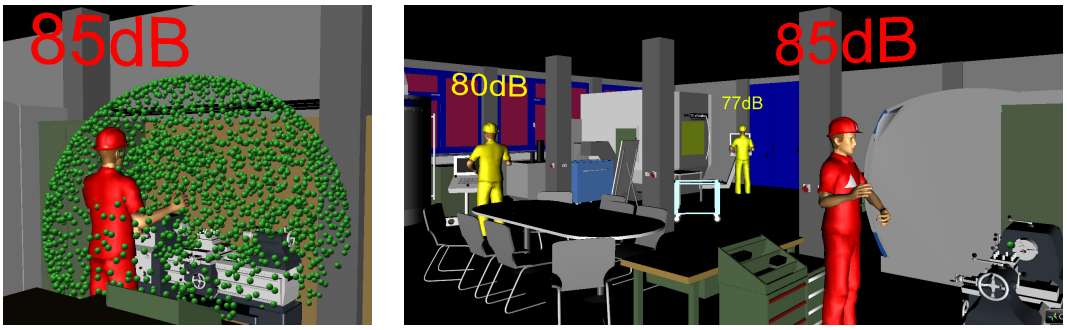
To implement the phonon tracing simulation, several input parameters are required. It is a geometric approach, the algorithm needs first the geometric model of the room and the objects inside. Each surface of the objects is assigned to one material with specific sound absorption property. Usually, the material's absorption coefficients are taken from the coefficient tables.

Then the position of sound source and the sound pressure at 1 meter distance to the sound source are required. The sound pressure can be estimated by using the measurement data. Better results are achieved by providing an anechoic signal of the sound source, i.e. the sound pressure at several frequencies. Furthermore, the user may define an arbitrary number of listener positions. Other settings such as the number of phonons, the speed of sound wave etc. are also made before simulation. For more details on the simulation and input settings, refer to [13].

After parameter settings, the model is loaded into the viewer application. The user can explore the room and place the sound source within the viewer application and starting the acoustic simulation. In Figure 6, two modules are shown. On the left is the module for sound source setting and simulation start. Using the right module, one or more listeners are placed in the room according to common operation positions, and then the phonon collection step can be performed. In 1.3, the simulation algorithm is already described.

4.3 Visualization in VR

When the simulation is done, users can investigate the sound propagation inside the room by view the animated phonon paths. In Figure 7a, the phonons are visualized as green spheres. A 3D user interface facilitates the user experience. The playback speed can be adjusted with the "++" and "–" buttons and the current simulation time step can be specified with a slider. The tool enables the user to place and remove listeners depicted by workers interactively. Then, a command for the calculation of the sound pressure levels at the defined listener positions is issued. The scene graph in Figure 7b is updated with corresponding listener colors according to the sound pressure level: green for low sound pressure levels <80dB, yellow/orange for critical sound pressure levels <84dB, and red when the sound pressure level is higher than the upper action value according to German ordinance LärmVibrationsArbSchV. The visualization of the sound propagation and noise level helps users to identify possible noise problems/positions and find out the matching solutions. The effect of different materials on the spectral energy/pressure distribution can be observed as well. Scene materials of high reflectance can be identified and replaced in the virtual environment. The potential improvements of the changes can be viewed directly after one more simulation, thereby reducing the noise level.



(a) Sound propagation **(b)** Color scaled sound level

Figure 7 Visualization of sound level.

After the successful test with desktop system, the application is implemented in CAVE-based VR. The CAVE system located at the FBK institute Kaiserslautern consists of 8 projectors, projecting on 4 walls (front, left, right and floor). It offers an immersive virtual environment more than 17 cubic meters. In this system, the passive stereo technology with circular polarization is used for stereoscopic rendering of the 3D scene. The CAVE is driven by a VR cluster, which contains 1 master and 8 clients. Four infrared (IR) cameras from are used for position/motion tracking. Users interact with the VR system via different input devices such as a fly stick. "COVISE" is selected as software platform to operate the CAVE, due to its wide range of hardware support and the variety of different functionality modules. For wide industrial applications, COVISE enables straightforward integration of different modules as well as visualization functionalities.

The web server opens a HTTP port for COVISE modules and gets feedback from COVISE. The two necessary COVISE modules are "VRML renderer" and "VR" modules. The former enables the VRML visualization of a scene graph in CAVE and the latter provides basic interaction functions, such as the navigation and user tracking. In Figure 8, the CAVE-based application is shown. The user explores the virtual factory and simulates the sound propagation to analyze the noise issue during workstation or layout planning.

(a) IR tracking system **(b)** Sound pressure level visualization in CAVE

Figure 8 Interactive visualization of sound level using CAVE.

5 Conclusion and Future Work

In this paper, a new method for interactive and fast noise investigation in virtual manufacturing is introduced. A software tool is implemented, which can be used to determine the noise level at operation positions in a factory and test different improvement scenarios in VR directly. A web server is used as data interface and user interface between simulation and VR. Users are allowed to set the simulation parameters and view the results through an interactive user interface. Further, the method provides facilities to check the limits of sound pressure level against federal laws and regulations. The proposed concept is effective and produces realistic results.

However, the industrial noise in outdoor area is not considered in this concept, which is the further research focus. Combining the wave-based simulation algorithm can enhance the phonon tracing results. However the integration of both methods needs high performance computing and optimized software structure. Also, this tool provides only basic human-computer-interaction facilities the sound investigation, which is to be extended in the future. Changing acoustic properties of the room has to be made in original room model and exported to VRML again. This process has to be improved to an more intuitive process within VR, e.g. the interactive adjustment of absorption coefficient for simulation of different floor materials. And based on the server/client structure, an implementation of web-based collaborative simulation and visualization is also considered in the future.

6 Acknowledgments

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

References

1 ISO 11202. Acoustics – Noise emitted by machinery and equipment – Determination of emission sound pressure levels at a workstation and at other specified positions applying approximate environmental corrections, 2010.

2 ISO/IEC 14772-2. The Virtual Reality Modeling Language International Standard, 2004.

3 DIN 45635-1. Geräuschmessung an Maschinen; Luftschallemission, Hüllflächen-Verfahren; Rahmenverfahren für 3 Genauigkeitsklassen, 1984.

4 DIN EN 61043. Elektroakustik; Geräte für die Messung der Schallintensität; Messung mit Paaren von Druckmikrofonen, 1994.

5 ISO 7731. Ergonomics – Danger signals for public and work areas – auditory danger signals, 2003.

6 DIN EN ISO 9614-1. Akustik – Bestimmung der Schallleistungspegel von Geräuschquellen aus Schallintensitätsmessungen – Teil 1: Messungen an diskreten Punkten, 2009.

7 ISO 9921. Ergonomics – Assessment of speech communication, 2003.

8 Jan C. Aurich, Dirk Ostermayer, and Martin Rößing. Models for vr-based reconfiguration of manufacturing systems – basic demands and requirements. In *Proceedings of the ProSTEP iViP Science Days*, pages 148–157, Darmstadt, Germany, 2005.

9 Jan C. Aurich, Dirk Ostermayer, and Christian Wagenknecht. Improvement of manufacturing processes with virtual reality-based cip workshops. *International Journal of Production Research*, 47:5297–5309, 2009.

10 Christian Brecher and Stephan Witt. Interactive analysis of the structural mechanic behaviour of machine tools. *Production Engineering*, 3, Numbers 4-5:475–481, 2009.

11 Eric Sundstrom; Jerri P. Town; Robert W. Rice; David P. Osborn; Michael Brill. Office noise, satisfaction, and performance. *Environment and Behavior*, 26(2):195–222, 1994.

12 William W. Clark and Paul R. Lambert. *Hearing loss: occupational and non-occupational*, pages 29–31. American Speech-Language-Hearing Association, Washington, D.C., 1991.

13 Eduard Deines. *Acoustic Simulation and Visualization Algorithms*. PhD thesis, University of Kaiserslautern, 2008.

14 Karl D. Kryter. *The handbook of hearing and the effects of noise: physiology, psychology, and public health*. Academic Press, Boston, 1994.

15 Frank Michel. *Simulation and Visualization of In- and Outdoor Sound*. PhD thesis, University of Kaiserslautern, 2008.

16 Jessica K. Ljungberg; Gregory Neely. Stress, subjective experience and cognitive performance during exposure to noise and vibration. *Journal of Environmental Psychology*, Volume 27, Issue 1:44–54, 2007.

17 David Martńez Oliveira, Sandra Castro Cao, Xulio Fernández Hermida, and Fernando Martń Rodríguez. Virtual reality system for industrial training. In *Proceedings of 2007 IEEE International Symposium on Industrial Electronics*, Vigo, Spain, 2007.

18 Wolfgang Probst. *Anwendung der Geräuschemissionsangabe in der Praxis*. Schriftenreihe der Bundesanstalt für Arbeitsschutz und Arbeitsmedizin. Wirtschaftsverl. NW, Verl. für Neue Wiss., Bremerhaven, 2002.

19 Ying Qiu, Weimin Li, and Zhi Wei. The simulation of vrml based manipulator and techniques. In *Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on*, volume 2, pages 949 –953, dec. 2008.

20 Karthik Ranga and Kurt Gramoll. 3-d finite element analysis on the internet using java and vrml. In *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, 2000.

21 Helmut Schmischke. *Lärm am Arbeitsplatz in der Metall-Industrie*. Berufsgenossenschaftliche Informationen für Sicherheit und Gesundheit bei der Arbeit (BGI), 2009.

22 Neil Sherwood and Michael J. Griffin. Effects of whole-body vibration on short-term memory. *Aviation, Space, and Environmental Medicine*, 61(12):1092–1097, 1990.

23 S. Siltanen, T. Lokki, and L. Savioja. Rays or waves? understanding the strengths and weaknesses of computational room acoustics modeling techniques. In *Proc. Int. Symposium on Room Acoustics*, Melbourne, Australia, Aug. 2010.

24 Ivan E. Sutherland. The ultimate display. In *Proceedings of the IFIP Congress*, pages 506–508, 1965.

25 U Peter Svensson. Modelling acoustic spaces for audio virtual reality. *IEEE Benelux Workshop on Model based Processing and Coding of Audio Leuven Belgium*, pages 109–116, 2002.

26 Verein Deutscher Ingenieure (VDI). VDI 2098 Blatt 2 – Assessment of noise with regard to the risk of hearing damages, VDI manual noise reduction, 1988.

27 Verein Deutscher Ingenieure (VDI). VDI 2098 Blatt 3 – Assessment of noise in the working area with regard to specific operations, VDI manual noise reduction, 1999.

Spherical Terrain Rendering using the hierarchical HEALPix grid

Rolf Westerteiger[1,3], Andreas Gerndt[1], and Bernd Hamann[2]

1 **German Aerospace Center (DLR)**
Braunschweig, Germany
rolf.westerteiger@dlr.de, andreas.gerndt@dlr.de
2 **Institute for Data Analysis and Visualization & CS Department**
University of California, Davis, USA
hamann@cs.ucdavis.edu
3 **Computer Graphics and HCI Group,**
University of Kaiserslautern, Germany

Abstract

We present an interactive spherical terrain rendering system employing a hierarchical subdivision of the HEALPix coordinate system using quadtrees. Compared to other parameterizations, the scheme avoids singularities and allows for efficient fusion of mixed-resolution digital elevation models and imagery. A Level-of-Detail heuristic is used to guarantee both high performance and visual fidelity. Unified treatment of DEM and imagery data is achieved by performing the HEALPix projection within a GPU shader. The system is applied to the exploration of Mars, using both MOLA (NASA) and HRSC (German Aerospace Center) data sets.

1998 ACM Subject Classification I.3.6 Methodology and Techniques – Graphics data structures and data types

Keywords and phrases terrain rendering

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.13

1 Introduction

Three-dimensional visualization of terrain is a well-studied problem with a history of algorithmic approaches. However, most of these solutions assume a "flat-earth" model, where topography is mapped to a plane. This is acceptable as long as the viewer is close to the surface but breaks down at distances where planetary curvature becomes relevant. Spherical terrain rendering aims to solve this problem by representing the planetary surface as a spheroid to allow for visualization at any scale.

Naive approaches to spherical terrain rendering parametrize the whole surface using a two-dimensional coordinate system. However, any 2D parametrization of the sphere exhibits so-called coordinate singularities, which lead to visible sampling and/or rendering artifacts. For example, in the canonical geographic coordinates (latitude / longitude), singular points appear at $\pm 90^\circ$ latitude (north and south pole).

To avoid these singularities, a 3D parametrization must be used. A common strategy uses a platonic solid as base geometry which is refined using recursive subdivision and extrusion to the spheroid surface. Each of the faces of the base geometry can be parametrized using 2D coordinates while the index of the face can be interpreted as a third (integer) coordinate. To implement Level-of-Detail (LoD) rendering, multi-resolution data structures are used which assign elevation values to vertices in each subdivision level.

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 13–23

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


These strategies have in common that the relation between parametric and euclidean coordinates is only given implicitly by the subdivision scheme. As euclidean coordinates are required for rendering, this implies that either the recursive subdivision has to be recomputed in every frame or that meshes have to be stored in euclidean coordinates in GPU memory. The former solution is computationally expensive while the latter incurs a high storage cost.

In this paper we propose applying the HEALPix [5] coordinate system to represent planetary topography for spherical terrain rendering. HEALPix decomposes the sphere into 12 curvilinear quadrilateral base patches of equal area. These are then uniformly subdivided as necessary to form a sampling grid for representing data on the sphere. Compared to other spherical rendering approaches which use implicit coordinate systems, the HEALPix projection is given in a closed form. This property allows for efficient on-the-fly projection of height fields from parametric to euclidean space within a vertex shader. To enable LoD rendering a multi-resolution database structure based on a forest of quadtrees is used. A subset of these trees is kept in GPU memory and updated as the viewer moves by loading data in the background (data streaming).

In the following, we present other work related to spherical terrain rendering. Then we describe the data structure used and give an efficient transformation algorithm to convert geological data sets to this storage scheme. The actual rendering algorithm for DEMs which is based on a top-down traversal of this data structure is subsequently described, including aspects of LoD selection, frustum culling, triangulation of tiles and background data streaming. This algorithm is then extended to also support the visualization of high-resolution imagery draped on top of the DEM.

The system is applied to the interactive exploration of Mars, using a hybrid of MOLA (NASA) and HRSC (German Aerospace) data sets. While HRSC is of higher resolution than MOLA, it does not yet provide full coverage of Mars (as of 2011). We therefore chose to integrate both data sets into a single database, demonstrating that the storage scheme can efficiently capture both at their native resolution.

In Section 4 we present our results using these data and give performance measurements to substantiate the interactivity claim. In the final section, some areas of further research are identified, focusing on aspects of performance and image quality.

2 Related work

Geometry Clipmaps [7] is a planar terrain rendering approach using rectangular, concentric rings of geometry centered around the viewer which decrease in resolution with increasing distance. The algorithm exploits temporal coherence in viewer movement by minimizing per-frame data structure updates using toroidal addressing.

Spherical Clipmaps [3] extend this scheme to spherical rendering by representing the planet's hemisphere which faces the viewer using a set of circular rings. However, due to low accuracy of the $\tan^{-1}$ function on the GPU, this method can produce visible cracks in the final triangulation. Furthermore, compared to the original scheme, vertices are no longer centered on actual height field samples. Due to the additional interpolation required, the data is essentially low-pass filtered when rendering.

The Planet-Sized Batched Adaptive Meshes (P-BDAM [2]) system also uses a subdivision of the sphere into a set of curved base patches. These are then further subdivided using adaptive triangulations. Compared to regular grids, adaptive triangulations require larger per-vertex storage costs but on the other hand need less vertices to represent smooth topography. To further reduce storage costs, the authors suggest an efficient packing scheme for per-vertex

data which is unpacked on the GPU during rendering.

Planetary Scale Composition [6] is an interesting approach in which the composition of raster input data is deferred to the rendering stage. A base icosahedron is interactively subdivided and extruded to the sphere surface using a ping-pong buffer scheme on the GPU. When the desired screen resolution has been achieved, subdivision stops and the resulting smooth sphere geometry is displaced by a set of given raster DEMs which are represented using textures. Using this scheme it becomes possible to add new input data on the fly during a running visualization.

Due to numerical instability, the required texture coordinates cannot be computed in a closed form. Instead they are maintained for each vertex as the subdivision progresses. Coordinates for newly generated vertices are interpolated using the haversine geodesic midpoint method. This method is computationally expensive, especially considering that it has to be evaluated once for every vertex in every frame. In our approach data is resampled to the HEALPix grid in an offline preprocessing step. Rendering, however, also requires generation of spherical coordinates which are computed in a numerically stable fashion using the closed HEALPix formulae.

Crusta [1] is a terrain rendering framework which uses a 30-sided polyhedron as base geometry, whose faces are recursively subdivided using a quadtree. Due to the recursive formulation of this geometric construction there is no closed form projection formula between parametric and Euclidean coordinates. This prohibits performing the projection on the GPU and requires computation and storage of Euclidean vertex coordinates for individual tiles at load time. In contrast, our approach only requires storing a single scalar elevation value for each vertex at runtime.

Google Earth is a popular tool for exploring the surface of Earth using data streamed over the internet, which performs well even over low-bandwidth internet connections and provides a high degree of interactivity. However, the system suffers from degenerate triangulations close to the north and south pole, leading to visible artifacts. See Section 4 for a visual comparison with the triangulations generated by our system.

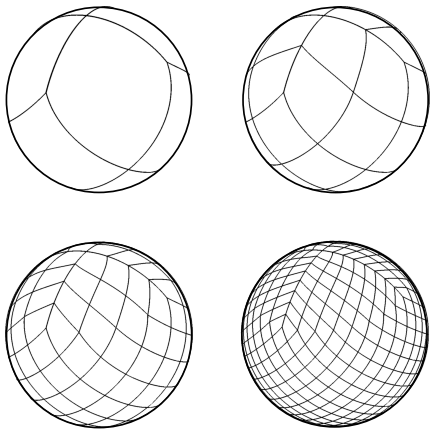


Figure 1 First four levels of the HEALPix sphere tessellation. The root of the hierarchy is formed by 12 curvilinear base patches which are recursively subdivided into quadrants. In our quadtree scheme, each patch corresponds to a 255×255 grid of samples.

The HEALPix [5] scheme is a general solution for representing data on a sphere. It decomposes the sphere into 12 curvilinear quadrilateral patches with associated parametric (u, v) coordinates in $[0, 1]^2$ (see Figure 1). For sampling and data storage, the authors suggest

using a hierarchy of uniform (in parametric coordinates) grids for each patch [5]. Samples are stored in quadtree order to optimize referential locality.

Our application demands a high-fidelity model of Mars, which requires composition of multiple data sets with different coverage and resolution. We therefore chose an explicit quadtree data structure to enable representation of sparse and mixed-resolution data.

3 Approach

To represent terrain we use a database consisting of 12 quadtrees, one for each HEALPix base patch. Each tree node stores a tile of 255×255 samples to improve batching in rendering, to reduce management overhead and to increase referential locality during disk I/O while still allowing for 16 bit vertex indices to be used in the representation of the triangle mesh for rasterization.

While the leaf nodes of the tree contain data resampled at the native input resolution, the inner nodes store subsampled representations of their children, forming a multi-resolution data structure suitable for LoD-rendering (see 3.2). Neighboring tiles overlap by one sample at their shared boundary, which incurs a small storage overhead but allows for gapless C_0 continuous rendering without needing to reference neighboring data.

3.1 Data resampling

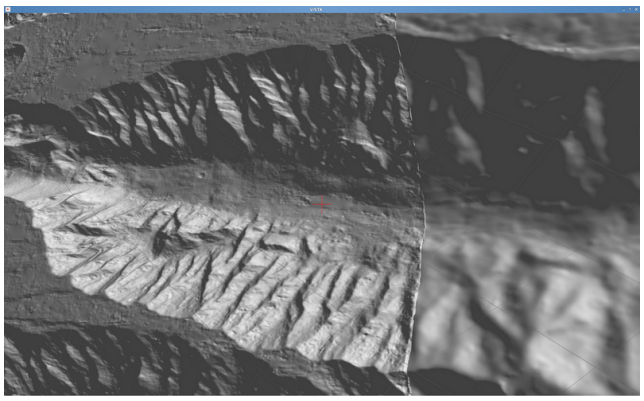


Figure 2 Embedding of high-res into low-res DEM (left: HRSC, right: MOLA).

Digital Elevation Models (DEMs) are frequently represented as georeferenced raster data. These data sets consist of a 2D matrix of height values (the raster data) as well as a so-called georeference which associates samples with their corresponding geographic location.

To transform a given data set to the quadtree database format (described previously), a bottom-up construction process is used. First, an optimal tree depth (resolution) is chosen to faithfully represent the input data. Then the subset of leaf nodes at this level which potentially intersect the data is identified. These nodes are then populated by resampling the raster input and finally the inner nodes are computed by downsampling. This process is repeated for each of the 12 base patches.

The choice of tree depth depends on the resolution of the input data. Due to the equal area property of HEALPix, the grid resolution for a given subdivision level is constant everywhere on the sphere. To faithfully represent the input data, we therefore chose a tree depth d such that the sample density of the leaf nodes is at least as high as in the input data.

In our application individual high-resolution data sets generally only cover a small fraction of the surface. Therefore the set of leaf nodes considered is limited to a subset which is likely to intersect the data. To approximate this set, we construct the boundary curve of the data by projecting each boundary pixel location to geographic coordinates (using the supplied georeference) and then to HEALPix coordinates. We select those leaf nodes which are located inside an axis-aligned box containing all of these points.

For each of the selected leaf nodes, we then iterate over all of the 255×255 sample positions, projecting each sample position first to geographic and then to raster coordinates. Bilinear interpolation within the raster data is used to compute the resampled value. After all intersecting leaf nodes have been populated in this manner, the inner nodes of the tree are derived by iterative downsampling.

Note that raster data sets can designate a special NODATA value, which is assigned to samples having no meaningful measurement. If any of the four input samples used in bilinear interpolation contain this value, the resulting interpolated sample is also marked as NODATA. This value is also assigned if the sampling coordinate (after projection) is not within the bounds of the raster image. If all samples of a node contain NODATA values after resampling, the node is not stored at all.

To support merging multiple data sets, it is also possible to insert data into an existing quadtree database. Already existing nodes are combined with newly generated ones by replacing their sample values. However if an incoming pixel is marked as NODATA, the previous value is kept. This treatment is required because the actual definition domain of many data sets is much smaller than their sampling support, with the difference areas being filled with NODATA samples. Using this scheme, it is possible to insert sparse high-resolution data into an existing low-res DEM database as shown in Figure 2.

The construction process can be trivially parallelized over the 12 base patches, as the quadtrees are mutually independent. By running 12 instances of the construction tool (possibly on different machines) and limiting each instance to only consider tiles within the associated base patch, a database file is generated for each base patch. A separate tool is then used to merge these files into a single database. While this is not optimal in terms of speed up (mainly due to the required I/O for merging the databases), it allows us to generate the hybrid MOLA and HRSC databases within a single day.

The input data sets have a size of 2 GiB for the MOLA DEM, 24 GiB for the HRSC DEM, 54 GiB for red, green and blue channels (HRSC) and 386 GiB for the high resolution B/W nadir channel (HRSC), for a total size of 466 GiB. This data is processed to a set of five databases (DEM, R, G, B, B/W) with a total size of 1.7 TiB.

3.2 Rendering

To render the terrain representation, each of the 12 base-patch quadtrees is recursively traversed in top-down fashion. Recursion stops at a tree node in any of three cases:

1. The node and therefore all of it's children are outside of the viewing frustum. Recursion returns without rendering.
2. The node is sufficiently subdivided to meet the screen space quality requirements (see 3.2.2). The node is rendered and recursion returns.
3. The LoD heuristic decides that further refinement is necessary, however the immediate child nodes are not in memory. In this case, the background I/O thread is instructed to load the four child nodes from disk and the current node is rendered as a placeholder until that data is available.

These cases are tested for in the order given. If none of these occur, recursion continues with the four children of the visited node. This rendering process is stateless in the sense that no information is kept about the set of nodes rendered in the previous frame, minimizing management complexity. In the following, the individual components of this architecture will be described in detail.

3.2.1 Frustum culling

Frustum culling is a technique to eliminate geometry which is located outside of the field-of-view of the camera and therefore guaranteed not to be visible. For performance reasons, bounding geometries are usually employed as proxy objects for this visibility test. Specifically, we follow the classical approach of maintaining an axis-aligned bounding box for each node which is resident in memory. In computing this bounding box, minimum and maximum height values within the node as well as any user-specified height exaggeration factor have to be considered.

Given the geometry of the view frustum, which is a pyramid truncated by two parallel planes, and the extents of the node's bounding box, the separating axis theorem is used to test the two bodies for intersection. This theorem states that, given two convex shapes, an axis exists onto which their projections are separate (non-intersecting) if and only if they are not intersecting.

When testing two polygonal meshes A and B for intersection, the set of axes which need to be tested in this manner is small. Specifically, only the set of face normals and the cross products of all pairs of edges where one edge is taken from A and the other is taken from B need to be considered (see [4]). If the projections of A and B onto any of these axes do not intersect, the original meshes do not intersect either.

The advantage of using axis-aligned bounding boxes is that the set of projection axes is constant for a given view-frustum as the set of face normals and edge directions of the bounding boxes are always the three canonical axes. Therefore, it needs to be computed only once per frame. Furthermore, the bounds of the projection of the view frustum onto this set of axes can likewise be pre-computed, accelerating the test.

3.2.2 LoD heuristic

Level-of-Detail rendering takes advantage of the limited resolution of raster displays by reducing geometric complexity depending on the apparent size of the geometry on the screen. In the visualization of large data sets, LoD techniques are often mandatory to achieve interactive rendering performance. We apply the following conservative approach using the screen-space area of bounding boxes.

One of the goals of the LoD scheme used in this work is guaranteeing maximum visual fidelity. When rendering polygon meshes, this translates into maintaining a geometric resolution of about one vertex per pixel when rasterizing. As our data-structure only allows to select between discrete resolution levels, nodes are refined during rendering until at least the desired level of resolution is reached. A node's bounding box is used to compute a conservative estimate of its screen coverage.

In order to estimate the number of pixels occupied by a node on the screen, the node's bounding box vertices are projected into screen space using the same projection and modelview matrices as used during rendering. Then for each of the six faces of the bounding box the signed area is computed. If the area of a face is negative, it is facing away from the viewer and ignored. All the positive areas are summed up to give the total screen area of the box in

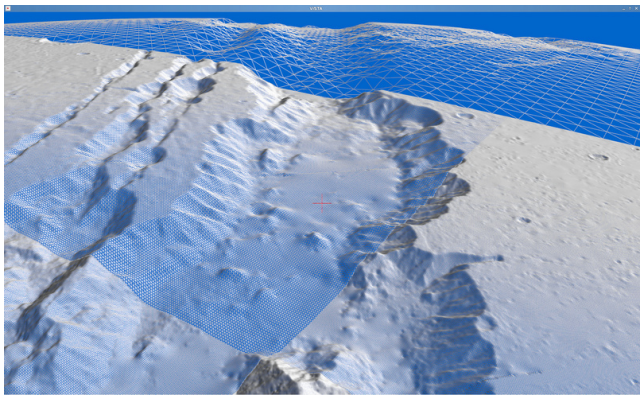


Figure 3 Distance-dependent LoD selection.

pixels, which is always greater or equal to the pixel area of the actual geometry if it were rasterized.

Finally, the estimated pixel area is compared with the number of mesh vertices, which is constant and equal to the number of height samples per node (255×255). If it is smaller, the subdivision level is adequate for the current view and the node is rendered. Otherwise, the recursion continues and the heuristic is applied again to the children.

As a guaranteed resolution of one vertex per pixel is excessive in most use-cases, we provide a user-selectable scaling factor which specifies the desired average number of vertices per pixel. In our experience, a choice of 0.2 provides high visual quality while maintaining good interactive performance.

Figure 3 shows how tree nodes close to the viewer are rendered at a high resolution which decreases with distance. Note that in this example the pixel area threshold was chosen very large for illustration purposes.

3.2.3 Data streaming

In order to guarantee interactivity, it is mandatory to perform slow disk operations asynchronously. For each database we maintain a separate I/O thread which performs these operations in parallel to the render thread. A job queue is used to store read-requests while a result queue contains the loaded data. If during rendering the LoD heuristic decides to refine a node but it's children are not yet in memory, a request is posted onto this queue to load the four child nodes. The I/O thread takes jobs out of this queue and processes them, appending the loaded data to the result queue.

At the beginning of each frame, before starting the actual rendering traversal, the render thread inspects the result queue and inserts any newly loaded nodes into the in-memory quadtree. These nodes are then available for subsequent rendering. This strategy restricts access to the quadtree to the render thread, reducing the complexity of thread synchronization, which is only required for shared access to the job and result queues.

3.2.4 Rasterization

Individual nodes are rendered using triangle meshes interpolating the height field. Each node represents a square sub-region in the parameter space of its associated HEALPix patch. The coordinates of individual mesh vertices are derived by equidistant interpolation within this region. For rendering, mesh vertices are projected to geographic coordinates

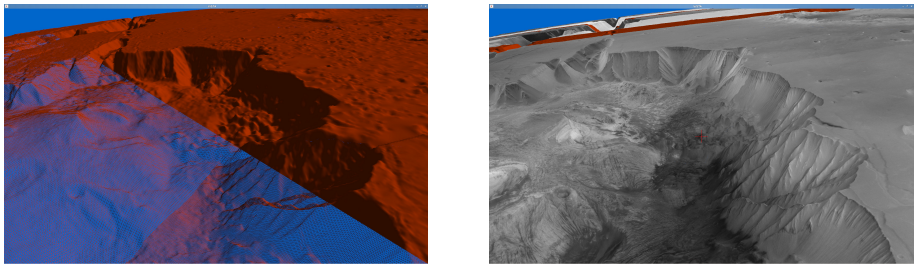
(latitude / longitude) using the HEALPix formulae. These coordinates are then combined with the corresponding height samples (radii) to form polar coordinates which are eventually converted to Euclidean coordinates. Storing vertex coordinates in Euclidean coordinates for each resident node would be expensive, however. Instead, we perform the coordinate system conversion on the fly during rendering using a vertex shader.

Our approach uses a single 2D proxy mesh which is a uniform tessellation of the $[0, 1]^2$ unit square. As this mesh is re-used for rendering each tree node, it's memory footprint is negligible. Per-node elevation data is stored in a 255×255 scalar texture. For each node, the vertex shader implements the following four steps to transform the proxy mesh to the final geometry:

1. Transform mesh to proper sub-region in parameter space
2. Convert parametric coordinates to geographic (latitude, longitude)
3. Read height values out of texture and generate spherical coordinates (r, ϕ, θ)
4. Convert to Euclidean coordinates (x,y,z)

Texture coordinates are centered on the height samples (texels), guaranteeing that the sample points are interpolated by the geometry. To implement shading, normal vectors are estimated using central differencing of the height field. To avoid analytical computation of the u, v direction vectors necessary for the normal estimation, we pre-compute them for the corners of each tile and pass these so-called tangent space matrices to the shader, which interpolates them across the patch.

3.3 Imagery overlay



(a) Shaded DEM (wireframe vs. opaque) **(b)** DEM with high-res BW imagery

Figure 4 Shading vs. Texturing.

In the following we extend the approach previously presented for rendering DEMs to incorporate imagery. Imagery data is processed in the same way as DEMs, producing parallel quadtree databases. The advantage of not combining DEM and imagery into a single database is that both can be arbitrarily mixed and matched at runtime.

In the rendering traversal, DEM and imagery database nodes are now visited in parallel. A straightforward approach to render each pair of data is to extend the vertex shader to assign vertex colors from an imagery texture. However, this is not sufficient due to the fact that imagery data in our applications is of higher resolution than the DEM. These additional levels of resolution are never displayed in this scheme, as the visualization is constrained by the DEM resolution.

Therefore, we chose a different approach which is illustrated in Figure 5. We introduce a parameter Δ_h which specifies the maximum resolution difference between a DEM node and

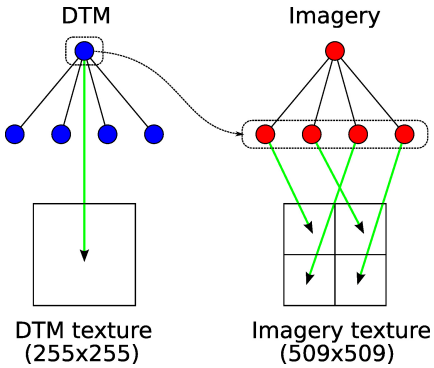


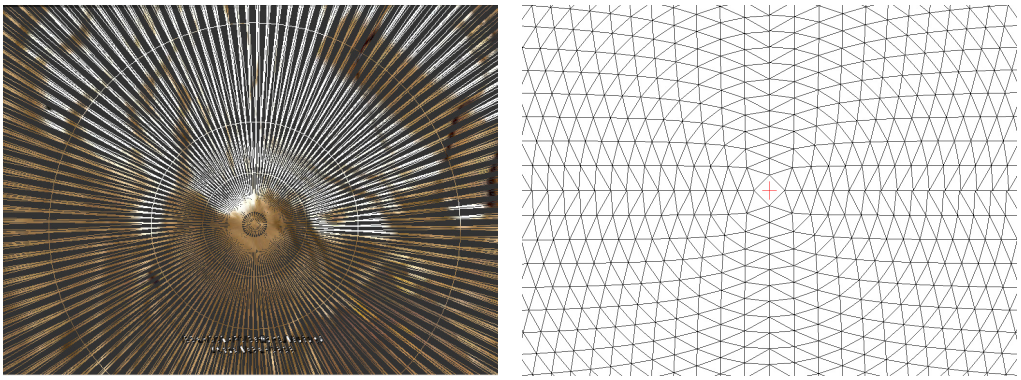
Figure 5 Merging multiple imagery nodes into single texture ($\Delta_h = 1$).

the corresponding imagery. When rendering, the DEM and imagery tree are traversed in parallel as described above. However, instead of displaying the imagery node at the same resolution level as the DEM node, we recurse further into the imagery database and collect all child nodes of degree Δ_h. We then merge the corresponding tiles into a single large texture. Figure 5 shows an example for $\Delta_h = 1$. Regular (fragment stage) texture mapping is then used to provide additional visual detail without increasing geometric primitive count.

The selection of which databases to display can be made at runtime. To simplify switching to another set of channels, we simply delete the resident quadtrees from memory and let the rendering traversal (re-)load any nodes required for the current view, which usually takes less than a second.

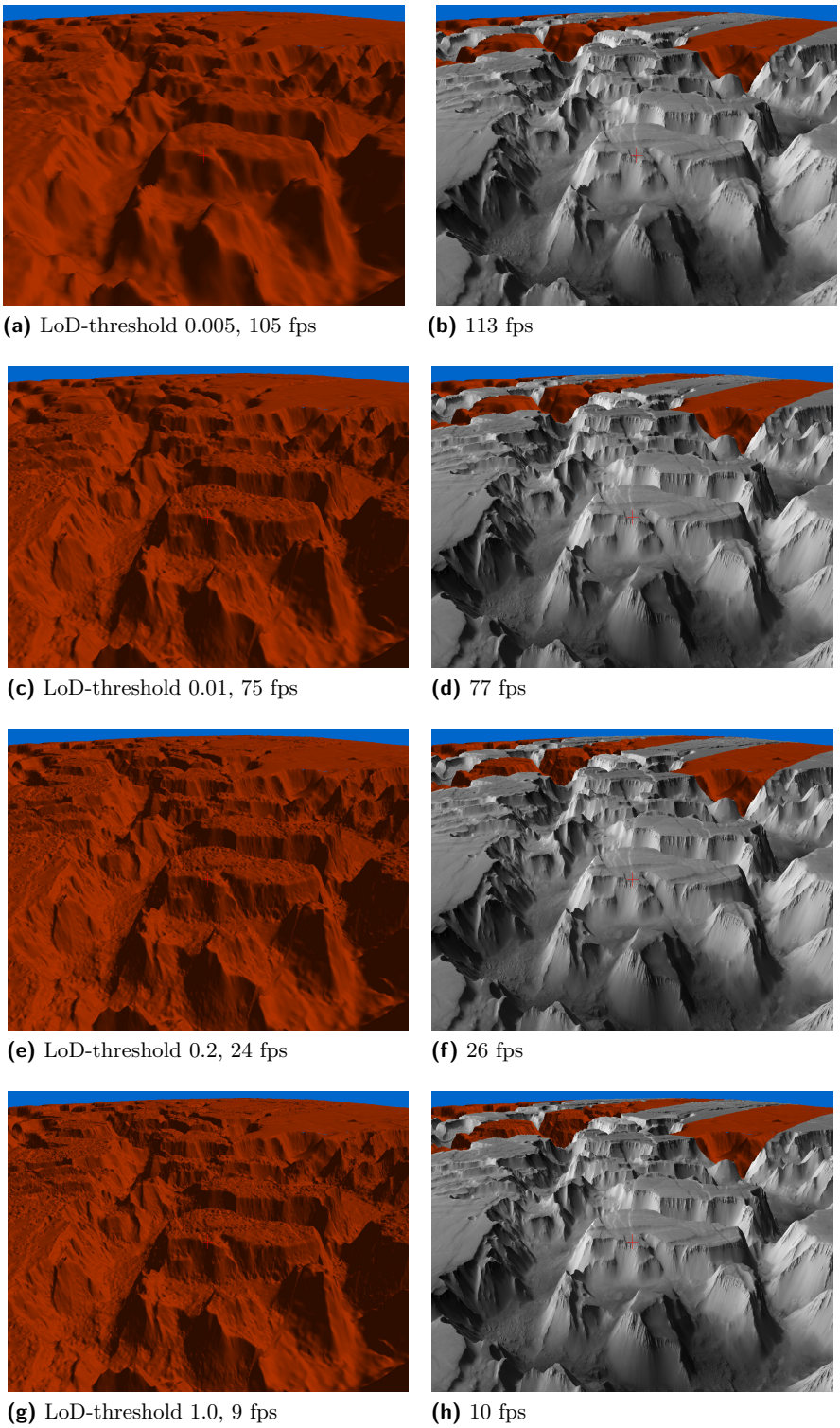
4 Results

Figure 6 shows the triangulation quality in the vicinity of the poles, comparing Google Earth to our approach. Google Earth produces a bad triangulation consisting of long, skinny triangles due to an obvious coordinate singularity at the pole. In contrast, using a uniform sampling of the parametric HEALPix coordinates yields a well-behaved triangulation consisting of equal-area triangles with good aspect ratios.



(a) Google Earth **(b)** Our system

Figure 6 Comparison of triangulations at the north pole.



(a) LoD-threshold 0.005, 105 fps **(b)** 113 fps

(c) LoD-threshold 0.01, 75 fps **(d)** 77 fps

(e) LoD-threshold 0.2, 24 fps **(f)** 26 fps

(g) LoD-threshold 1.0, 9 fps **(h)** 10 fps

Figure 7 Visual quality and rendering performance at different LoD-thresholds (left: shaded DEM, right: DEM textured with high-res B/W channel ($\Delta_h = 2$)).

Figure 7 shows the surface of Mars, visualized at different LoD-thresholds. Window size is 1024×768 pixels in each case. While rendering performance is not fully interactive at a threshold of 1.0 (min. 1 triangle per pixel), visual quality at 0.2 is not noticeably worse while providing interactive frame rates. The fact that adding imagery textures does not significantly affect the results indicates that performance is limited by GPU geometry processing performance.

While image quality degradation is obvious when comparing the shaded DEMs at thresholds 0.2 and 0.01, it is hardly noticeable when the same geometries are compared with imagery texturing. Therefore, very low LoD-thresholds can be used when imagery is present, resulting in highly interactive frame rates.

5 Conclusion and future work

We have presented an interactive terrain rendering architecture using the HEALPix coordinate system to provide spherical rendering without singularities. By performing critical computations on the GPU, both memory consumption and management complexity are reduced compared to other schemes.

Possible future extensions including incorporation of LiDAR data by using scattered data interpolation schemes to resample data to our uniform grids, locally refining the tree depending on sample spacing.

In the algorithm presented, normal vectors at tile boundaries are discontinuous, resulting in small rendering artifacts. To solve this problem and to provide additional visual detail a low LoD-thresholds, we want to implement normal mapping at a higher resolution than the topography, using the same scheme currently used for imagery overlays.

References

1 Tony Bernardin, Eric Cowgill, Oliver Kreylos, Christopher Bowles, Peter Gold, Bernd Hamann, and Louise Kellogg. Crusta: A new virtual globe for real-time visualization of sub-meter digital topography at planetary scales. *Computers & Geosciences*, 37(1):75–85, 2011. Virtual Globes in Science.

2 Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. Planet–sized batched dynamic adaptive meshes (p-bdam). In *Proceedings IEEE Visualization*, pages 147–155, Conference held in Seattle, WA, USA, October 2003. IEEE Computer Society Press.

3 Malte Clasen and Hans-Christian Hege. Terrain rendering using spherical clipmaps. In *EuroVis*, pages 91–98, 2006.

4 S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 171–180, New York, NY, USA, 1996. ACM.

5 K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2):759, 2005.

6 R. Kooima, J. Leigh, A. Johnson, D. Roberts, M. SubbaRao, and T.A. DeFanti. Planetary-scale terrain composition. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):719–733, sept.-oct. 2009.

7 Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 769–776, New York, NY, USA, 2004. ACM.

Visualization and Evolution of Software Architectures

Taimur Khan[1], Henning Barthel[2], Achim Ebert[1], and Peter Liggesmeyer[2]

1 **Computer Graphics and HCI Group**
University of Kaiserslautern, Germany
{tkhan,ebert}@informatik.uni-kl.de

2 **Fraunhofer IESE**
Kaiserslautern, Germany
{Henning.Barthel,Peter.Liggesmeyer}@informatik.uni-kl.de

Abstract

Software systems are an integral component of our everyday life as we find them in tools and embedded in equipment all around us. In order to ensure smooth, predictable, and accurate operation of these systems, it is crucial to produce and maintain systems that are highly reliable. A well-designed and well-maintained architecture goes a long way in achieving this goal. However, due to the intangible and often complex nature of software architecture, this task can be quite complicated. The field of software architecture visualization aims to ease this task by providing tools and techniques to examine the hierarchy, relationship, evolution, and quality of architecture components. In this paper, we present a discourse on the state of the art of software architecture visualization techniques. Further, we highlight the importance of developing solutions tailored to meet the needs and requirements of the stakeholders involved in the analysis process.

1998 ACM Subject Classification D.2.11 Software Architectures

Keywords and phrases Sofware architecture visualization, software comprehension, software maintenance, software evolution, human perception

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.25

1 Motivation

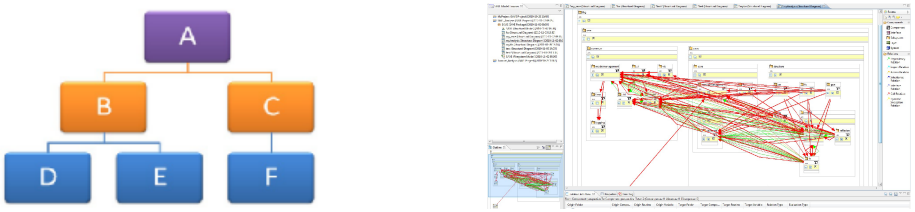
The field of software visualization is centered on visual representations aimed at making the software more comprehensible. These representations are a necessity for analysts to examine software systems due to their "complex, abstract, and difficult to observe" nature [53]. These difficulties are further compounded in large-scale software systems where it becomes increasingly difficult for analysts to examine the systems' behavior and properties, due to the systems' scale.

Software visualization focuses on various aspects of software systems, such as source code, software structure, runtime behavior, component interaction, or software evolution, to unravel patterns and behaviors through the different software development stages [1]. Due to the diverse nature of these data sets, different types of visualizations can be found in literature. However, for the focus of this research we highlight the visualization of software architecture as well as software architecture evolution.

The visualization of software architecture is an essential component of software visualization. "Not only are architects interested in this visualization but also developers, testers, project managers, and even customers" [32]. From the perspective of a software analyst,

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 25–42

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

(a) General node-link diagram **(b)** As found in SAVE [23]

Figure 1 General and tool specific node-link diagram.

software architecture focuses on the structure of a software system – the focal point of which is to examine composing entities, their metrics, and relationships [11]. Additionally, recent studies have shown an increased interest in not only the visual exploration of software modules, their structure, and interrelations, but also in the evolution of these modules [19]. The key feature of software architecture visualization is to uncover visual metaphors that are both efficient and effective in depicting the software architecture of a system and to encode software code metrics within these representations. Several questions need to be addressed in finding such solutions, such as: who is the end-user of the architecture visualization [50], what needs to be analyzed through the visualization [52], and how can appropriate visualization metaphors and interaction techniques be chosen [2].

2 Visualizing Software Architectures

One of the core topics in the field of software visualization is a means to effectively visualize, navigate, and explore the software architecture of a system [31, 32, 34]. Generally, object-oriented software tends to be structured hierarchically – with packages containing sub-packages, which in turn contain classes that hold methods and attributes. It is this hierarchy and relationships between software components that are of interest when it comes to software architecture visualization [15].

In this section, we explore representations of the global architecture of a system, such as tree, graph, and diagram model depictions. Further, we also investigate representations that highlight relationships between components as well as the importance of visualizing software metrics.

2.1 Architecture Representations

Tree structures are an ideal way of representing the hierarchical structure of software architecture. However, research in this area has shown the need to move forward from well-known techniques such as node-link layouts to more sophisticated ones to handle the larger hierarchies found in software systems nowadays [70]. Fig. 1 shows both a generic node-link diagram as well as one found in a commercial tool. Inspection of these representations shows that they quickly become too large and utilize available screen space far too poorly for proper investigation. Further, the amount of textual information represented in the nodes as well as the way relationships are depicted should be revisited to avoid visual clutter and information overload [41].

This section inspects several 2D visual representations [10] that may not be specific to just software visualization, but have been effectively applied to highlight the hierarchal structure of a software system [70, 4]. Here, it is important to note that a lot of these representations

(a) Rectangular TreeMap **(b)** Circular TreeMap

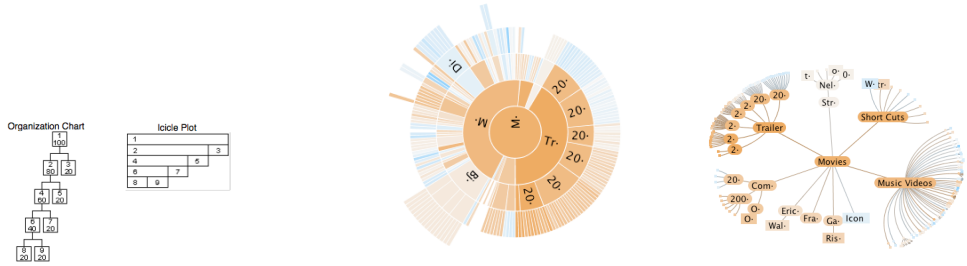
Figure 2 Rectangular and Circular TreeMaps from [57].

have been extended to 3D visualizations [2, 6, 49]. While 3D approaches have been shown to display larger hierarchies and minimize clutter [58], they have also suffered from the well documented drawbacks of 3D visualizations, such as: object occlusion, cumbersome view adjustments, performance issues, as well as poor readability of 3D texts [48, 17]. Due to these drawbacks and the requirements of our stakeholders, this survey focuses mostly on 2D representations.

The *Treemap visualization* (Fig. 2a), first introduced by Johnson and Schneiderman [39], is an effective means to visualize an entire software hierarchy. It is essentially a space-filling technique that displays hierarchical data as a set of nested rectangles. This is usually performed by a tiling algorithm that slices a box into smaller boxes for each level of the hierarchy, recursively, alternating between horizontal and vertical slices. "The resulting visualization displays all the elements of the hierarchy, while the paths to these elements are implicitly encoded by the Treemap nesting" [15]. In the context of software architecture visualization, Treemaps are used to represent methods as elementary boxes and classes as composed boxes. Several modifications of Treemaps appear in literature and in practise – some improve readability by enforcing an aspect ratio as close as possible to 1, while others have used irregular shapes such as Voronoi instead of rectangles to show more information [8]. Typically, designers are limited to the encoding of a single metric – the box color. While this provides a symbolic idea of how such a metric value is spread through the hierarchy, it is not simple to determine or represent metrics of enclosing entities [22]. Treemaps provide an extremely compact layout, however, they are limited by mainly showing the leaves of the software structure. Similarly, the *circular Treemap visualization* (Fig. 2b) and variations of it have been researched in order to have circles fill the available space [74]. However, as shown in Fig. 2b circular treemaps are not efficient with respect to the used space.

The *Icicle Plot* principle of Fig. 3a is where a line represents a tree level and each line is split according to its number of children [10]. While Icile Plots provide better understanding of structural relationships as packages can be used as root and classes and methods as tree elements, scalability and navigation may be an issue with hierarchies of large systems [22]. Typically, two metrics maybe encoded in the visual representations: node size and color.

An alternative space-filling technique to nested geometry is the use of a *Sunburst visualization* that focuses on adjacencies instead [62]. This technique was first proposed by Stasko and Zhang [65], where they utilized a circular or radial display to depict the hierarchy rather than a rectangular layout (Fig. 3b). In a sunburst, the hierarchy is laid out radially with the root at the center and dics or portions of discs as deeper levels further away from this center [3]. In contrast to the Treemap techniques mentioned earlier and similar to the Icile Plot, designers have the added flexibility to encode two distinct metrics: the angle swept out by



(a) Icicle Plot **(b)** Sunburst Tree Layout **(c)** Hyperbolic Tree Layout

Figure 3 Icicle Plot [10], Sunburst Tree Layout [57], and Hyperbolic Tree Layout [57].

an item and its color [22]. Studies have shown the performance of localization, comparison, and identification tasks in Treemap and Sunburst visualizations to be comparable, however the Sunburst is found to be easier to learn and more pleasant [64]. While screen-space is better utilized as compared to node-link diagrams, scalability and navigation may still be an issue in larger systems.

Another approach is to make use of the hyperbolic space, which intrinsically provides more space than a layout that employs Euclidean coordinates. This well-established technique is more commonly referred to as the *hyperbolic tree layout* (Fig. 3c) and was first introduced in the context of information visualization by Lamping et al. [43]. Essentially, it lays out the hierarchy in a uniform manner on a hyperbolic plane and maps the results back on to the Euclidean space. The resulting hierarchy is laid out on a circular display region and may be complemented with focus and context techniques such as *fisheye distortion* [40], where components tend to diminish in size as they move outwards. This leads to a larger representation of the center or focused area while still displaying the overall structure of the tree. Hyperbolic trees show detail and context at once; initially the root of the hierarchy is placed in the center, however, the display can be transformed to bring another node into focus through interaction. It would probably be best to encode metrics through the use of color alone, as varying the node size would adversely affect the layout algorithm. When the graph is deemed too large to be rendered effectively, nodes are pruned together and may be interactively expanded to reveal the subtree structure.

2.2 Visualizing Relationships

In contrast to visualizing the software hierarchy of a system, visualizing relationships of the software system is a more complex task. This is due to both the higher amount and the different types of relations that exist in a system, such as: inheritance, method calls, dynamic invocation, accesses, etc.

Generally, *graphs* have all the characteristics required to represent relationships of a software system. This is typically done by expressing software components as nodes and relationships between them as edges [63]. However, this often leads to the visualization of an extremely large graph due to the high interconnectivity between the large amount of components found in software systems nowadays. Thus, the resulting visualization tends to be extremely confusing and cluttered – it becomes difficult to discern between nodes and edges due to the cluttering, overlapping, and occlusion of edges (Fig. 4).

A well-known approach to remedy this clutter issue is to replace node-link diagrams with a square matrix that has matching row and column labels. The matrix then highlights the

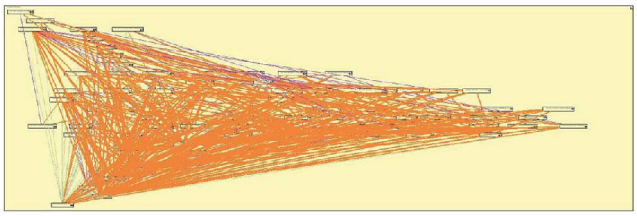


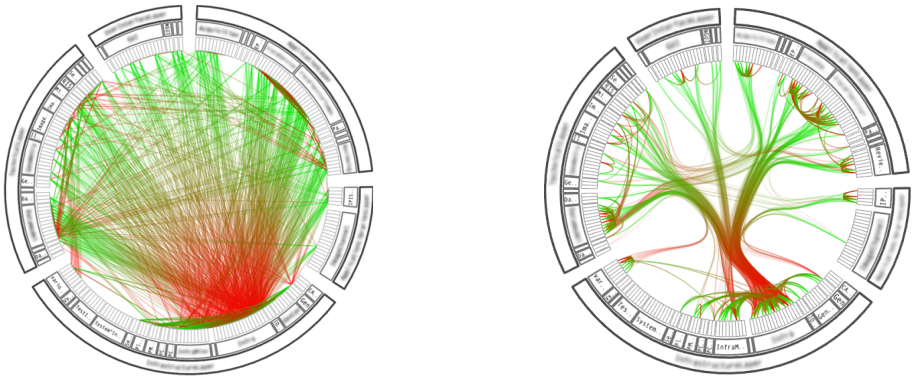
Figure 4 Cluttered Software Architecture [23].

number of relations between row and column elements within each matrix entry, possibly through some visual representation [78]. This well-known technique is often referred to as the *Dependency Structure Matrix* [59] in literature and provides a compact and uncomplicated representation of relations in a complex system. However, keeping a mental map of the system hierarchy can still be an issue in these visualizations.

The most accepted graph-based software visualization in the field of object-oriented software engineering are *UML class diagrams*. This modeling language was created and developed by the Objected Management Group and has since become the industry standard for modeling software systems [28]. Its main purpose is to portray inter-class relations, such as: composition, inheritance, generalizations, aggregations, and associations. However, due to the amount of textual information depicted by each component such as the listing of methods and variables, these graphs grow exponentially with each additional component or class notation and are highly prone to information overload. Some researchers have looked at reducing the visual complexity associated with such graphs by reducing the number of overlapping edges, the use of orthogonal layouts, the horizontal writing of the labels, and edge bundling [24, 56, 68]. While some success in reducing the complexity has been achieved, the drawbacks associated with node-link diagrams such as poor screen-space management and information overload still need to be tackled.

Some researchers have experimented with different layout and filter techniques in order to resolve the clutter issue. An example of this is the work of Pinzger et al. [55] that focuses on the creation of condensed and aesthetically pleasing graphs that show information relevant to solve a given program comprehension task. Their solution was to use nested graphs and a feature that allowed to add and filter appropriate nodes and edges. Other researchers such as Holten [36] have chosen to implement better space-filling techniques in combination with improved edge representations. Holten's approach was to place software elements on concentric circles according to their depth in the hierarchical tree and then to display edges above the hierarchical visualization (Fig. 5). Further, he extended the work of Fekete et al. [26] that used spline edges to replace explicit arrow directions, in order to reduce the visual clutter and edge congestion by allowing edges to bundle together according to a parameter (Fig. 5a and 5b). Similarly, techniques displaying, clustering, and filtering edges on top of structural representations can be utilized in other visualizations (Treemaps, circular trees, etc.) to represent the hierarchical graph structure of a software system.

Another approach to resolve the issues of cluttered 2D graphs is the use of 3D visualizations [29], where the user can access a view without occlusions. However, 3D representations of large graphs have their own problems, such as: navigation can not only be difficult but also disorienting [60], object occlusion, performance issues, and text illegibility [48]. For the purpose of completion it would be prudent to mention some of the more prominent work in the area of 3D software architecture visualization. Some researchers in this field experimented with real-world metaphors to take advantage of the intuitiveness of these representations [51].



(a) $\beta = 0$ **(b)** $\beta = 0.75$

Figure 5 Hierarchical Edge Bundles [36].

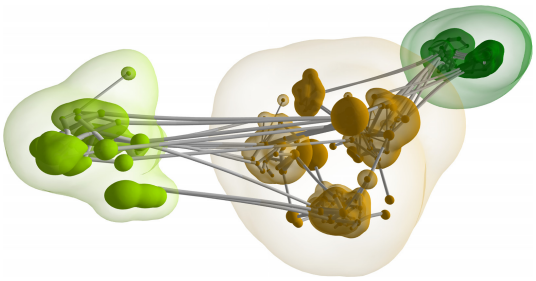


Figure 6 Clustered graph layout [7].

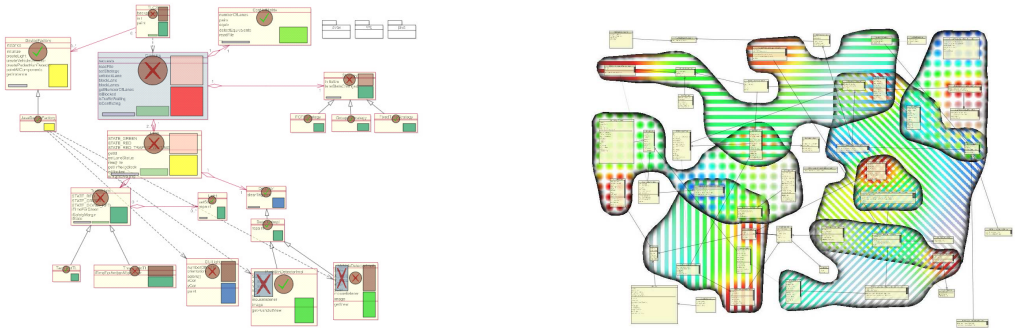
For example, the *City* or *Cities* metaphors are often used to depict relationships through a visually understandable metaphor [2, 52], where cities (packages) are connected via streets (two-directional calls) and water (uni-directional calls). Similarly, researchers have realized the *Solar System* [33], *Island* [52], and *Landscape* [6, 9] metaphors, where the respective relationships between each contributing element is exploited to depict packages, classes, and their relationships. Another interesting approach towards handling large and complex graphs is the *clustered graph layout* (Fig. 6), where clustering, dynamic transparency, and edge bundling are used to visualize a graph without altering its structure or layout [7].

2.3 Visualizing Software Metrics

The incorporation of software metrics is an important component in the analysis of a software systems architecture, as they not only provide an insight into the quality of the software design [14, 27] but also a means to monitor this quality throughout the design process [12]. Typical static software metrics express different aspects of a complex system, such as: design complexity, resource usage, and system stability.

The idea behind metric-centered visualizations is to transform numerical statistical data into a visual representation that is easier to understand and grasped far more intuitively and instantaniously [75]. Here, the greatest challenge is to find an effective mapping from a numerical representation to a graphical one that enhances the structural visualization [38].

In this section, selected visualization techniques that implement static software metrics are highlighted – the purpose of which is to provide an idea of the implemented approaches. One such approach is to combine them with UML class diagrams. An example of this is



(a) Metric View **(b)** Areas of Interest

Figure 7 Metric View [71] and Areas of Interest Visualizations [13].

the *MetricView* (Fig. 7a) visualization that displays metric icons on top of UML diagram elements [71].

An extension of this approach is the *areas of interest* (Fig. 7b) technique developed by Byelas and Telea [13]. They apply a layout algorithm that groups software entites with common properties, encloses these entities with a contour, and adds colors to depict software metrics. In order to distinguish overlapping areas, each area is given its own texture, such as: horizontal lines, vertical lines, diagonal lines, and circles. Further, shading and transparency techniques are used to improve the distinction between several areas.

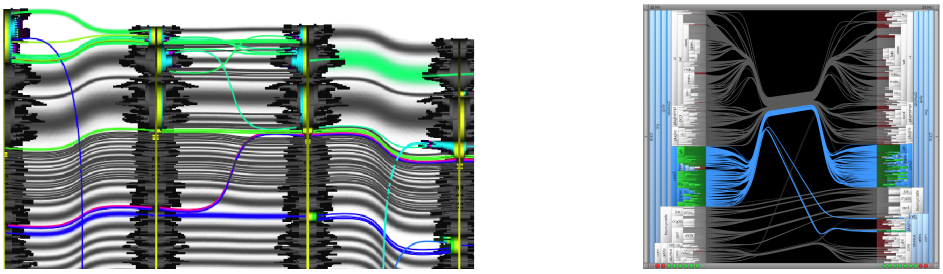
In visual representations other than UML Diagrams, similar approaches have to be implemented in order to combine metrics and structural information. An example of this is the work of Holten et al., where they used texture and color to show two different software metrics on a Treemap [35]. Their results show that the combination of color and texture provides high information density, assists in finding correlations between metrics, and can reveal patterns and potential problem areas.

To visualize multiple aspects of a software system, Lanza et al. introduced the concept of *polymetric views*, where the visualization of a software is enriched with software metrics [46]. Essentially, they propose a node representation that encodes up to five distinct metrics; node width, height, x and y-coordinates and color, and edge width and color. They applied this to an inheritance tree where nodes represent classes and edges depict the inheritance relationship between them. Node width and height is used to encode the number of attributes and the number of methods. Further, a color tone is applied to represent the number of lines of code.

Similarly, in 3D visualizations the encompassing visual entities have been encoded with software metrics [33, 76]. Another technique that may be applied in the analysis of system metrics is the use of filters. An example of this can be found in the Solar system metaphor, where filters may be applied to the overall system to visualize planets with metric values that lie within a chosen interval [33].

3 Visualization of Architecture Evolution

A general obstacle with regards to software evolution visualization is coping with the complexity that emerges from the huge quantity of evolution data; it is quite common to have hundreds of versions of thousands of files [72]. The technical challenges associated with extrapolating these historical data are deemed out-of-context with respect to this paper, instead, the focus will be on visualizing the evolution of the software architecture.



(a) Code Flow

(b) Visual comparison of hierarchically organized data

Figure 8 Code Flow technique [69] and structural comparison of two source code versions [37].

Real software solutions undergo continuous change to meet new requirements, adapt to new technology, and to repair errors [47]. Inevitably, the software in question magnifies in both size and complexity, often leading to a situation where the original design gradually decays unless proper maintenance is performed [20]. As such, visualizing the evolution of the software architecture is one of the key topics in the field of software evolution visualization [15]. It is essential to have a global overview of the entire system evolution in order to explain and document how a system has evolved to its present state and to predict its future development [18].

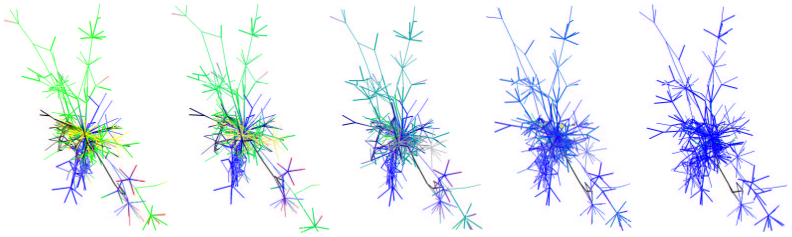
This section follows the same pattern as the previous one, where we first focus on how the global architecture of the software changes with each release and then examine how relationships and metrics evolve within each version.

3.1 Visualizing Hierarchical Changes

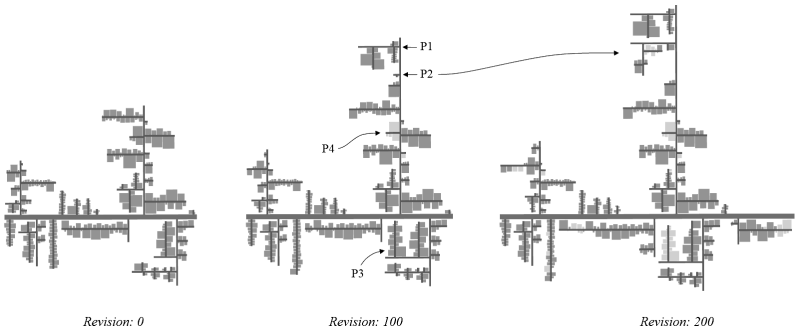
Since software maintenance is performed mainly at code level, most visualizations have implemented a 2D line-based approach to represent the software evolution [25, 69, 73]. Generally, the adopted approach is to visually map a code line to pixel line, where color is typically used to show the age of a code fragment [25]. Additional focus has been to develop interaction techniques that allow users to effectively navigate and explore the data [73]. In order to highlight the state of the art in this traditional approach, the *Code flows* visualization technique [69] is briefly examined. Fig. 8a shows an evolution from left to right of four versions of a source code class. This technique employs an icile layout and bundled edges to show how a source code line changes over subsequent versions. Source code lines that do not change from one version to another are colored black, while code lines that changed are highlighted using different colors. In general, these tools are successful in tracking the line-based structure of software systems and reveal change dependencies at given moments in time [73]. However, they lack the sophistication to provide insight into attribute changes and more so the structural changes made throughout the development process.

In contrast, there are only few visualizations aimed at representing structural changes of a system architecture over time [15]. As explained earlier, there definitely exists a requirement to monitor the evolution of a systems architecture, however, current graph animation algorithms are limited and need to mature further to handle this requirement [21].

One such approach is the work of Holten et al. [37] that presents a technique aimed at comparing the software hierarchies of two software versions . To better compare the two versions, the algorithm tries to position matching nodes opposite to each other. This



(a) Inheritance graph of successive versions



(b) Evolving Cities

Figure 9 Successive Inheritance graphs [16] and development stages of CrocoCosmos [66].

technique is presented in Fig. 8b, where the source code of Azureus v2.2 is displayed on the left and v2.3 is portrayed on the right. Nodes that are present in one version but not the other are highlighted via red shading. Further, the Edge Bundles technique of Section 2.2 is used to highlight and track the selected hierarchy.

Collberg et al. [16] describe a system that visualizes the evolution of a software system using a graph drawing technique that handles a temporal component for the visualization of large graphs. They accomplish this by utilizing a force-directed layout to plot call graphs, control-flow graphs, and inheritance graphs of Java programs. Changes that the graphs have gone through since inception are highlighted through the use of color. Nodes and Edges are initially given the color assigned to its author (red, yellow, or green) and progressively age to blue (Fig. 9a).

Lately, there has been some effort by researchers to extend known metaphors to handle the evolution of software systems. Steinbrückner et al. [66] have an interesting approach that implements the city metaphor for the representation of large software systems in the form of evolving software cities. Their work is illustrated in Fig. 9b, where a system grows from an initial 389 classes to 439 classes in revision 100 and 466 classes in revision 200. In this implementation of the city metaphor, streets represent Java packages and building plots represent Java classes. The sequence of visual depictions aims to highlight basic changes in the software structure, how elements may be added, removed, and moved within the software hierarchy. Further, they extend this general representation to address the needs of two distinct application scenarios by: 1) applying an *evolution map* that uses contour lines to show different versions of each subsystem and 2) using a *modification history map* that uses a contour line map combined with property towers that depicts the number of modifications as height and modification date as color.

3.2 Visualizing Software Metrics Evolution

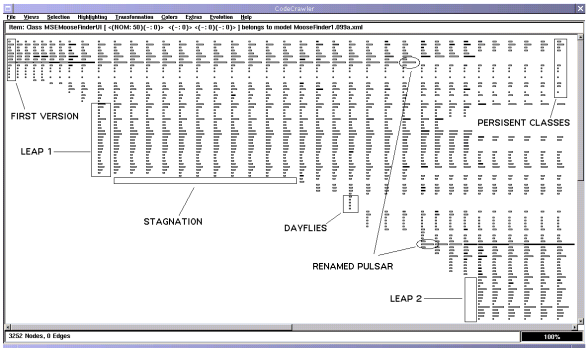
As covered in Section 2.2, visualizing relationships is an extremely complex task that is further compounded in the case of software evolution. Typically, researchers and practitioners focus more on the logical coupling between source code artifacts, as it can be encoded easily into metric values [30].

Software metrics are an ideal abstraction as they encapsulate, summarize, and provide essential quality information about source code [44]. As such, they are essential in providing a continual understanding and analysis of the quality of a system during all phases of the product life cycle. Instead of tedious, inefficient, and hard to grasp numerical representations, metrics tend to be mapped to graphical characteristics so that they may be intuitively interpreted. In this section, we explore the state of the art in the visualization of software metrics across different software versions.

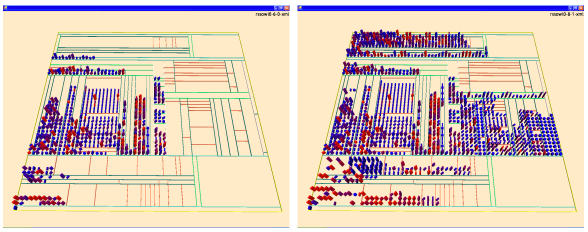
The *Evolution Matrix* is a visualization technique that provides an exploratory view of an object-oriented systems evolution, both at the system and class granularity levels [45]. In this work, Lanza et al. combine software visualization and software metrics by using two-dimensional boxes to represent classes and encoding metric measurement of the classes to the width and height of the boxes. In the example of Fig. 10a, they use the metric *number of methods* for the width and *number of instance variables* for the height, columns to represent different versions of the software, and rows to depict different versions of the same class. At the system level, this technique recovered the following characteristics regarding the evolution of a system: size of the system, addition and removal of classes, and growth and stagnation phases in the evolution. While at the class level, it shows if the class grows, shrinks, or stays the same from one version to another. These features allow the expert to analyze a number of interesting aspects, such as a class growing and shrinking repeatedly, a class suddenly exploding in size, or a class that had a certain size but lost its functionality.

The visualization framework by Langelier et al. also facilitates the analysis of software over many versions [44], albeit in a slightly different manner. Instead of employing a technique that displays the entire system evolution in one picture [45], they rely on animated transitions from one version to another. As Fig. 10b shows, there are different static representations for each subsequent version; the image on the left is a previous version and the image on the right is the next. The user controls forward and backward navigation in time, which in turn animates three graphical characteristics that are mapped to metric values – color, height, and twist. While the animations are of a short duration, they are well-designed and help attract the attention of the viewer towards program modifications [44]. This work of Langelier et al. contains references to extensive case studies aimed at detecting both evolution patterns and known anomalies. With respect to evolution patterns, users were able to identify constantly growing classes, quick birth and death of classes, and explosions in complexity in a short time-span. On the other hand, while looking for common anomalies, patterns such as the *God Class* or *Shotgun Surgery* were observed. The former is detected when a class constantly grows in complexity and coupling, while the latter occurs when a class constantly grows in terms of coupling and whose complexity increases globally but with an up-and-down local pattern.

Wettel and Lanza present interactive 3D visualizations in their *CodeCity* tool that examines the structural evolution of large software systems at both a coarse-grained and a fine-grained level [77]. At a coarse-grained level of granularity, classes are shown as monolithic blocks that lack details of the internal structure. At the fine-grained level, the focus is on methods that appear as building bricks. Fig. 11a shows this fine-grained representation, where classes are illustrated as buildings located in districts that represent the packages in



(a) Evolution Matrix

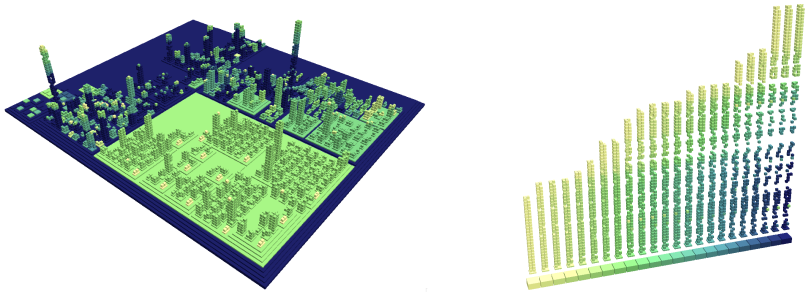


(b) Two frames of RSSSowl using VERSO

Figure 10 Evolution Matrix [45] and two frames from VERSO [44].

which the classes are defined. Metric values are then encoded in the visual properties of the city artefacts; class properties such as the number of methods and number of attributes are mapped on to the buildings' height and base size, package depth is mapped on the districts' color saturation. Further, the age distribution of classes is represented through an Age Map color mapping, where the color scheme ranges from light-yellow for recent entities to dark blue for earlier versions. Similar to the work of Langelier et al., back and forth transitions through the history of the system allow the city to update itself and reflect the currently displayed version. Additionally, at a finer level-of-detail the entire evolution of a single class or package may be tracked (Fig. 11b).

Pinzger et al. introduced a multivariate visualization technique that can display the evolution of numerous software metrics related to modules and relationships [54]. This is accomplished through a combination of *graphs* and *Kiviat diagrams* to graphically represent several metric values by plotting each value on its corresponding line (Fig. 12). The individual Kiviat diagrams present quantitative metrics, where low values are placed near the center of the Kiviat diagram and high values are found further away from the center. Dependency relationships between source code entities are highlighted by the layout of the diagram and the relationship between modules. Furthermore, this approach encodes the temporal aspects of multiple versions through a rainbow color gradient, where different colors indicate the time period between subsequent releases. Finally, the amount of coupling between two modules is represented by the width of edges connecting Kiviat diagrams. While this visualization contains lots of informations and can help identify critical source code entities or critical couplings, it requires a good knowledge of software metrics. A positive feature of this technique is that all information regarding metrics and evolution is represented in a single static view that requires no animation. However, at times the color stripes overlap, making it futile to discern the corresponding metric values. This problem of overlapping has been



(a) A fine-grained CodeCity Age Map **(b)** Timeline of a single class

Figure 11 Fine-grained CodeCity Age Map and Timeline of a single class [77].

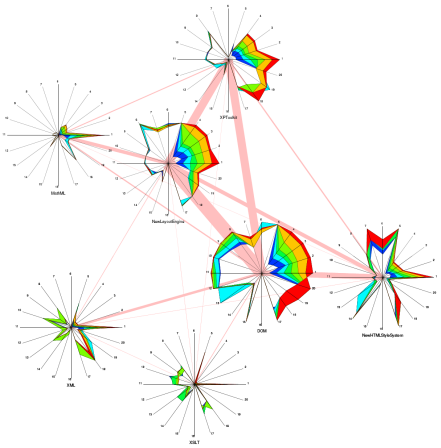


Figure 12 Kiviat graph with 20 metrics, 7 modules, and 7 subsequent releases [54].

solved using 3D Kiviat diagrams that displays each version of the software on a different level of elevation [42].

4 Tools

There are a number of tools available both in academia and industry that cater to the various needs of stakeholders. On the one side, vendors have developed commerical Architecture Visualization Tools (AVTs): Lattix, Enterprise Architect, NDepend, Klockwork Architect, IBM Rational Architect, Bauhaus [70], etc. On the other hand, the research community has also produced numerous tools: SHriMP [67], BugCrawler [19], DiffArchViz [61], etc. Commercial tools are generally designed to be used as-is, while research tools are open-source that allow users to customize them.

The main aim of these tools is to employ a combination of metaphors and techniques presented in this paper to assist technical users, project managers, and researchers in analyzing software architectures. The study of Telea et al. shows that the mainstream masses are starting to realize the potential of these visualization techniques. For example, tools such as Lattic and NDepend have incorporated newer diagram-layout techniques, realizing the limitations of traditional node-link diagrams [70]. However, this modernization of AVTs is much slower than the advent of cutting-edge visualization solutions.

AVTs typically support a combination of the following tasks: "comparing desired and actual architectures, identifying architecture violations, highlighting architecture patterns or layers extracted from code bases, assessing architecture quality, and discovering evolutionary patterns such as architectural erosion" [70]. However, no single tool can satisfy all these needs and requirements, as they differ in the features they provide, the audience they cater to, and the tasks they support [50, 70].

The reader may refer to the work of Babu et. al [5] for a thorough comparison of AVTs according to the taxonomies they support. A closer inspection of these taxonomies is required, as it is imperative that visualizations are constructed to address problems and issues faced by the users of the system, rather than just provide 'pretty pictures'. The challenge often is that different stakeholders, such as: architects, developers, maintainers, and managers, require contrasting tools and techniques to delve into different levels of details. In the context of software architecture, several researchers, such as: McNair et al. [50] and Panas et al. [52], have conducted in-depth analysis of what to visualize and how best to achieve it. A good synopsis of these findings can be found in the survey of Ghanam et al. [32].

The most significant lesson learnt from the above-mentioned surveys is not to lose sight of the audience and to conduct appropriate evaluations where possible to determine the true worth of a proposed software architecture visualization; does it allow for a more thorough analysis (number of issues detected) or for a more efficient one (task completion time).

5 Conclusion

In this paper, we provided a comprehensive and up-to-date review of both literature and mainstream practices in the field of software architecture visualization. Our research shows that the architecture visualization domain has evolved significantly in recent years giving developers new tools to better understand, evaluate, and develop software and helping managers to monitor design and refactoring issues. However, there remains the need to incorporate these cutting-edge tools and techniques with standard software development and maintenance practices.

Some visualization techniques like parallel coordinates and bundled diagram layouts are less known in industry, while other techniques such as node-link layouts are well known. The software architecture community has not made widespread use of these recent advances. There is a definite need to bridge this gap, as software systems are getting far too large to be analyzed through traditional means alone. This delay in adopting new technology may be due to the stakeholders not having enough time to try out every new tool, lack of knowledge with respect to technical visualization terms often used in marketing these tools, or simply a reluctance to try unknown visualization metaphors and techniques.

The way forward is for researchers to work closely with experts, tailor tools to meet specific requirements, and to conduct comprehensive evaluations. This would lead to research prototypes making their way into mainstream tools and a widespread adoption. It is envisioned that this transition would improve quality and reduce the time and cost factors. Lastly, we would like to point out the need for both industry and academia to look into the evolution of software at higher level of abstraction than current linebased methods; this remains an open area for future research.

References

1 SOFTVIS 2008. ACM Symposium on Software Visualization, September 2008. Online; Accessed 17-November-2011.

2 Sazzadul Alam and Philippe Dugerdil. Evospaces: 3d visualization of software architecture. In *SEKE*, pages 500–505. Knowledge Systems Institute Graduate School, 2007.

3 Keith Andrews and Helmut Heidegger. Information slices : Visualising and exploring large hierarchies using cascading , semi-circular discs. *Information Visualization*, pages 9–12, 1998.

4 Daniel Archambault, Tamara Munzner, and David Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14:900–913, 2008.

5 K. Delhi Babu, P. Govindarajulu, and A.N. Aruna Kumari Ahmed. Development of the conceptual tool for complete software architecture visualization: Darch (da). *International Journal of Computer Science and Network Security (IJCSNS)*, 9(4):277–286, April 2009.

6 Michael Balzer and Oliver Deussen. Hierarchy based 3d visualization of large software structures. In *Proceedings of the conference on Visualization '04*, VIS '04, pages 598.4–, Washington, DC, USA, 2004. IEEE Computer Society.

7 Michael Balzer and Oliver Deussen. Level-of-detail visualization of clustered graph layouts. In Seok-Hee Hong and Kwan-Liu Ma, editors, *APVIS*, pages 133–140. IEEE, 2007.

8 Michael Balzer, Oliver Deussen, and Claus Lewerentz. Voronoi treemaps for the visualization of software metrics. In *Proceedings of the 2005 ACM symposium on Software visualization*, SoftVis '05, pages 165–172, New York, NY, USA, 2005. ACM.

9 Michael Balzer, Andreas Noack, Oliver Deussen, and Claus Lewerentz. Software landscapes: Visualizing the structure of large software systems. In Oliver Deussen, Charles D. Hansen, Daniel A. Keim, and Dietmar Saupe, editors, *VisSym*, pages 261–266. Eurographics Association, 2004.

10 Todd Barlow and Padraic Neville. A comparison of 2-d visualizations of hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, pages 131–, Washington, DC, USA, 2001. IEEE Computer Society.

11 Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. Addison-Wesley, Boston ; Munich [u.a.], 2005.

12 I. Brooks. Object-oriented metrics collection and evaluation with a software process. In *Proc. OOPSLA '93 Workshop Processes and Metrics for Object-Oriented Software Development*, Washington, D.C., 1993.

13 Heorhiy Byelas and Alexandru Telea. Visualizing metrics on areas of interest in software architecture diagrams. In Peter Eades, Thomas Ertl, and Han-Wei Shen, editors, *PacificVis*, pages 33–40. IEEE Computer Society, 2009.

14 David N. Card and Robert L. Glass. *Measuring Software Design Quality*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

15 Pierre Caserta and Olivier Zendra. Visualization of the static aspects of software: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 17:913–933, 2011.

16 Christian Collberg, Stephen Kobourov, Jasvir Nagra, Jacob Pitts, and Kevin Wampler. A system for graph-based visualization of the evolution of software. In *Proceedings of the 2003 ACM symposium on Software visualization*, SoftVis '03, pages 77–ff, New York, NY, USA, 2003. ACM.

17 Raimund Dachselt and Jürgen Ebert. Collapsible cylindrical trees: A fast hierarchical navigation technique. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, pages 79–, Washington, DC, USA, 2001. IEEE Computer Society.

18 Marco D'Ambros and Michele Lanza. Reverse engineering with logical coupling. In *Proceedings of the 13th Working Conference on Reverse Engineering*, pages 189–198, Washington, DC, USA, 2006. IEEE Computer Society.

19 Marco D'Ambros and Michele Lanza. Bugcrawler: Visualizing evolving software systems. In *11th European Conference on Software Maintenance and Reengineering, 2007. CSMR '07.*, pages 333 –334, march 2007.

20 Marco D'Ambros and Michele Lanza. Visual software evolution reconstruction. *J. Softw. Maint. Evol.*, 21:217–232, May 2009.

21 Stephan Diehl. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

22 Stéphane Ducasse, Simon Denier, Françoise Balmas, Alexandre Bergel, Jannik Laval, Karine Mordal-Manet, and Fabrice Bellingard. Visualization of Practices and Metrics (Workpackage 1.2). Research report, Squale Consortium, March 2010.

23 Slawomir Duszynski, Jens Knodel, and Mikael Lindvall. Save: Software architecture visualization and evaluation. In Andreas Winter, Rudolf Ferenc, and Jens Knodel, editors, *CSMR*, pages 323–324. IEEE, 2009.

24 Holger Eichelberger. *Aesthetics and automatic layout of UML class diagrams*. PhD thesis, UniversitÃđt WÃijrzburg, Am Hubland, 97074 WÃijrzburg, 2005.

25 Stephen G. Eick, Joseph L. Steffen, and Eric E. Sumner, Jr. Seesoft – a tool for visualizing line oriented software statistics. In Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors, *Readings in information visualization*, pages 419–430. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

26 Jean-Daniel Fekete, David Wang, Niem Dang, and Catherine Plaisant. Overlaying graph links on treemaps. IEEE Symposium on Information Visualization Conference Compendium (demonstration), Oct 2003.

27 Ronald B. Finkbine, Ph.D. Metrics and models in software quality engineering. *SIGSOFT Softw. Eng. Notes*, 21:89–, January 1996.

28 UML Forum. Uml faq @ONLINE, December 2011.

29 Alexander Fronk, Armin Bruckhoff, and Michael Kern. 3d visualisation of code structures in java software systems. In *Proceedings of the 2006 ACM symposium on Software visualization*, SoftVis '06, pages 145–146, New York, NY, USA, 2006. ACM.

30 Harald Gall, Karin Hajek, and Mehdi Jazayeri. Detection of logical coupling based on product release history. In *Proceedings of the International Conference on Software Maintenance*, ICSM '98, pages 190–, Washington, DC, USA, 1998. IEEE Computer Society.

31 K Gallagher, A Hatch, and M Munro. Software architecture visualization : an evaluation framework and its application. *IEEE Transactions on Software Engineering*, 34(2):260–270, 2008.

32 Y. Ghanam and S. Carpendale. A survey paper on software architecture visualization. *Technical Report, University of Calgary*, pages 1–10, June 2008.

33 Hamish Graham, Hong Yul Yang, and Rebecca Berrigan. A solar system metaphor for 3d visualisation of object oriented software metrics. In *Proceedings of the 2004 Australasian symposium on Information Visualisation – Volume 35*, APVis '04, pages 53–59, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

34 A. Hatch. *Software Architecture Visualization*. Phd dissertation, University of Durham, 2004.

35 D. Holten, R. Vliegen, and J. J. van Wijk. Visual realism for the visualization of software metrics. In *Proceedings of the 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis*, VISSOFT '05, pages 12–, Washington, DC, USA, 2005. IEEE Computer Society.

36 Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12:741–748, September 2006.

37 Danny Holten and Jarke J. van Wijk. Visual comparison of hierarchically organized data. *Computer Graphics Forum*, 27(3):759–766, 2008.

38 Warwick Irwin and Neville Churcher. Object oriented metrics: Precision tools and configurable visualisations. In *Proceedings of the 9th International Symposium on Software Metrics*, pages 112–, Washington, DC, USA, 2003. IEEE Computer Society.

39 Brian Johnson and Ben Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd conference on Visualization '91*, VIS '91, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

40 T.A. Keahey. A brief tour of nonlinear magnification @ONLINE, November 2011.

41 Andreas Kerren, Achim Ebert, and Jörg Meyer, editors. *Human-Centered Visualization Environments*. Lecture Notes in Computer Science, LNCS. Springer-Verlag Gmbh, 1 edition, 2007.

42 Andreas Kerren and Ilir Jusufi. Novel visual representations for software metrics using 3d and animation. In Jürgen Münch and Peter Liggesmeyer, editors, *Software Engineering (Workshops)*, volume 150 of *LNI*, pages 147–154. GI, 2009.

43 John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

44 Guillaume Langelier, Houari Sahraoui, and Pierre Poulin. Exploring the evolution of software quality with animated visualization. In *Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, VLHCC '08, pages 13–20, Washington, DC, USA, 2008. IEEE Computer Society.

45 Michele Lanza. The evolution matrix: recovering software evolution using software visualization techniques. In *Proceedings of the 4th International Workshop on Principles of Software Evolution*, IWPSE '01, pages 37–42, New York, NY, USA, 2001. ACM.

46 Michele Lanza and Stéphane Ducasse. Polymetric views-a lightweight visual approach to reverse engineering. *IEEE Trans. Softw. Eng.*, 29:782–795, September 2003.

47 M. M. Lehman and L. A. Belady, editors. *Program evolution: processes of software change*. Academic Press Professional, Inc., San Diego, CA, USA, 1985.

48 J.D. Mackinlay. Opportunities for information visualization. *Computer Graphics and Applications, IEEE*, 20(1):22 –23, jan/feb 2000.

49 Andrian Marcus, Louis Feng, and Jonathan I. Maletic. Comprehension of software analysis data using 3d visualization. In *Proceedings of the 11th IEEE International Workshop on Program Comprehension*, IWPC '03, pages 105–, Washington, DC, USA, 2003. IEEE Computer Society.

50 Andrew McNair, Daniel M. German, and Jens Weber-Jahnke. Visualizing software architecture evolution using change-sets. In *Proceedings of the 14th Working Conference on Reverse Engineering*, pages 130–139, Washington, DC, USA, 2007. IEEE Computer Society.

51 T Panas, R Berrigan, and J Grundy. A 3d metaphor for software production visualization. *Proceedings on Seventh International Conference on Information Visualization 2003 IV 2003*, 314:314–319, 2003.

52 Thomas Panas, Thomas Epperly, Daniel Quinlan, Andreas Saebjornsen, and Richard Vuduc. Communicating software architecture using a unified single-view visualization. In *Proceedings of the 12th IEEE International Conference on Engineering Complex Computer Systems*, pages 217–228, Washington, DC, USA, 2007. IEEE Computer Society.

53 M. Petre and E. Quincey. A gentle overview of software visualization. *PPIG News Letter*, pages 1–10, September 2006.

54 Martin Pinzger, Harald Gall, Michael Fischer, and Michele Lanza. Visualizing multiple evolution metrics. In *Proceedings of the 2005 ACM symposium on Software visualization*, SoftVis '05, pages 67–75, New York, NY, USA, 2005. ACM.

55 Martin Pinzger, Katja Graefenhain, Patrick Knab, and Harald C. Gall. A tool for visual understanding of source code dependencies. In *Proceedings of the 2008 The 16th IEEE International Conference on Program Comprehension*, ICPC '08, pages 254–259, Washington, DC, USA, 2008. IEEE Computer Society.

56 Helen C. Purchase, Jo-Anne Allder, and David A. Carrington. User preference of graph layout aesthetics: A uml study. In *Proceedings of the 8th International Symposium on Graph Drawing*, GD '00, pages 5–18, London, UK, 2001. Springer-Verlag.

57 Werner Randelshofer. Visualization of large tree structures @ONLINE, November 2011.

58 Jun Rekimoto and Mark Green. The information cube: Using transparency in 3d information visualization. In *In Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS'93*, pages 125–132, 1993.

59 Neeraj Sangal, Ev Jordan, Vineet Sinha, and Daniel Jackson. Using dependency models to manage complex software architecture. In *Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, OOPSLA '05, pages 167–176, New York, NY, USA, 2005. ACM.

60 C. Russo dos Santos, P. Gros, P. Abel, D. Loisel, N. Trichaud, and J. P. Paris. Metaphor-aware 3d navigation. In *Proceedings of the IEEE Symposium on Information Vizualization 2000*, INFOVIS '00, pages 155–, Washington, DC, USA, 2000. IEEE Computer Society.

61 Amit P. Sawant and Naveen Bali. Diffarchviz: A tool to visualize correspondence between multiple representations of a software architecture. *Visualizing Software for Understanding and Analysis, International Workshop on*, 0:121–128, 2007.

62 Hans-Jorg Schulz, Steffen Hadlak, and Heidrun Schumann. The design space of implicit hierarchy visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 17:393–411, April 2011.

63 Hans-Jorg Schulz and Heidrun Schumann. Visualizing graphs – a generalized view. In *Proceedings of the conference on Information Visualization*, pages 166–173, Washington, DC, USA, 2006. IEEE Computer Society.

64 John Stasko. An evaluation of space-filling information visualizations for depicting hierarchical structures. *Int. J. Hum.-Comput. Stud.*, 53:663–694, November 2000.

65 John Stasko and Eugene Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proceedings of the IEEE Symposium on Information Vizualization 2000*, INFOVIS '00, pages 57–, Washington, DC, USA, 2000. IEEE Computer Society.

66 Frank Steinbrückner and Claus Lewerentz. Representing development history in software cities. In *Proceedings of the 5th international symposium on Software visualization*, SOFTVIS '10, pages 193–202, New York, NY, USA, 2010. ACM.

67 Margaret-Anne Storey, Casey Best, and Jeff Michaud. Shrimp views: An interactive environment for exploring java programs. *The 9th International Workshop on Program Comprehension*, 0:111–112, 2001.

68 Dabo Sun and Kenny Wong. On evaluating the layout of uml class diagrams for program comprehension. In *Proceedings of the 13th International Workshop on Program Comprehension*, pages 317–326, Washington, DC, USA, 2005. IEEE Computer Society.

69 Alexandru Telea and David Auber. Code flows: Visualizing structural evolution of source code. *Comput. Graph. Forum*, 27(3):831–838, 2008.

70 Alexandru Telea, Lucian Voinea, and Hans Sassenburg. Visual tools for software architecture understanding: A stakeholder perspective. *IEEE Software*, 27:46–53, 2010.

71 M. Termeer, C. F. J. Lange, A. Telea, and M. R. V. Chaudron. Visual exploration of combined architectural and metric information. In *Proceedings of the 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis*, VISSOFT '05, pages 11–, Washington, DC, USA, 2005. IEEE Computer Society.

72 Lucian Voinea and Alexandru Telea. Multiscale and multivariate visualizations of software evolution. In *Proceedings of the 2006 ACM symposium on Software visualization*, SoftVis '06, pages 115–124, New York, NY, USA, 2006. ACM.

73 Lucian Voinea, Alexandru Telea, and Michel R. V. Chaudron. Version-centric visualization of code evolution. In Ken Brodlie, David J. Duke, and Kenneth I. Joy, editors, *EuroVis*, pages 223–230. Eurographics Association, 2005.

74 Weixin Wang, Hui Wang, Guozhong Dai, and Hongan Wang. Visualization of large hierarchical data by circle packing. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 517–520, New York, NY, USA, 2006. ACM.

75 Colin Ware. *Information visualization: perception for design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

76 Richard Wettel and Michele Lanza. Visualizing software systems as cities. In Jonathan I. Maletic, Alexandru Telea, and Andrian Marcus, editors, *VISSOFT*, pages 92–99. IEEE Computer Society, 2007.

77 Richard Wettel and Michele Lanza. Visual exploration of large-scale system evolution. In *Proceedings of the 2008 15th Working Conference on Reverse Engineering*, pages 219–228, Washington, DC, USA, 2008. IEEE Computer Society.

78 Dirk Zeckzer. Visualizing software entities using a matrix layout. In *Proceedings of the 5th international symposium on Software visualization*, SOFTVIS '10, pages 207–208, New York, NY, USA, 2010. ACM.

Improving Safety-Critical Systems by Visual Analysis

Yi Yang[1], Patric Keller[1], Yarden Livnat[2], and Peter Liggesmeyer[1]

1 **Software Eningeering: Dependability Group**
University of Kaiserslautern, Germany
{yang,pkeller,liggesmeyer}@cs.uni-kl.de
2 **Scientific Computing and Imaging Institute**
University of Utah, USA
yarden@sci.utah.edu

Abstract

The importance analysis provides a means of analyzing the contribution of potential low-level system failures to identify and assess vulnerabilities of safety-critical systems. Common approaches attempt to enhance the system safety by addressing vulnerabilities using an iterative analysis process, while considering relevant constraints, e.g., cost, for optimizing the improvements. Typically, data regarding the analysis process is presented across several views with few interactive associations among them. Consequently, this hampers the identification of meaningful information supporting the decision making process. In this paper, we propose a visualization system that visually supports engineers in identifying proper solutions. The visualization integrates a decision tree with a plot representing the cause-effect relationship between the improvement ideas of vulnerabilities and the resulting risk reduction of system. Associating a component fault tree view with the plot allows to maintain helpful context information. The introduced visualization approach enables system and safety engineers to identify and analyze optimal solutions facilitating the improvement of the overall system safety.

1998 ACM Subject Classification B.4.5 Reliability, Testing, and Fault-Tolerance, I.3.8 Applications, D.2.4 Software/Program Verification

Keywords and phrases fault tree analysis, importance and sensitivity analysis, information visualization, decision tree, safety analysis

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.43

1 Introduction

Fault tree analysis is a widely used technique for the identification of vulnerabilities of safety-critical systems. This analysis uses a graphical model called fault tree to logically relate undesired failures at the system level (called top event) with failures at the component level (named basic events). A component fault tree is an advanced modularization concept supporting the fault tree analysis of complex systems. This allows to extend the regular fault tree model by decomposing it according to the architecture of the system under investigation into a hierarchical representation where each component is represented by an extended fault tree. The fault tree analysis provides a basis of the importance analysis and sensitivity analysis of those failure relations. It mainly focuses on the risk contributions of individual basic events to a top event. The important basic events represent the critical vulnerabilities of a system. Sensitivity analysis is applied to investigate relations between changes of basic events and the resulting impacts on a top event.

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 43–58

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In order to improve the system safety, engineers usually carry out an iterative risk reduction approach consolidating importance and sensitivity analysis. As a result of the approach, engineers may identify an improvement solution consisting of a group of modifications with respect to system design. By a solution, the failure probability of top event is reduced to an acceptable level. In many cases, engineers may identify multiple possible solutions by various alternative design modifications in the analysis process. Thus, the safety improvement process consists of the determination of modifications and the review of solutions by taking the essential questions into account:

- Aspects of modifications:
  - What are the most important basic events contributing to a system failure?
  - What are possible modifications of the system design?
  - What are the impacts of the modifications regarding system safety?
  - Which modifications are optimal taking certain constraints into consideration?
- Aspects of solutions:
  - How good are the improvement solutions?
  - What is the best solution?

Usually, the data related to the questions is separated across individual views having various representation forms, e.g., fault trees, tables, histograms, plots, and decision trees. However, there are few interactive associations among the views. Mostly, engineers need to frequently switch views for accessing meaningful data during the analysis process. Additionally, there is no sufficient context information when engineers focus on a specific view. For example, modifications are organized using a decision tree and the detailed data of the queried modification is represented in a separate table. When focusing on the table of detail information, the context with respect to the overview of modifications may be lost. Furthermore, when analyzing the basic event corresponding to this modification, engineers need to manually locate the basic event in the fault tree view because the decision tree does not provide this information. Engineers spend more additional efforts for switching views and identifying significant information.

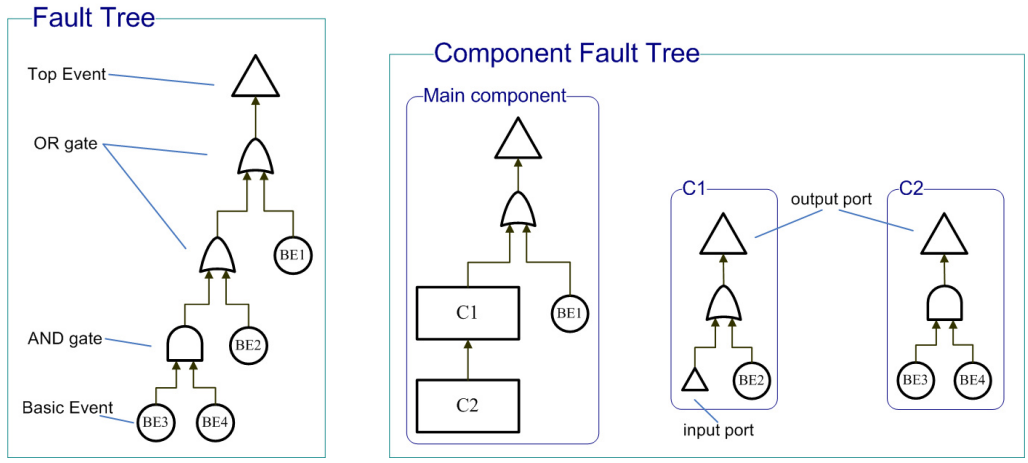
In this paper, we propose a visualization system that effectively integrates data which is essential for the analysis of the safety improvement process. To support the information access within different contexts, we additionally provide suitable interaction possibilities. The proposed visualization system facilitates to identify and analyze vulnerabilities of safety-critical systems, as well as determine the optimal/appropriate solution(s) by simulating system design modifications on an abstract level.

The remainder of the paper is organized as follows: Section 2 describes the basic principles of the (component) fault tree analysis, importance and sensitivity analysis, as well as the related representation concepts. We introduce our visualization system in Section 3. We review the proposed methods on the basis of a short application example in Section 4. The conclusion is subject to Section 5.

2 Background

2.1 Safety Analysis

The term *safety* often refers to a state of a system where the danger of a personal injury or property damage lies within an acceptable level [15, 13, 4, 24]. A *failure* is defined as an inconsistent behavior that deviates from the given specification of a system or a component



(a) Fault tree **(b)** Component fault tree

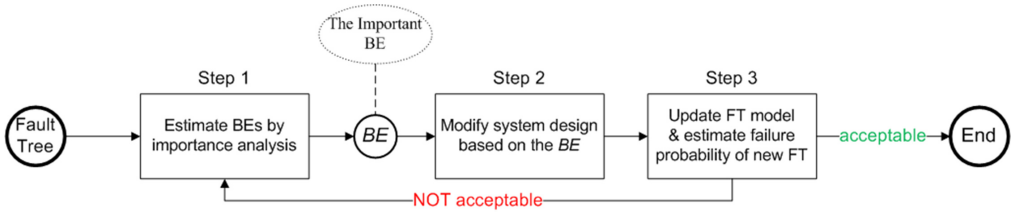
Figure 1 Fault tree and component fault tree. (a) A fault tree consisting of four basic events connected by an AND-gate and two OR-gates. (b) The component fault tree model based on the fault tree in (a), which contains a main component and sub component "C1" and "C2". "C2" inputs its failure to "C1" via ports.

[4, 24]. In this context a system is said to be *safety-critical* if the failure of the system could cause consequences that harm people [18, 13, 31, 24]. *Risk* is a combination of the frequency of a harmful failure and the severity of the harm caused by that failure [13, 24]. When talking about *safety analysis* we often refer to a process whose goal is to provide a reliable assessment and improvement of the risk of a safety-critical system [19, 20, 24]. To achieve this a variety of methods and techniques exist, e.g., fault tree analysis.

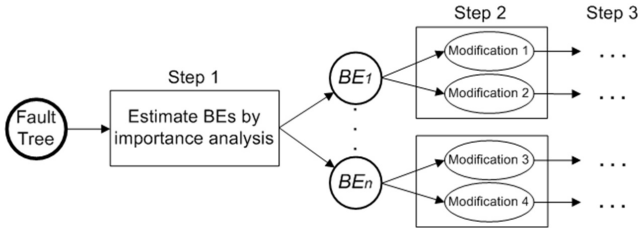
2.2 Fault Tree Analysis

Fault tree analysis (FTA) [13, 36, 35, 12] is a deductive method allowing to trace the causes of an undesired system state back to its roots. The method is based upon the usage of so called *fault trees (FTs)*. A fault tree is a tree-like structure composed of different types of nodes. The root of the tree termed *top event* represents the undesired system state (e.g., system failure or outtake). The leaves are *basic events (BEs)* that represent the low-level failures which are connected by logic gates, such as "AND-gate"and "OR-gate". The way the leaf nodes are connected reflects how the low-level failures logically contribute to the undesired system state (see Figure 1(a)).

The ordinary modularization concept of fault tree allows to partition the independent sub-trees as modules. However, these modules are not be mapped to identify technical components of the system design. To solve this issue, Kaiser et al. [17] proposed an advanced modeling concept called *component fault trees (CFTs)*. Technical components of a system are represented as the corresponding CFT components in the component fault tree model. The influences between technical components are transferred via in- and out-ports of CFT components. In this way, engineers may treat each CFT component as a black-box. Figure 1 (b) shows an example of the concept of component fault tree. The component fault tree modularizes the sub-trees as CFT components and replaces the sub-trees with rectangles in the main component. The detailed sub-trees are separately represented in individual views.



(a) Procedure of single solution construction



(b) Multiple solutions construction

Figure 2 Construction of improvement solutions.

2.3 Importance and Sensitivity Analysis

Importance analysis and sensitivity analysis are quantitative approaches for evaluating (component) fault trees. Vesely et al. [35] suggested that, in general, more than 90% of the failure probability of a top event is due to less than 20% of the basic events. This implies that we only need to focus on a small subset of basic events having major contribution. To identify those, we determine the importance of each basic event with regard to the failure probability of the top event. Importance analysis considers both the failure probability and the logical relations of basic events. The Fussell-Vesely (FV) importance measure [11, 27] assigns each basic event an importance value between zero and one: the larger the value, the more important the basic event in terms of influence on the top event. The sum over the importance of all basic events of a system may be greater than one since, in some cases, simultaneous failures of multiple sub-systems may cause the system failure [10]. On the other hand, the sensitivity analysis investigates the resulting impact of changes applied to the basic events on the top event [14, 35, 23]. It is used for analyzing the accuracy of basic events as well as the effects of safety improvements [25, 7, 9].

2.4 Improvement of Safety-Critical Systems

The improvement of the system safety may necessitate design modification involving the replacement of critical parts of the system by elements having a better failure performance (substitution concept) or introduction of identical redundant parts (redundancy concept). Finding a satisfying solution in general is a non-trivial task underlying constraints and restriction for which formal methods are not always available. Generally, the procedure associated with this approach iteratively applies a set of alternative modifications until the complete solution is found, which reduces the risk of a system to an acceptable level. Taking the results obtained from the improvement analysis into account, it is possible to derive such solutions in a more guided fashion [7, 9, 8, 35, 25]. Each iteration consists of mainly three steps (see Figure 2 (a)):

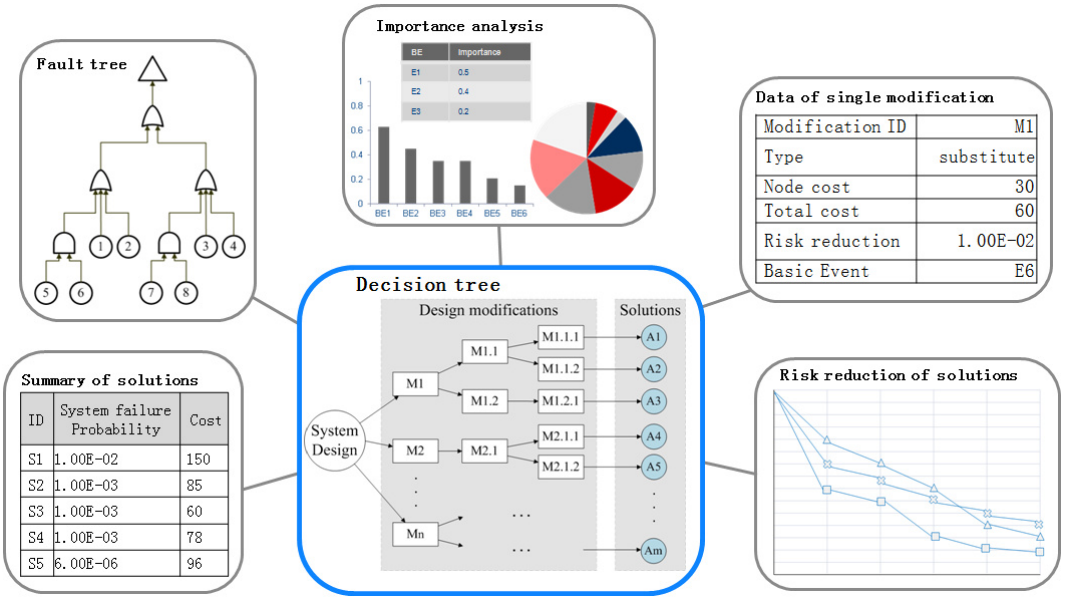


Figure 3 Ordinary representations used by the analysis of safety improvement process. Improvement solutions are arranged by a decision tree [9]. The relevant data is distributed across several views [1, 7, 32, 9]. Commonly used representations are fault trees, charts or tables to show importance of basic events, and the summary of possible solutions, design modifications, and individual risk reductions.

1. Perform the importance analysis to identify the basic event having the largest contribution.
2. Find the hardware component related to that basic event. Modify the system design by replacing the component by another one featuring a better quality or by introducing identical ones in order to increase redundancy.
3. Update the (component) fault tree model and assess the modification with respect to the impact on the top event in terms of reduction of failure probability. If required, engineers may determine the optimal modification under consideration of additional constraints, e.g., the costs of the modifications. If a complete solution is found, i.e., the failure probability of the top event is reduced to the goal value, stop the process, otherwise start next iteration from step 1.

Constructing a solution necessitates to choose the proper basic event, and to decide for a suitable design alternative. In many cases, multiple improvement solutions exist because of multiple important basic events or/and various alternative design modification ideas corresponding to the identical basic event (see Figure 2 (b)). After constructing solutions, engineers may additionally identify the optimal one. An important assumption of the safety improvement process is that basic events are stochastically independent so that the change of a basic event does not influence other basic events.

2.5 Related Representation Concepts

Fault tree analysis tools provide the view on fault trees using standardized graphical symbols (see Figure 1 (a)). The data of the fault tree, e.g., failure probability of the top events and the basic events are represented by text or data-aggregated forms. Most fault tree analysis tools [16, 28, 1, 7, 32, 9] summarize the importance of basic events using a data table. Faulttree+ [16] shows importance values in a table associated with table presenting the properties of

events, e.g., failure probability. Relex Architect [28] provides a table in which users may filter and show the importance of basic events belonging to a specific sub tree. RAMCommander [1] additionally provides charts for the importance values, e.g., histogram, pie-chart, and 2D/3D scatter plot. BlockSim [29] assigns colors to the histogram according to the failure probability of basic events. Additionally, BlockSim proposed a variant of pie-chart called "square pie-chart" that anti-clockwise arranges the basic events in descending order with respect to the importance values. Project CISA [7, 9] arranges data of design modifications in separate views and logically links them to a decision tree that represents the summary of improvement solutions (Figure 3).

A decision tree is a tree-based predictive model that is widely used for facilitating the decision-making in many domains. It partitions a data set into subsets according specific rules. The root represents the original data, the edges represent the partitioning rules, and the non-root nodes represent the outcomes of different rules. To construct a decision tree, the users need to quickly identify the nodes to be partitioned by navigating through the tree. Decision tree is a good way to provide overviews about complex decision-making process. Ankerst et al. [2, 3] applied an indentation diagram to represent a decision tree for arranging the partitioning steps in data mining. The work [34] integrated decision trees and data visualizations of attributes of each node for purposes at data classification. Pham et al. [26] presented a decision tree using the sunburst layout to visualize machine-learning algorithms. The decision trees represented by an icicle diagram were provided by the work [21, 3, 6]. The icicle concept represented tree hierarchies without wasting display space. Project PaintingClass [33] integrated parallel coordinates and star coordinates with a decision tree for exploring classified multi-dimensional data. Barlow et al. [6] proposed a visualization system that linked views of various decision tree layouts to represent the decision data of data mining process.

3 Visualization for Safety Improvement

3.1 Requirements of Analysis Process

The safety improvement process of a system concentrates on two phases: construction of solutions and review of solutions. Requirements of the analysis (in short "R") are summarized as follows:

- Construction of solutions: A solution comprising a sequence of design modifications. It is constructed by performing an iterative analysis procedure (see Figure 2). The steps are:
  - Step 1: identify the the important basic events (R1).
  - Step 2: apply and test the risk reduction hypothesis by different modifications.
    - R2: identify the type of modification: substitution or redundancy.
    - R3: identify the value of modification: change of failure probability of the initial basic event.
    - R4: identify the cost of modification.
  - Step 3: evaluate the results of risk reduction:
    - R5: evaluate the update of the component fault tree model.
    - R6: evaluate the impact of top event by design modification.
    - R7: evaluate the cost-effectiveness of modification.
    - R8: evaluate the gap between updated failure probability of top event and the goal value.
- Review of solutions: Reviewing the constructed solutions may facilitate the understanding of solutions and determining the optimal ones resulting from the following requirement.

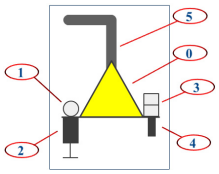


Figure 4 Risk-state node for a design modification. (0) Risk state of top event. Color indicates the level of failure probability of the top event. (1) Type of the modification. Circle indicates substitution concept, while small triangle indicates redundancy concept. (2) Change of failure probability of the corresponding important basic event. (3) Cost of the modification. (4) Cost-effectiveness of the modification. (5) An edge connecting the node with its predecessor node. The vertical part represents the resulting reduction of failure probability of the top event.

3.2 Visual Support for Construction of Solutions

3.2.1 Representing Design Modification

Along with performing the safety improvement process, the design modifications are sequentially connected as a decision tree to construct one or more solutions (see Figure 3). A branch of the decision tree is caused by either multiple important basic events or multiple modification ideas. We finally apply the node-link diagram for representing the decision tree by taking two points into consideration: readability and data integration. Barlow et al. [5] evaluated the readability of the treemap layout, the sunburst layout, the node-link diagram, and the icicle diagram. The authors conducted that the node-link diagram and the icicle diagram were the most favorable for representing the tree structure data. The node-link diagram has the sufficient space to integrate the visual attributes in nodes. However, the icicle diagram (as well as treemap and sunburst layout) is a compact layout in which the aspect ratio needs to be maintained for the semantic meaning. In this case, the nodes on the deeper hierarchy of the tree do not have sufficient space for representing the attributes. In sum, the node-link diagram is more appropriate for our decision tree than other layouts.

For understanding a modification, the cause (i.e., corresponding basic event) and the effect (i.e., resulting risk state of system) is the primary information. In order to represent the cause-effect relation, we place the modification nodes of the decision tree in a risk-reduction plot (see Figure 5 (2)) where x-axis represents ordinal basic events, while the achieved change in risk (in terms of failure probability) is projected along the y-axis that represents a range from the initial failure probability of a top event to the goal value in a top-down direction.

We then introduce the visualization properties of the node-link decision tree. In order to represent the associated significant data, for each modification, we propose a risk-state node that consists of a central triangle icon and four attached visual items representing data associated with the modification (see Figure 4).

- **Triangle icon**: shows the risk state of the system corresponding to the modification. This is the most significant data based on the updated component fault tree (with respect to R5). In many cases, the safety and system engineers intend to quickly and roughly estimate the change of risk of a system, e.g., by which step the failure probability is reduced from critical level to moderate level. The shape of triangle is applied because it is consistent with the shape of the top event of the component fault tree. Color is recommended for representing the ordinal data by the work at [22]. The color of the triangle depends on the level of failure probability described in Section 3.2.3. Using colors, one may quickly estimate the criticality of the top event, and decide whether the failure probability is acceptable or not. When the color becomes green, the risk reduction can

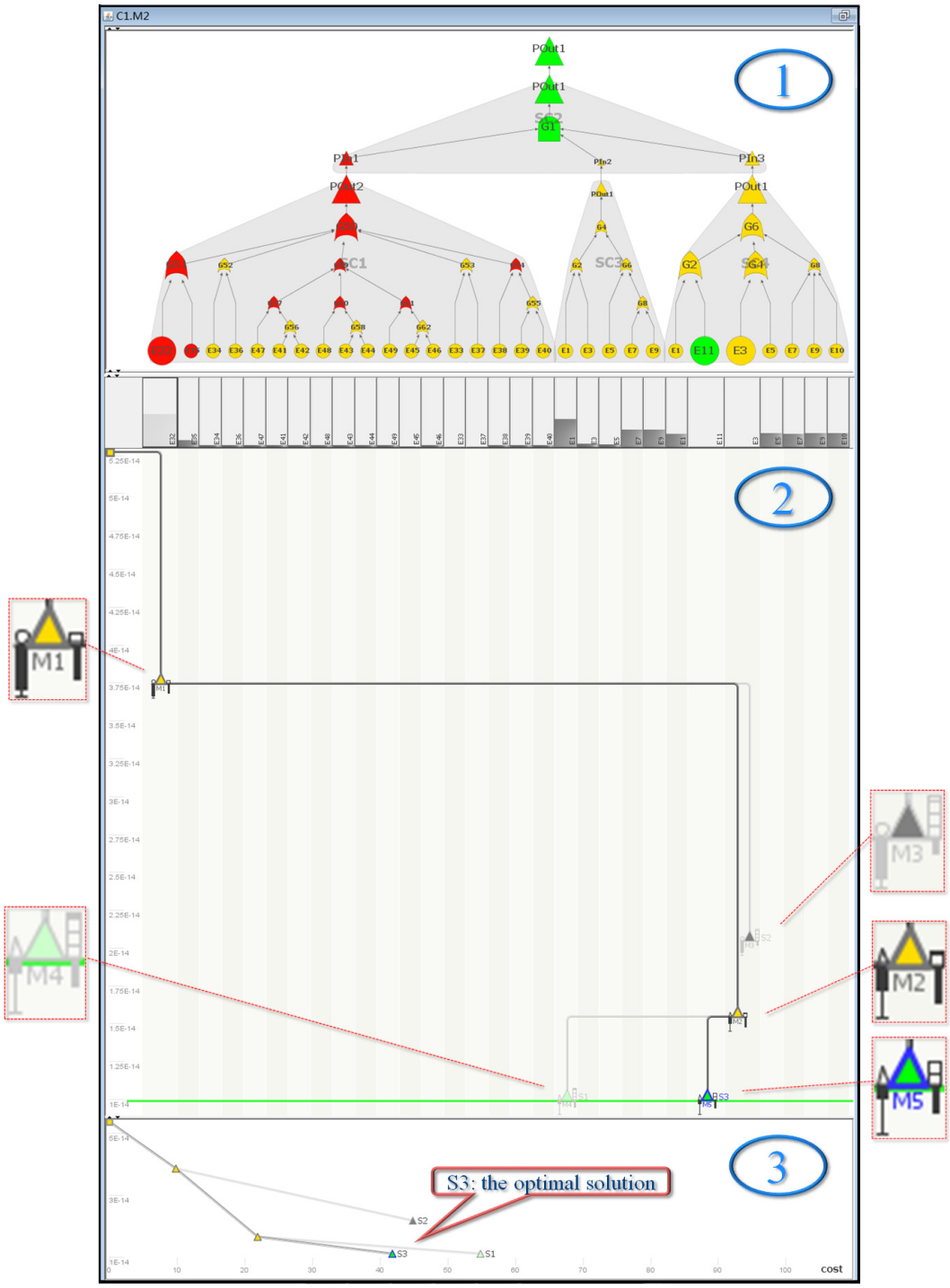


Figure 5 Visualization system for improving system quality with respect to safety. (1) The associated component fault tree view. (2) The risk-reduction plot. (3) The solution overview plot.

be finished and the corresponding solution is complete. Additionally, the label of the corresponding modification is presented below the icon.

- Item 1: the type of modification (with respect to R2). The types are nominal data that may be effectively represented by the graphical properties of position, color, texture, connection, density, and shape. The graphical properties of position, color and connection are already used in our visualization. Taking the size of the triangle icon into account, the graphical property of density is not suitable, too. Thus, we apply shape for representing the basic types of design modifications: a circle represents a component substitution whereas a small triangle represents the introduction of redundant components.
- Item 2: the reduction of the failure probability of the original basic event (with respect to R3). If engineers replace the initial hardware, the value of the new part becomes current. If engineers apply the redundancy concept, the new value is the failure probability of the new sub-tree of the redundant parts. The difference between the failure probability of the original basic event and the new basic event (or sub-tree) introduced represents the improvement of the vulnerability being addressed. To present this information in an intuitive way we have designed a bar graph using the graphical property of length that is recommended for representing the quantitative data [22]. The bottom line of the bar indicates the failure probability of the initial basic event. The filled part shows the new value. The item provides information about the context under which the modification has been applied. For example, following the substitution approach, it is possible to intuitively compare the existing with the new part in terms of failure probability.
- Item 3: the cost of modification (with respect to R4). In our work, the cost is an value representing a quantity consumed for the modification, e.g., money, time, and human-resources. The type of the cost needs to be defined at the beginning of the safety improvement process. It is an important information for evaluating design modifications (see Section 3.2.2) and solutions (see Section 3.3). We propose a scale bar to visualize the cost not only for the comparison of cost of modifications but also for the investigation of the absolute cost value. Engineers are allowed to define the scale of the bar, e.g., each box represents 10 dollars.
- Item 4: the cost-effectiveness ratio of a modification (with respect to R7). In cases where multiple design modifications exist it is important to choose those providing the proper balance between risk reduction and cost (see Section 3.2.2). We use the graphical property of length to represent the quantitative cost-effectiveness. Thus, a bar is introduced to represent the cost-effectiveness ratio for a given design modification. The larger the bar, the more cost-effective the modification.

There are two possible ways to composite the central icon and the visual items: the inside strategy and the outside strategy. When placing the visual items inside the central icon, the icon needs to be enlarged. In this case, the large icon cannot exactly indicate its position in the plot that represents significant semantic meaning of the analysis process. Thus, we apply the outside composition strategy. We place the four visual items closely around the central icon. The visualization properties corresponding to the method of the modification (items (1) and (2)) are placed at the left; the factors of evaluation of the modification are represented at the right (items (3) and (4)). This way, engineers may investigate the method and evaluation of modification in the corresponding side.

We connect a new risk-state node with its direct predecessor using a two-part orthogonal edge. A line between the predecessor node and the horizontal position of the new risk-state node represents the subsequent design modification. The vertical part of that line represents the reduction of the failure probability resulting from modification (with respect to R6).

This provides a reliable basis for guiding the analysis process. When there are alternative modifications for a basic event, multiple risk-state nodes are created. Between two nodes, there is an even distance dividing the width of the x-axis scale of the basic event (see Figure 5: "M2" and "M3"). This may address the overlapping issues of edges as well as of nodes.

To conveniently identify important basic event(s) in each iteration of the analysis procedure (with respect to R1), we present bars on a list of indicators of basic events along the x-axis on top of the risk-reduction plot (the more important a basic event, the longer the bar). Additionally, we provide a horizontal green line in the lower part of the plot for indicating the goal value. This enables us to assess the distance from the goal (with respect to R8).

3.2.2 Identifying optimal Modifications

The decision tree of the analysis process may exponentially grow because of its number of branches. Consequently, engineers might spend much efforts for analyzing a large set of modifications. In this case, engineers need to identify the optimal design modification(s) in each iteration of the process in order to effectively construct adequate solutions (with respect to the step 3 of the analysis procedure). The commonly used criterion is the maximal cost-effectiveness of the modification (referring to visual item (4)). Engineers may alternatively apply the criteria with respect to the largest reduction of failure probability of the top event (referring to the vertical position of risk-state node). The non-optimal modifications may be refused leading to the termination of the corresponding branches. We assign black color to fill up the risk-state node of the modification. This way, one can easily realize that the modification was considered and has been refused.

3.2.3 Adapting Component Fault Trees

Fault trees provide meaningful information for the safety improvement process. We apply the component fault tree in our visualization system instead of the ordinary fault tree because the component fault tree additionally provides the possibility to link failure mechanisms with the elements/components of the system design. According to the definition of the component fault tree, a CFT component reflects an architectural component of the system model in the design phase. This supports the identification of the vulnerable parts of the system design corresponding to the important basic events identified. Additionally, the structure of component fault tree supports the understanding of the effects of modifications along the way a failure propagates through the system when reviewing solutions.

We provide a visually enhanced component fault tree view for supporting the safety improvement process (see Figure 5 (1)). In order to associate the component fault tree view with the risk-reduction plot, we project the ordinal data of the x-axis of the plot (i.e., basic event list) according to their locations within the component fault tree view. This allows to link information from both views. We provide interaction mechanism on the component fault tree view in order to dynamically show the sub-trees of the desired CFT components. Each sub-tree is arranged inside a gray blob that indicates the scope of the CFT component. Our system automatically updates the component fault tree model in the background during the analysis process. In order to quickly assess the updated failure probabilities of nodes of the modified component fault trees, we propose a qualitative estimation method to classify failure probabilities into three levels and assign them colors: critical level (red), moderate level (yellow), and acceptable level (green).

In order to preserve the overview about the vulnerable basic events addressed in a solution, we maintain the initial structure of the component fault tree during the safety improvement

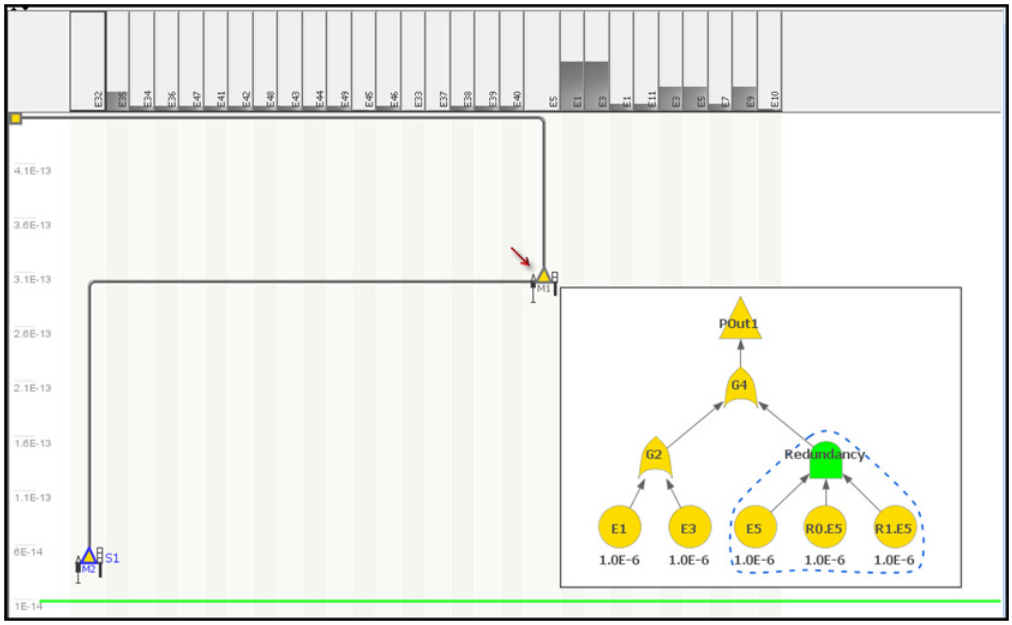


Figure 6 Pop-up window shows the updated logical structure of the CFT component with respect to the modification "M1". The new created sub-tree is arranged in a scope having a dotted border. Two basic events were added and connected with the initial basic event using an AND-gate.

process. That means, by modification performed according to the redundancy concept, the identified important basic events are not directly replaced by sub-trees. Instead, we adapt the color of the initial basic event node with respect to the failure probability of either the substitutional part or the new sub-tree of redundant parts. This can avoid disturbances caused by subsequently updating the component fault tree.

In case engineers intend to review the modified structure of the component fault tree of the specific design modification, our visualization system allows them to show a pop-up view representing the updated logical structure by a right-click on a risk-state node. Instead of displaying the whole component fault tree, the view only presents the structure of the CFT component that contains the basic event related to the design modification. The adapted part of the component fault tree is arranged in a scope indicated by a dashed border. This enables us to intuitively and flexibly view the adapted structures of the component fault tree. This is particularly useful for reviewing design modifications utilizing the concept of redundancy. For example, Figure 6 shows the adaption of a CFT component by a modification. A parallel redundancy is applied by adding two new homogeneous parts and connecting with the initial basic event by an AND-gate.

3.3 Visual Support for the Review of Solutions

While the risk-reduction plot supports the construction of improvement solutions, the overview of the solutions is not intuitive for analyzing the cost-related patterns of the proposed solutions. Such as the trend of risk reduction and of cost increase. It is not suitable for identifying the optimal solutions having the minimum total cost. Thus, we provide a simple and effective plot to present an overview about these quantities (Figure 5 (3)). The x-axis and the y-axis respectively represent the cost of modifications and the failure probability of the top event. We present a triangular node on the overview plot for each

modification. We provide a brushing-and-linking interaction between the risk-reduction plot and the overview plot in order to simultaneously highlight the information associated with the same design modification. Engineers may obtain an intuitive summary of solutions and identify the optimal ones.

We focus on the reduction up to the goal value rather than the exhaustive risk reduction. The overloaded reduction may lead to a large improvement, however, simultaneously also take large costs. In our work, we assume that all complete solutions reduce the initial risk to the same goal value. In this case, for estimating a solution, we consider the total costs of a solution instead of the total cost-effectiveness because this has the identical risk reduction effects to other complete solutions.

4 Application Example

We provide an example intended to illustrate the use of our system with respect to two important aspects: construction of solutions and the review of existing solutions. The applied data originates from a component fault tree of a safety-critical sub-system of an autonomous mobile robot [30]. This model contains 30 basic events and 4 CFT components. The goal is to identify the most cost-effective solution. The initial failure probability of the top event amounts "1.2e-13", the specified acceptable value is "1e-14".

4.1 Construction Process

The construction process of the improvement solutions consists of three iterations that are illustrated in Figure 5 and described as follows:

- Iteration 1:
  - Step 1: Identifying of the important basic event(s). We identify the important basic event by examining the bars on the indicators of the risk-reduction plot. The basic event "E32" proves to be more important than others.
  - Step 2: Applying design modifications. By viewing the labels of the blobs in the component fault tree view, we know that the basic event belongs to the CFT component "SC1". According to this, we may easily identify the corresponding hardware component of the system. Based on experience, we decide to replace the identified hardware component with a new part. The cost of this modification amounts to 10 units (see Figure 5: modification "M1"). The structure of the component fault tree is automatically updated according to the modifications performed and a new risk-state node appears on the risk-reduction plot. A solution "S1" is being constructed starting from this modification. By having a closer look at the solution, we come to the conclusion that the overall failure probability is not acceptable yet because the color of the node is not green. Thus, we start the next iteration.
- Iteration 2:
  - Step 1: Identifying of the important basic event(s). The basic event "E3" is identified as the important one.
  - Step 2: Applying design modifications. There are two possible ways to modify the system design for addressing this basic event. One is to add an homogeneous redundant component causing the costs of 11 units (see Figure 5: "M2"). Another one is to use a substitute causing the costs of 34 units (see Figure 5: "M3"). Because of the branches of the modification ideas, a new solution "S2" appears for the branch of "M3".
  - Step 3: Evaluating the modifications. We compare the cost-effectiveness bars of both created risk-state nodes (referring to item (4)). The modification "M2" is obviously

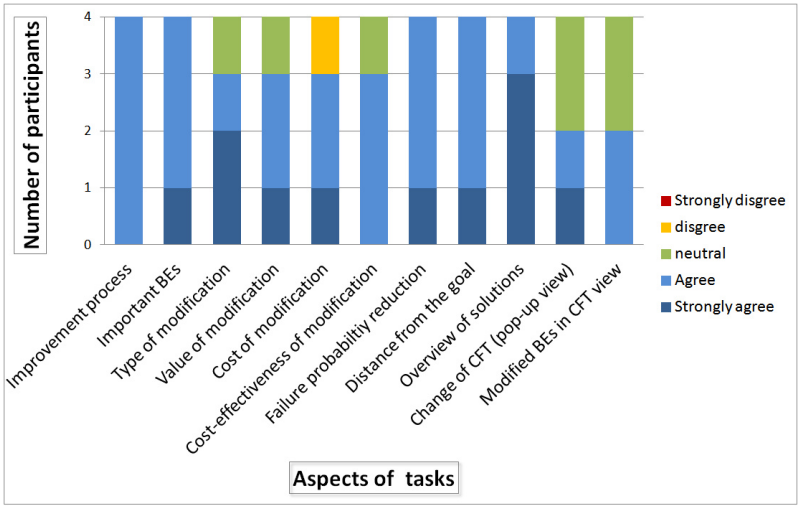


Figure 7 Qualitative evaluation.

more cost-effective. Hence, we abandon "M3" and terminate the corresponding solution "S2" (the last modification step of the solution "M3" is filled with black). The failure probability resulting from the updated component fault tree is not acceptable yet. Thus, we still need to perform the next iteration of risk reduction.

- Iteration 3:
  - Step 1: Identifying of the important basic event(s). We identify two important basic events having similar values.
  - Step 2: Applying design modifications. We apply redundancy-related modifications ("M4" and "M5") for the both basic events. A new solution "S3" appears for the branch generated by "M5".
  - Step 3: Evaluating the modifications. We decide to approve both modifications because the bars of the cost-effectiveness have similar length. The colors of both of the newly created risk-state nodes are now green. This indicates that the risk of the component fault tree is reduced to an acceptable level by applying either "S1" (ending in "M4") or "S3" (ending in "M5"). Because all the possible solutions are identified, we stop the construction process at this iteration.

4.2 Review Process

In this section, we review the solutions in the overview plot (see Figure 5 (3)) for identifying the optimal one. The fact that the total costs of solution "S3" is less than those of "S1" yields that "S3" is the more optimal way to improve the system safety.

5 Evaluation

We have performed an informal evaluation for our visualization approach. We invited four experts of the safety domain from the University of Kaiserslautern, all having profound proficiencies in the field of (component) fault tree analysis. We first introduced our approach to the participants, and then they were allowed to personally experience the visualization functionalities. Tasks with respect to the safety improvement process were provided for

the experience. Finally, the participants filled a Likert scale questionnaire for a qualitative evaluation.

The results (see Figure 7) showed that the feedback was mostly positive. The risk-reduction plot was preferred because this visually provided a sequence of modifications, while intuitively presenting the important data of each modification in the same view. When comparing modifications or analyzing patterns, using the plot was more intuitive than investigating data in separate views. The bars for the importance of BEs also had good reviews because they were easy to understand and dynamically linked to the visualization of the modifications.

The risk-state node visualizing the modification data had got a little different opinions. Most complaints concentrated on the small size of the node. The graphic properties attached to the node was too small to be effectively used, particularly the comparison of the cost-effectiveness bars. A suggestion was to apply an interactive fish-eye zoom for the interesting node. A participant commented that a small risk-state node with all graphic properties looked crowded. For example, although the attributes of modifications (i.e., the modification cost, modification type, and modification value) provided significant information for analysis of the existing modifications, the graphical representations of the data did not play an important role when identifying a modification. He suggested to dynamically represent the data: show specific graphic properties only when requested.

The representations for the effects of modifications had good comments. Participants could clearly understand how much the risk reduced by a modification is and how much the actual risk still needed to be reduced. Considering the different points of the analysis view, participants also positively commented the overview plots of the solutions. For the adaptation of the CFT structure, participants commonly thought that the views for showing CFT structure was relative small, whether for the pop-up view or for the main CFT view. The suggestions included a size adjustable pop-up view, and space-efficient alignment between the main CFT view and the risk-reduction plot.

In sum, the invited domain experts preferred our approach because they believed that the proposed visualization methods and interactions could effectively facilitate the identification and analysis of the improvement solutions.

6 Conclusion

A safety-critical system may be improved by a set of design modifications developed by using a component fault tree-based safety improvement process. In case, where multiple design solutions exist the proposed method allows to identify appropriate solutions by taking the actual costs into account. Traditional representation methods separate the information generated in the safety improvement process across individual views which hampers the identification of solutions. We propose a visualization system that integrates all information that is relevant in a risk-reduction plot associated together with a component fault tree view. This allows to quickly identify and review individual design modification steps in the context of different solutions, while considering the optimization of solutions with respect to the cost of modifications. An assumption of our approach is that design modifications do not introduce new critical failures. Otherwise, we would need to apply additional modification steps for the newly introduced important basic events. We also assume that engineers address a vulnerability by only one design modification in a solution. In general, our visualization system supports engineers to identify a series of design modifications leading to an significant improvement of the overall system safety.

7 Acknowledgment

This work was supported by the International Research Training Group (IRTG 1131) of the DFG and the German Ministry of Education and Research (BMBF) in the context of the ViERforES (Virtuelle und Erweiterte Realität für höchste Sicherheit und Zuverlässigkeit von Eingebetteten Systemen) project. We thank our colleagues from the Robotics Research Lab of the University of Kaiserslautern for providing us the data as well as our colleagues of the University of Kaiserslautern and the University of Utah for their support.

References

1 Aldservice. RAMCommander. `http://www.aldservice.com`, accessed 15-May-2012.

2 M. Ankerst, C. Elsen, M. Ester, and H.P. Kriegel. Visual classification: An interactive approach to decision tree construction. *In Proc. 5th Int. Conf. on Knowledge Discovery and Data Mining (KDD '99)*, pages 392-396, 1999.

3 M. Ankerst, M. Ester, and H.P. Kriegel. Towards an effective cooperation of the user and the computer for classification. *In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '00)*, pages 179-188, 2000.

4 A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. Dependable Secur. Comput.*, 1(1):11-33, 2004.

5 T. Barlow and P. Neville. A Comparison of 2-D Visualizations of Hierarchies. *In Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS '01)*, pages 131-138. IEEE Computer Society, 2001.

6 T. Barlow and P. Neville. Case Study: Visualization for Decision Tree Analysis in Data Mining. *In Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS '01)*, pages 149-152. IEEE Computer Society, 2001.

7 S. Contini, L. Fabbri, and V. Matuzas. Concurrent Importance and Sensitivity Analysis Applied to Multiple Fault Trees. *JRC IPSC report, EUR 23825 EN, Ispra.*, 2009.

8 S. Contini, L. Fabbri, and V. Matuzas. A novel method to apply Importance and Sensitivity Analysis to multiple Fault-trees. *Journal of Loss Prevention in the Process Industries*, 3, 2010.

9 S. Contini, S. Scheer, and M. Wilikens. Sensitivity Analysis for System Design Improvement. *In Proceedings of the 2000 International Conference on Dependable Systems and Networks (DSN '00)*, pages 243-248. IEEE Computer Society, 2000.

10 R.B. Cross and J.E. Ballesio. An Integrated Quantitative Risk Assessment of an Oil Carrier. *In Safety and reliability: proceedings of ESREL 2003, European Safety and Reliability Conference 2003, Maastricht, The Netherlands*, 2003.

11 J. Fussell. How to hand calculate system reliability characteristics. R-24:169-174, 1975.

12 A.F. Hixenbaugh and The Boeing Company. Fault Tree for Safety. *D6-53604*, 1968.

13 Functional safety of electrical/electronic/programmable electronic safety-related systems. *International Standard IEC 61508*, 2000.

14 R.L. Iman. A matrix-based approach to uncertainty and sensitivity analysis for fault trees. *Risk Analysis*, 7(1), 1987.

15 ISO. Quality management and quality assurance – Vocabulary. *DIN EN ISO 8402*, 1994.

16 IsographSoftware. FaultTree+. `http://www.isograph-software.com`, accessed 15-May-2012.

17 B. Kaiser, P. Liggesmeyer, and O. Maeckel. A new component concept for fault trees. *In Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software (SCS '03), Adelaide*, pages 37-46, 2003.

18 J. Knight. Safety critical systems: challenges and directions. *In Proceedings of the 24th International Conference on Software Engineering (ICSE '02)*, pages 547-550. ACM, 2002.

19 H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1st edition, 1997.

20 John C. Lee and Norman J. McCormick. *Risk and Safety Analysis of Nuclear Systems*. Wiley, 2011.

21 Y. Liu and G. Salvendy. Interactive visual decision tree classification. *In Proceedings of the 12th international conference on Human-computer interaction: interaction platforms and techniques (HCI '07)*, pages 92-105. Springer-Verlag, 2007.

22 J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5:110-141, 1986.

23 K. B. Misra. *Handbook of Performability Engineering*. Springer-Verlag, 2008.

24 Arbeitsgruppe Software Engineering: Dependability of University of Kaiserslautern. Lecture of Safety and Reliability of Embedded Systems, 2011.

25 Y. Ou and J.B. Dugan. Sensitivity Analysis of Modular Dynamic Fault Trees. *In Proceedings of the 4th International Computer Performance and Dependability Symposium*, page 35. IEEE Computer Society, 2000.

26 N-K Pham, T-N Do, F. Poulet, and A. Morin. Interactive Exploration of Decision Tree Results. *Applied Stochastic Model and Data Analysis International Conference (ASMDA '07)*, pages 152-160, 2007.

27 J. Rausand and A. Hoylany. System reliability theory: models, statistical methods, and applications, pages 183-206, Wiley Inter-Science, 2 edition, 2003.

28 RelexSoftware. RelexArchitect. `http://www.relexsoftware.co.uk`, accessed 15-May-2012.

29 ReliaSoft. BlockSim. `http://www.reliasoft.com/BlockSim`, accessed 15-May-2012.

30 Robotics Research Lab. The Robotics Research Lab of the University of Kaiserslautern. `http://agrosy.informatik.uni-kl.de`, accessed 20-may-2012.

31 Smith, D. J., and Simpson, K. G. L., and ScienceDirect (Online service). *Safety critical systems handbook: A Straightforward Guide to Functional Safety, IEC 61508 (2010 Edition) and Related Standards*. Butterworth-Heinemann, 2010.

32 SyncopationSoftware. DPL-faulttrees. `http://www.syncopationsoftware.com/faulttree.html`, accessed 15-May-2012.

33 S.T. Teoh and K-L Ma. PaintingClass: Interactive Construction, Visualization and Exploration of Decision Trees. *In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03)*, 2003.

34 Stef van den Elzen and J.J. van Wijk. BaobabView: Interactive construction and analysis of decision trees. *IEEE VAST*, pages 151-160, 2011.

35 W.E. Vesely, J. Dugan, J. Fragola, J. MinarickIII, J. Railsback, and M. Stamatelatos. Fault Tree Handbook with Aerospace Applications. *NASA*, 2002.

36 W.E. Vesely, F.F. Goldberg, N.H. Roberts, and D.F. Haasl. Fault Tree Handbook. *U.S.Nuclear Regulatory Commission*, 1981.

CFD Simulation of Liquid-Liquid Extraction Columns and Visualization of Eulerian Datasets

Mark W. Hlawitschka[1,4], Fang Chen[2,4], Hans-Jörg Bart[1,4], and Bernd Hamann[3]

1 **Chair of Separation Science and Technology, University of Kaiserslautern, Germany**
mark.hlawitschka@mv.uni-kl.de, bart@mv.uni-kl.de
2 **Computer Graphics and HCI Group, University of Kaiserslautern, Germany**
chen@informatik.uni-kl.de
3 **Institute for Data Analysis and Visualization & CS Department University of California, Davis, USA** hamann@ucdavis.edu
4 **Center for Mathematical and Computational Modelling (CM)^2, University of Kaiserslautern, Germany**

Abstract

In this joint work, a complete framework for modeling, simulating and visualizing multiphase fluid flow within an extraction column is presented. We first present a volume-of-fluid simulation, which is able to predict the surface of the droplets during coalescence. However, a fast and efficient model is needed for the simulation of a liquid-liquid extraction column due to the high number of occurring droplets. To simulate the velocity and droplet size in a DN32 extraction column, a coupled computational fluid dynamic-population balance model solver is used. The simulation is analyzed using path-line based visualization techniques. A novel semi-automatic re-seeding technique for droplet path-line integration is proposed. With our technique, path-lines of fluid droplets can be re-initialized after contact with the stirring devices. The droplet breakage is captured, allowing the engineer to improve the design of liquid-liquid columns layout.

1998 ACM Subject Classification I.6.6 Simulation Output Analysis

Keywords and phrases computational fluid dynamics, multiphase fluid, droplet collision, Eulerian, path-line

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.59

1 Introduction

In chemical industry, liquid-liquid extraction is an important separation technique based on the relative solubilities of the compounds in two different immiscible liquids. It is mainly applied when distillation is impractical (similar boiling points of the materials) or the mixtures contain temperature sensitive compounds. The efficiency of the column is mainly influenced by the choice of the solvent and by the hydrodynamics inside the column. The characterization of the hydrodynamics without experiments was made possible through the use of computational fluid dynamics (CFD). Multiphase flow is commonly simulated by the use of the volume-of-fluid (VOF) model, Euler-Euler model or Euler-Lagrange model. VOF allows a tracking of the dispersed phase surface but requires a detailed resolution of the droplet and its curvature by the computational mesh. Due to this, it is mainly used for the investigation of single droplet interactions. The Euler-Lagrange method is limited by the

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 59–70

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


phase fraction and requires a higher computational load compared to the Euler-Euler model. Therefore, the Euler-Euler model still is the workhorse for dispersed multiphase simulations.

The simulation of extraction columns with CFD started with Rieger et al. [3], who used the CFD Code Fire for single-phase simulations of an extraction column of type rotating disc contactor (RDC). Modes & Bart [4] used the code FIDAP to perform two-phase simulations. The droplet size change in an RDC column was accounted by Vikhansky & Kraft [5] and Drumm & Bart [6] using population balance models. Whereas Drumm & Bart [6] mainly focuses on the moment based population balances, Vikhansky & Kraft [5] used the Monte Carlo method. Hlawitschka et al. [7] transferred the results of Drumm [8] to a three-dimensional test case of a Kühni extraction column. For population balance modeling, a one-group model (OPOSPM) is used which guarantees a low computational time and a good prediction of the hydrodynamics and flow field [9]. An adaption of the coalescence and breakage kernels using models from Martínez-Bazán et al. [10], Prince & Blanch [11] and Luo & Svendsen [12] was investigated by Hlawitschka & Bart [13].

In this paper, the volume-of-fluid method is used to illustrate a binary droplet coalescence and to show up the challenges for a whole column simulation. In a next step, a coupled CFD-population balance code is used to simulate a section of a Kühni Miniplant column using the Eulerian model coupled with a population balance model. In the output of our presented simulation, the interface is not directly tracked and the dispersed phase is only represented by the phase fraction and droplet size within each computational cell. Therefore, a direct analysis of the simulation output (e.g. droplet position) is not possible with the current visualization techniques. The Eulerian model requires an adequate representation scheme of the dispersed phase, which couples both the cell volume fraction as well as the particle size. Recent research on stochastic modeling [14] gives us a pointer on how those two fields can be negotiated and brought together. Apart from droplet distribution in each time step, the dynamic behavior of sets of droplets is of great interest among engineers and researchers. Recent research in fluid visualization has shown that line integrals have been helpful visual tools in tracking fluid particles [15, 16, 17]. Aside from accurate computation of line integrals, the start and end points of a droplet path should be handled with special care in the extraction column, since fluid droplet break or merge along their path. A novel re-seeding approach for capturing droplet path-lines for time-varying multiphase flow field is proposed.

2 Binary Droplet Coalescence

2.1 Numerical Background

To illustrate the basic droplet coalescence, we simulate a binary droplet coalescence using the open source CFD tool OpenFOAM. In our implementation, we used a volume-of-fluid type of solver, called multiphaseInterFoam. This method is combined with dynamic mesh refinement for a better resolution of the droplet interface. Due to the VOF model theory, a direct contact of the droplets leads to a direct coalescence of the droplets. The VOF model was applied by several groups to simulate bubble coalescence [18, 19, 20] and also applied for droplet coalescence [21]. A droplet in the VOF method is represented by the volume fraction α, which is given by:

- surrounding phase for $\alpha = 0$
- droplet for $\alpha = 1$
- existence of an interphase when $0<\alpha<1$

The transport of mass is described by the continuity equation

$$\frac{\partial}{\partial t}\alpha + \nabla(\alpha \mathbf{u}) = 0, \tag{1}$$

where the velocity is given by $\mathbf{u}$. In addition, a single momentum equation is used for the mixture of two-phase-fluid. The viscosity and the density of the mixture is expressed by

$$\begin{aligned} \mu &= \alpha_2\mu_2 + (1-\alpha_2)\,\mu_1 \\ \rho &= \alpha_2\rho_2 + (1-\alpha_2)\,\rho_1 \end{aligned} \tag{2}$$

The momentum equation hence is described by

$$\frac{\partial}{\partial t}(\rho\mathbf{u}) + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) + \nabla\mathbf{u}\cdot\nabla[\mu] = -\nabla p + \mathbf{F}, \tag{3}$$

where $\mathbf{F}$ is the surface tension force $\mathbf{F} = \sigma\kappa(x)\mathbf{n}$. κ is the curvature of the interface and $\mathbf{n}$ is a unit vector normal to the interface.

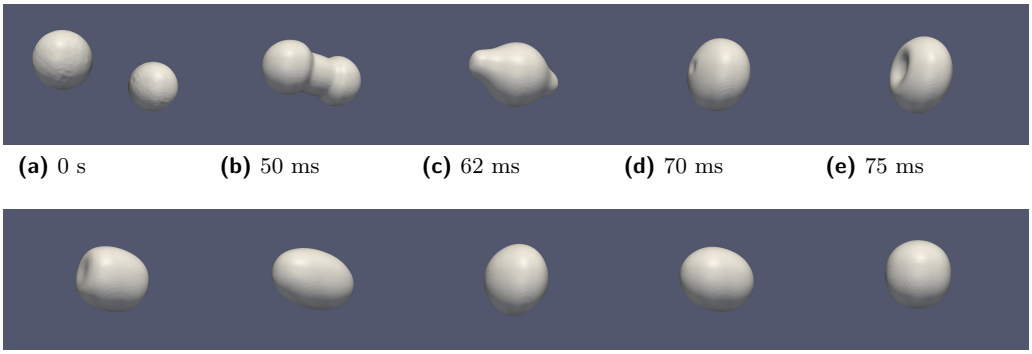
2.2 Numerical Setup

The grid is generated using the open source grid generator blockMesh creating a total simulation area of size x=0.01 m, y= 0.006 m and z=0.006 m. The initial mesh is generated with 200 cells in x-axis, 120 cells in y-axis and 120 cells in z-axis. The boundaries of the mesh are defined by a zero gradient boundary condition. The two toluene droplets (ρ=866 kg/m^3, ν=6.3$\cdot$ 10^{-7}m^2/s) are defined by spherical set functions defining a phase fraction of 1 in a radius of 0.002 m around the locations (-0.004,0,0) and (0.004,0,0) that represent the center of the droplets. The surrounding liquid is defined as water. The surface tension between the droplets and the water is set to 35 mN. A dynamic mesh refinement is used to refine the interface of the droplets in advance of the simulation and during the simulation. The upper refine level based on the volume fraction is set to 0.999, while the lower is set to 0.1. Forty buffer layers were used to generate a smooth mesh. In each refinement step, the maximum refinement is set to 20 resulting in an initial mesh of 3 million cells. The time step is set to 10^{-5} s and the simulation is run to simulate 0.25 s until the droplets coalesce and a new round droplet is formed. Each of the two droplets has an initial velocity of 0.1 m/s in the direction towards the other droplet. Gravity is neglected in order to ensure droplet motion only in the x-axis direction.

2.3 Binary Droplet Coalescence Result

The simulated droplet coalescence is shown in Fig. 1. The droplets come in touch with each other after 42 ms and a liquid bridge is formed. The formed droplet changes its shape from a bar-bell at 50 ms (b), to a citrus form at 62 ms (c), over a sphere form at 70 ms (d) to a dented form at 75 ms (e). The droplet is elongated again in x-axis to a cylindrical form at (100 ms) (f) and changes to an oval droplet at 0.12 s (g) and back to the round sphere form at 160 ms (g). The droplet is stretched again at 180 ms (h) forms a round droplet at 200 ms (i). The periodic elongation in x-axis decreases with time until the newly generated droplet is stable.

The droplet coalescence can be described in good agreement with literature data [22]. However the first contact of the droplet is still a challenging task. Repulsion effects and delayed coalescence, which describes a coalescence that does not directly start after the initial contact, is still simulated using simplified models based on a contact time and contact



(a) 0 s **(b)** 50 ms **(c)** 62 ms **(d)** 70 ms **(e)** 75 ms

(f) 100 ms **(g)** 120 ms **(h)** 160 ms **(i)** 180 ms **(j)** 200 ms

Figure 1 Droplet coalescence of two 4 mm droplets simulated using the VOF method.

area [21]. In a pilot plant column with diameters of 150 mm, several 100,000s of droplets exist. To model a full column, the number of mesh cells will increase dramatically and exceed the computational resources. Moreover, droplet breakage is becoming problematic to model, because they heavily rely on the mesh size and quality. Hence, a simulation without describing the droplet interface directly is preferred.

While VOF focuses on single droplet boundaries, Eulerian models provide the possibility to simulate complex scenarios with large number of dispersed droplets. In the next section, we present a Eulerian model combined with population balance modeling to simulate the hydrodynamics in a section of a Kühni Miniplant column.

3 Eulerian Modeling

The Eulerian model is part of the Euler-Euler approach, where the phases are treated as interpenetrating continua. The occurring phases are represented by the volume fraction α, which is identical to a statistical probability of the droplets being at a specific position. The sum of the phase fractions of the continuous (c) and dispersed phase (d) has to be 1:

$$\alpha_c + \alpha_d = 1. \tag{4}$$

The transport of the phases is described by the conservation equations and is solved for each phase. The continuity equation consists of the storage term and the convective term on the left hand side of the equation and of a source term on the right hand side of the equation.

$$\frac{\partial(\alpha_c \rho_c)}{\partial t} + \nabla \cdot (\alpha_c \rho_c \mathbf{u}_c) = \rho_c S_c. \tag{5}$$

The momentum balance of the continuous phase is given by:

$$\frac{\partial(\alpha_c \rho_c \mathbf{u}_c)}{\partial t} + \nabla \cdot (\alpha_c \rho_c \mathbf{u}_c \mathbf{u}_c) = -\alpha_c \nabla p + \nabla \tau_l + \alpha_c \rho_c \mathbf{g} + \mathbf{F}_c + \rho_c \mathbf{u}_c S_c. \tag{6}$$

The first term on the left hand side is the rate of change of momentum and the second term is the conservation of momentum. On the right hand side the effect of pressure and stress-strain is given by the first two terms. The interaction of phases is taken into account by the resistance forces $\mathbf{F}_c$. The source term S is set to 0. The momentum balance and continuity equation is formed for the dispersed phase respectively. Analogous to the work of

[23] only the drag force is taken into account as interphase resistance. The mass force and the buoyancy force for liquid-liquid systems can be neglected. The resistance between the continuous phase and the dispersed phase is calculated with the help of the model of Schiller & Naumann [24] for the drag coefficient C_D

$$\mathbf{F}_{c,d} = \frac{3\rho_c\alpha_c\alpha_d C_d|\mathbf{u}_d - \mathbf{u}_c|(\mathbf{u}_d - \mathbf{u}_c)}{4d_d} \tag{7}$$

where

$$C_D = \begin{cases} 24(l + 0.15Re^{0.678})/Re, & Re \leq 1000 \\ 0.44 & Re > 1000. \end{cases} \tag{8}$$

3.1 Droplet Size Calculation

Since the dispersed phase surface is not directly tracked, events such as droplet coalescence and droplet breakage as well as droplet growth cannot be accounted for directly. In the last decade, population balance models in combination with breakup and coalescence kernels became popular to account for these effects. The variable droplet size can be accounted for different models, as represented by the class method or the Quadrature Method of Moments (QMOM). In this work, however, a one group model, the One Primary One Secondary Particle Method, is used [9]. The model calculates the mean droplet size in each cell based on the volumetric diameter and is calculated by the third moment and the zeroth moment:

$$d_{30} = \sqrt[3]{\frac{m_3}{m_0}} = \sqrt[3]{\frac{6\alpha}{\pi m_0}}. \tag{9}$$

The third moment refers to the total volume of the droplets, where the zeroth moment gives the total number of droplets. The number of droplets increases due to breakage and decreases due to coalescence. In addition to the Eulerian modeling, only one additional transport equation has to be added to the solver for the number concentration of the dispersed phase d:

$$\frac{\partial(\alpha_d m_0 \rho_d)}{\partial t} + \nabla \cdot (\alpha_d \rho_d m_0 \mathbf{u_d}) = \alpha_\mathbf{d}\rho_\mathbf{d}\mathbf{S_d}. \tag{10}$$

The source term on the right hand side describes the generation and reduction of the number of droplets. For this description, the coalescence model of Coulaloglou & Tavlarides [2] and the breakage model of Andersson & Andersson [1] are used in this work. The coalescence of two droplets is described by the coalescence efficiency τ and by the coalescence rate h:

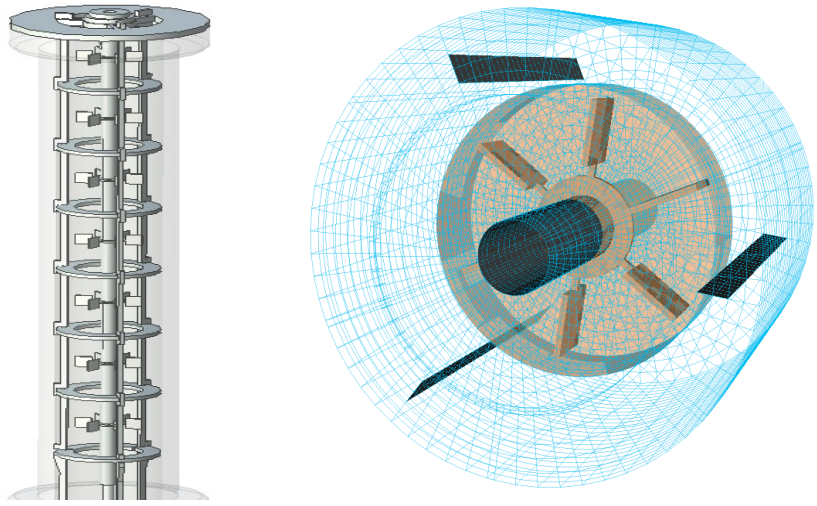
$$\begin{aligned} \tau(d_1, d_2) &= \exp(-C_1 \frac{\mu_c \rho_c \epsilon}{\sigma^2}(\frac{d_1 d_2}{d_1 + d_2})^4) \\ h(d_1, d_2) &= C_2 \frac{\epsilon^{1/3}}{1+\alpha}(d_1 + d_2)^2(d_1^{2/3} + d_2^{2/3})^{1/2} \end{aligned} \tag{11}$$

The constants C_1 and C_2 are adjusted to the system of butyl acetate/water. The breakage kernel consists of the breakup rate g, which is given by:

$$g(d) = \int_{d_{0/10}}^{10d_0} \omega_s(d_0, \lambda) P(d_0, \lambda)\, d\lambda \tag{12}$$

The interaction frequency ω_s between a droplet of size d_0 and an eddy of size λ is:

$$\omega_s(d_0, \lambda) = \frac{C_3 \pi d_0^3 \epsilon^{1/3}}{6\lambda^{14/3}}. \tag{13}$$



(a) (b)

Figure 2 Column geometry (left) and a visualization of the generated mesh of a single compartment containing the moving reference frame around the stirrer (right).

The constant C_3 is taken from the model of Luo & Svendsen [12] and is 0.822. The probability of an eddy breaking up a fluid particle of size d is given by the integral of the normalized energy distribution $\phi(\chi)$:

$$P(d_0, \lambda) = \int_{\chi_{\min}}^{\infty} \phi(\chi)\, d\chi. \tag{14}$$

χ is defined by the ratio of the eddy viscosity to the average eddy viscosity. In addition to the breakup rate, the available energy by turbulence must be larger than the increase in interfacial energy due to deformation that breakup occurs. In comparison to the direct simulation of a droplet coalescence and breakage, the breakup kernel and coalescence kernel has to account for the fluctuations (e.g. elongation) of the droplets taking place in the real column. Furthermore, surfactants can have an influence to the coalescence and breakage and have to be accounted by the adjustable parameters of the models.

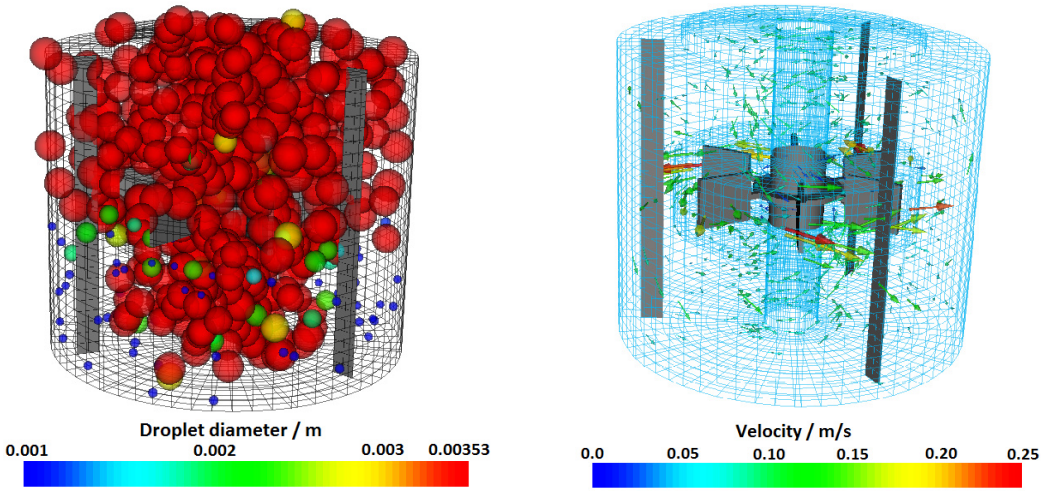
3.2 Mesh and Boundary Conditions

In this section, we describe the geometry of our extraction column, the numerical mesh for simulation and the boundary conditions for solving PBM. For the simulation of a seven-compartment section of a Kühni Miniplant column, a corresponding mesh is built in Gambit resulting in over 500 000 cells. The simulated column geometry is shown in Fig. 2a. An in- an outflow section are added to the numerical mesh at the bottom and top of the column. A closer look inside a single compartment is given in Fig. 2b, where the part around the stirrers is treated as moving reference frame (MRF). In the middle of each compartment, a six-baffled stirrer is installed. Neighboring compartments are separated by two stators, which in this case consist of two metal rings. Three stream-breakers are orientated in an angle of 120° to each other. The volume stream of each phase (water and butyl acetate) is set to 8 l/h and the stirring speed is set to 300 rpm. The constants in the model of Coulaloglou & Tavlarides

[2] are set adequate to the results of Drumm & Bart [6] to 0.005 and $1.0 \cdot 10^{11}$ to fulfill the coalescence and breakage characteristics of the used system. The simulation is performed for 20 000 time steps using a time step size of 0.05 s, where the standard relaxation factors of the commercial CFD code FLUENT are used. A converged and steady state solution is reached at the end of the simulation.

4 Visualization and Results of the Eulerian Simulation

A novel approach in visualizing the Eulerian multiphase fluid simulation dataset is presented with the aim of demonstrating the dynamics of droplet movement. The simulated droplet distribution is visualized in Fig. 3a [25] based on a stochastic modeling combining the information of phase fraction and droplet size. The simulated droplet size is homogeneous for the whole compartment. Slightly smaller droplets can be seen due to numerical diffusion. Compared to literature data [26], the droplet size is slightly over-predicted by the used breakage and coalescence models. A further reduction of the coalescence kernel could lead to a better prediction of the measured droplet size data. The velocity field of the continuous phase in Fig. 3b indicates the swirl structure inside the compartment, where the velocity has its highest values at the stirrer tip. For a better understanding of the flow structure, a visualization of the droplet path through the compartment is needed.



(a) Droplet distribution **(b)** Velocity field

Figure 3 Visualization of the simulation result gained by the Eulerian CFD-PBM simulation.

4.1 Path-Line Based Flow Visualization

Line integrals, such as streamlines and streak-lines are well-known representations for flow visualization [16, 17]. In the case of our particular simulation, domain experts benefit from path-lines that indicates the trace of a droplet movement over time.
A path-line denotes the trace/path of a droplet and is defined mathematically as:

$$\begin{cases} \frac{\partial}{\partial t} l(x, y, z) = \mathbf{u}(l, t), \\ l(t_0) = l_0. \end{cases} \tag{15}$$

u is the droplet velocity and l is the position of the particle. However, conventional path-line computation suffers two main deficits:

- Path-line hits mesh boundaries around stirrers and no longer continues.
- Path-line computation continues even if a droplet vanishes after a certain time point.

For simulation experts, it is important to identify flow regions with the following features:

1. regions where a droplet breaks
2. regions where droplets collide and merge.

In order to visualize these features based on a path-line representation, we propose the following coalescence and breakage detection and re-seeding method to incorporate more information into path-lines:

- Re-seeding near stirrers: if a path-line is interrupted at a mesh point near the stirrer, take another point which is near the stirrer, and start a new path-line integral from this point.
- Path-line termination after droplet interaction: Physically a trace of a droplet vanishes while the droplet itself disappears. Therefore, further integration should also be stopped after a larger droplet break. We incorporate this breakage detection into path-line integration, in order to record the life span of the droplet along its path-line.

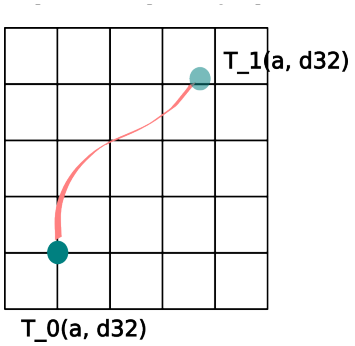


Figure 4 Path-line intergration.

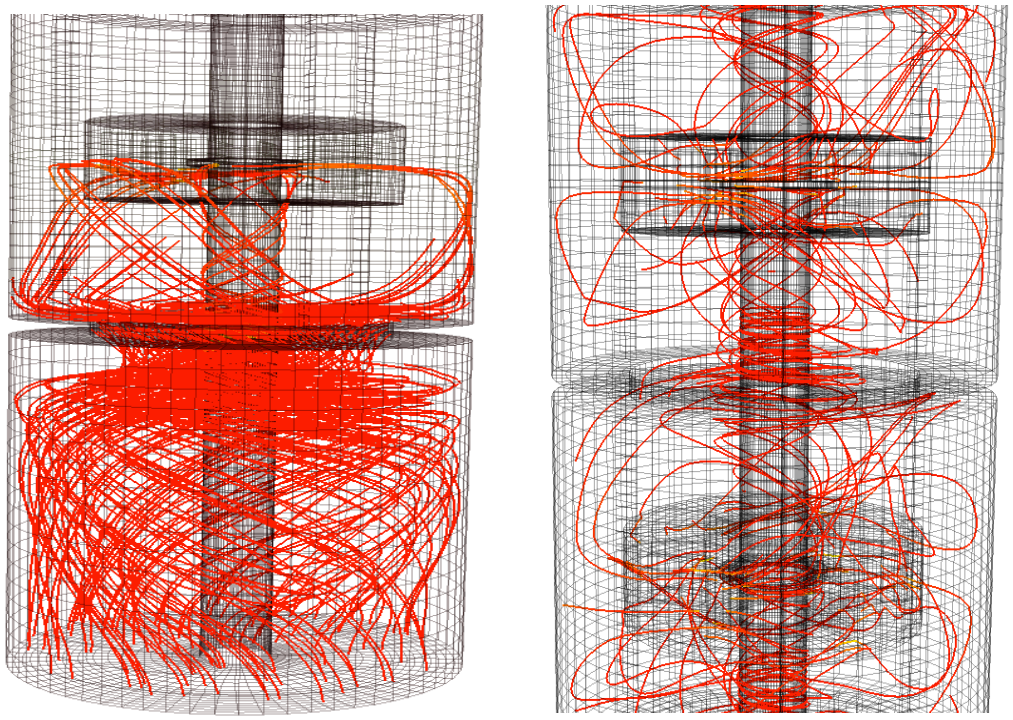
We will elaborate on the path-line termination process in detail (see Algorithm 1). Suppose a droplet has position P_0 at point t_0, after time t_1, the droplet moves over to P_1, see Fig. 4. Volume fraction and particle size at each time step are given as α_i , d_i. With the help of the following detection, we test if t_1 is a time frame where a larger droplet breaks:

Algorithm 1 Adaptive path-line integration with breakup detection.

```
for each path-line do
  for t = 0 → t_i do
    if d_i/d_{i+1} > ∛2 and α_i > α_threshold then
      terminate current path-line integration,
      take a neighboring point, start a new path-line
    end if
  end for
end for
```

A droplet breakage is detected by the condition $d_i/d_{i+1} > \sqrt[3]{2}$. We assume that each droplet is a sphere with diameter d_i. Thus the volume of a droplet is given by $\frac{4}{3}\pi(\frac{d_i}{2})^3$. We further

assume a droplet will break in to two or more equal parts. Due to volume conservation, the diameter of the resulting droplet children r d_{i+1} is smaller than $\sqrt[3]{(\frac{1}{2})}d_i$. The threshold checking on volume fraction $\alpha_i > \alpha_{threshold}$ ensures that a droplet exists in this region. If a droplet breaks up, we terminate the path-line calculation at this break point. We re-seed a new point randomly within a given neighborhood. In the future, we plan to extend this re-seeding idea with further droplet breakage model, in order to provide a better location of the children droplets. Furthermore, the re-seeding of a boundary point is done by flipping the velocity vector. We assume if a particle hits the boundary, it bounces back with the same velocity magnitude and a reflecting direction.



(a) Straightforward path-line integration **(b)** Path-lines with collision and bifurcation detection

Figure 5 Comparison of straight forward integration method and the particle reseeding method.

The example in Fig. 5a shows a straightforward integration without feature detection and line re-seeding. Apparently all the impaired particle paths were interrupted once they reach the stirrers. Furthermore, following all path-lines reveals the fact that no droplet breakage can be recorded. We implement our proposed re-seeding and termination criteria with a Runge-Kutta4 integration method in C++. Integration length was set smaller than the average cell size in order to obtain a smoother line output. Fig. 5b shows an example of our algorithm whose path-lines are re-seeded after droplet breakage, and integration can be continued after an interruption at the stirrer. From the presented path-line technique, the path of the droplets through the column can be visualized. Thereby, the typical double vortex structure of the flow can be seen [27].

5 Conclusions

In this paper, we presented different modeling techniques for the simulation of the hydrodynamics in extraction columns. We have shown that a full resolution of the droplet coalescence is only possible for two droplets or small number of droplets due to the required computational resources. The simulation of dispersed phase systems with thousands of droplets requires the use of simplified models. Therefore, the Eulerian model is used combined with population balance modeling to simulate the flow field as well as the changing droplet size in extraction columns. The used breakage and coalescence model for population balance modeling showed a slight over-prediction of the droplet size. Compared to the VOF method, the droplet surface is not directly tracked. Therefore, the droplets were visualized using a stochastic positioning based on the phase fraction and the droplet size. Furthermore, detection and re-seeding algorithms for droplet interactions and tracing were presented that enable the engineer to detect points of breakage and are the basis for further improvements. As future work, we will use a multi-group model that allows us the consideration of more than one droplet size in a cell. With that model, the effect of small satellite droplets that are generated during breakage can be studied. Therefore, the positioning of the droplets based on the phase fraction and droplet size will be extended to account for a stochastic visualization of the multi-group data set, which allows a direct comparison of real pictures with the visualization. In addition, a visualization of elongated droplets will help to identify these points based on the shear rate of the fluids. A visualization of the droplet movement (transient visualization) will lead to a better understanding of the start-up phase of the column. The visualization of the multi-group data will show up the position of the smaller sized droplets compared to larger sized droplets inside the compartments at one time (without using slides or probes) to reveal dead zones. Finally, the visualization of mass transfer specific data together with the droplet size data will allow combined analyses of both the interacting properties. Further investigations of different column designs (e.g. height of a compartment) will be done. Therefore, a script based mesh generation tool will be developed. The presented algorithms were only used for liquid-liquid flows, but could also be applied for the simulation and visualization of bubbly flows as it is given in bubble column reactors.

Acknowledgements The authors would like to acknowledge the financial support from DFG (Deutsche Forschungsgemeinschaft) and the State Research Centre of Mathematical and Computational Modelling (CM)^2.

A Nomenclature

c	continuous phase
d	dispersed phase
$C_1 - C_3$	model constants
C_D	drag coefficient
d_{30}	volumetric diameter
d_1, d_2	droplet diameter
f	probability density distribution function
F	surface tension force
F_r	resistance force
$g(d)$	breakup rate
g	gravity constant
h	coalescence rate
l	location of a point
m_0	zeroth moment
m_3	third moment
n	unit vector normal to the interface
N	number of droplets
p	pressure
P	probability of an eddy breaking up
P_0, P_1	position 0, position 1
S	source term
t	time
$\mathbf{u}$	velocity
V	volume
x, y, z	coordinates
α	phase fraction
ϵ	energy dissipation
κ	curvature of the interphase
λ	length of an eddy
μ	viscosity
δ	surface tension
ϕ	energy distribution function
χ	eddy viscosity/average eddy viscosity
ω	interaction frequency
τ	phase strain stress tensor

References

1 R. Andersson and B. Andersson. Modeling the breakup of fluid particles in turbulent flows. *A.I.Ch.E Journal*, 6:2031–2038, 2006.

2 C.A. Coulaloglou and L. L. Tavlarides. Description of interaction processes in agitated liquid-liquid dispersions. *Chemical Engineering Science*, 32:1289–1297, 1977.

3 R. Rieger, C. Weiss, G. Wigley, H.-J. Bart and R. Marr. Investigating the process of liquid-liquid extraction by means of computational fluid. *Computers & Chemical Engineering*, 12:1467–1475, 1996.

4 G. Modes and H.-J. Bart. CFD-Simulation der Strömungsnichtidealitäten der dispersen Phase bei der Extraktion in gerührten Extraktionskolonnen. *Chemie Ingenieur Technik*, 73, (4):332–338 2001.

5 A. Vikhansky and M. Kraft. Modelling of a RDC using a combined CFD-population balance approach. *Chemical Engineering Science*, 59:2597–2606, 2004.

6 C. Drumm and H.-J. Bart. Coupling of CFD with DPBM, drop size distributions and flow fields in a RDC extractor. *6th International Conference on Multiphase Flow*, Leipzig, Germany, July 9-13, 2007.

7 M. W. Hlawitschka, M. Mickler and H.-J. Bart. Simulation einer gerührten Miniplant-Extraktionskolonne mit Hilfe eines gekoppelten CFD-Populationsbilanzmodells. *Chemie Ingenieur Techik*, 82 (9):1389–1390, 2010.

8 C. Drumm. Coupling of Computational Fluid Dynamics and Population Balance modeling for liquid-liquid extraction. *Ph. D Dissertation*, TU Kaiserslautern, 2010.

9 C. Drumm, M. Attarakih and M. W. Hlawitschka and H.-J. Bart. One-group reduced population balance model for CFD Simulation of a pilot-plant extraction column. *Industrial & Engineering Chemistry Research*, 49 (7):3442–3451, 2010.

10 C. Martínez-Bazán, J. L. Monatenes and J. C. Lasheras. On the breakup of an air bubble injected into a fully developed turbulent flow. Part 1. Breakup frequency. *Journal of Fluid Mechanics*, 401:157–182, 1999.

11 M. J. Prince and H. W. Blanch. Bubble coalescence and break-up in air-sparged bubble columns. *A.I.Ch.E Journal*, 36 (10):1485–1499, 1990.

12 H. Luo and H. F. Svendsen. Theoretical model for drop and bubble breakup in turbulent dispersions. *A.I.Ch.E Journal*, 42 (5):1225–1233, 1996.

13 M. W. Hlawitschka and H.-J. Bart. Simulation of a miniplant kühni extraction column coupled with PBM. Proceeding ISEC 2011, 2011.

14 Goodson, M. and Kraft, M., Simulation of coalescence and breakage: an assessment of two stochastic methods suitable for simulating liquid-liquid extraction. *Chemical Engineering Science*, 59 (18):3865–3881, 2004.

15 T. Salzbrunn, C. Garth, G. Scheuermann, J. Meyer. Pathline predicates and unsteady flow structures. *The Visual Computer*, 24 (12):1039–1051, 2008.

16 C. Garth, H. Krishnan, X. Tricoche, T. Bobach and H. Joy. Generation of accurate integral surfaces in time-dependent vector fields. *Proceeding of IEEE Visualization 2008*, 14 (6), 2008.

17 A. Lez, A. Zajic, K. Matkovic, A. Pobitzer, M. Mayer, H. Hauser. Interactive exploration and analysis of path-lines in flow data. *Proceeding International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*:17-24, 2011.

18 E. Delnoij,J. A. M. Kuipers and W. P. M. Van Swaaij. Numerical simulation of bubble coalescence using a Volume of Fluid (VOF) model. *Third International Conference on Multiphase Flow*, Lyon, France, June 8th-12th, 1998.

19 D. Bothe, M. Koebe, K. Wielage, J. Pruss and H.-J. Warnecke. Direct numerical simulation of mass transfer between rising gas bubbles and water. In *Bubbly Flows:* Analysis, Modelling and Calculation (M. Sommerfeld Ed.), Heat and Mass Transfer, Springer-Verlag:159–174, 2004.

20 M. Koebe. Numerische Simulation aufsteigender Blasen mit und ohne Stoffaustausch mittels der Volume of Fluid (VOF) Methode. *Ph. D Dissertation*, Universität Paderborn, 2004.

21 F. Krause, X. Li and U. Fritsching. Simulation of droplet-formation and -interaction in emulsification processes. *Engineering Applications of Computational Fluid Mechanics*, 5 (3):406-415, 2011.

22 R. T. Eiswirth and H.-J. Bart. Untersuchung zur binären Tropfen-Tropfen-Koaleszenz in flüssig-flüssig-Systemen. *Chemie Ingenieur Technik*, 81 (8):1060, 2009.

23 C. Drumm and H.-J. Bart. Hydrodynamics in a RDC extractor, single and two-phase PIV measurements and CFD simulations. *Chemical Engineering Technology*, 29:1297–1032, 2006.

24 L. Schiller and Z. Naumann. A drag coefficient correlation. *Zeitschrift des Vereines deutscher Ingenieure*, 77:318, 1935.

25 M. W. Hlawitschka, F. Chen, H.-J. Bart and H. Hagen. CFD Simulation und verbesserte Datenauswertung einer Extraktionskolonne vom Typ Kühni. *Proc. of 1st Young Researcher Symposium (YRS) 2011*, Kaiserslautern, Germany, February 15, 2011, CEUR-WS.org, ISSN 1613-0073, online CEUR-WS.org/Vol-750/yrs04.pdf

26 T. Steinmetz. Tropfenpopulationsbilanzgestütztes Verfahren zur Skalierung einer gerührten Miniplant-Extraktionskolonne. Fortschritts-Bericht, VDI Verlag, Düsseldorf, 2007.

27 M. W. Hlawitschka and H.-J. Bart. Determination of local velocity, energy dissipation and phase fraction with LIF- and PIV- measurement in a Kühni Miniplant extraction column. *Chemical Engineering Science*, in press., doi:10.1016/j.ces.2011.10.019, 2011.

Feature-based Visualization of Dense Integral Line Data

Simon Schröder[1,2], Harald Obermaier[4], Christoph Garth[3], and Kenneth I. Joy[4]

1 **Computer Graphics and HCI Group**
University of Kaiserslautern, Germany
2 **Fraunhofer ITWM, Kaiserslautern, Germany**
simon.schroeder@itwm.fraunhofer.de
3 **Computational Topology Group**
University of Kaiserslautern, Germany
garth@cs.uni-kl.de
4 **Institute for Data Analysis and Visualization,**
University of California Davis, USA
hobermaier@ucdavis.edu, joy@cs.ucdavis.edu

Abstract

Feature-based visualization of flow fields has proven as an effective tool for flow analysis. While most flow visualization techniques operate on vector field data, our visualization techniques make use of a different simulation output: Particle Tracers. Our approach solely relies on integral lines that can be easily obtained from most simulation software. The task is the visualization of dense integral line data. We combine existing methods for streamline visualization, i. e. illumination, transparency, and halos, and add ambient occlusion for lines. But, this only solves one part of the problem: because of the high density of lines, visualization has to fight with occlusion, high frequency noise, and overlaps. As a solution we propose non-automated choices of transfer functions on curve properties that help highlighting important flow features like vortices or turbulent areas. These curve properties resemble some of the original flow properties. With the new combination of existing line drawing methods and the addition of ambient occlusion we improve the visualization of lines by adding better shape and depth cues. The intelligent use of transfer functions on curve properties reduces visual clutter and helps focusing on important features while still retaining context, as demonstrated in the examples given in this work.

1998 ACM Subject Classification I.3.8 Computer Graphics Application

Keywords and phrases Flow simulation, feature-based visualization, dense lines, ambient occlusion

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.71

1 Introduction

Flow simulation has a long history in scientific computing. For several decades simulation was limited to 2D because of symmetry or for computational considerations. Hence, a lot of excellent methods for integral flow visualization have been developed, e. g. LIC (Line Integral Convolution). Now, with the availability of more computational power, simulations have expanded to three dimensions. This leaves the scientific visualization community with new challenges for the visualization of flows in 3D. Existing techniques from two dimensions cannot be easily adapted for 3D as most display devices are still 2D.

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 71–87

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Integral lines, i. e. streamlines and pathlines, are a common way to show the structure of a flow. Several techniques like illuminated lines, halos, and tube-like rendering have been developed to enhance rendering of these lines. In many cases there are too few or too many lines showing a lot of unnecessary information instead of focusing on important flow features.

For our visualization we solely use integral lines as output from simulations without the need of the underlying flow field. This makes it easy to use our visualization approach as a post-processing step with any common flow simulation software that can produce integral lines as output. There also are some tools available for visualizing integral lines. In contrast to these existing methods we require a dense sampling of integral lines. The idea behind this is that we are then able to provide high resolution visualization results in important regions and adaptively dim out unwanted information.

Visualization of dense lines has two major problems: overlapping of lines and occlusion of lines. These problems are even more severe with our requirement of dense sampling of integral lines. There are already solutions to these two problems: (1) halos around lines reduce overlap and provide visual separation, and (2) transparency reduces occlusion and reveals hidden lines.

In this paper we combine illuminated lines, transparency, and halos for an improved line visualization. To stress the spatial relationship of lines we introduce an ambient occlusion technique that is suitable for dense line data sets. Transparency is mainly used to highlight interesting flow features. The second contribution of our work is the definition of interactive transfer functions on curve properties that allow to extract otherwise hidden structures, e. g. vortices and turbulent regions. In addition, transfer functions on colors or standard color maps can be applied for illustrative purposes or deeper insight into the flow.

In the next section we start by discussing existing related work. Section 3 motivates the two problems that we are solving: improvement of visual quality and feature extraction. Section 4 lays the mathematical foundation for the calculation of curve properties, e. g. curvature and torsion. Section 5 picks up on this and explains how to use curve properties to extract flow features. Ambient occlusion is described in Section 6. Here, we discuss ambient occlusion in more detail, including the theoretical background, and explain the structure of our voxel based algorithm. Section 7 details all necessary parts of the visualization: we shortly describe lighting and transfer functions. In the results section we show some examples of feature-based visualization of integral lines. In a separate subsection we investigate different parameter settings of our algorithm for ambient occlusion computation. We conclude the paper in the last section and suggest areas for future research.

2 Related Work

2.1 Lines

The basis for illuminated lines has been laid out by Zöckler et al. [30]. They introduced new techniques for the calculation of Phong illumination for line primitives. Later, Schussman and Ma used this approach in volume rendering [23]. Also, the techniques for basic illuminated streamlines have been updated to current graphics hardware using shaders [11].

In many cases, line data is not rendered as line primitives on graphics cards at all. Instead, visualization is achieved by either drawing tubes or, more often, self-orienting surfaces [22, 10, 3]. The latter technique has been enhanced introducing bump mapping [10], where the surfaces are rendered like tubes.

A common problem, especially when introducing transparency, is a lack of depth perception for the entire scene. This problem has been addressed using halos [13]. Special operations on

the graphic card depth buffer can improve depth perception through halos by making their width depth-dependent [3]. In this paper, we discuss an implementation which has the same properties, but uses OpenGL's line primitives instead of self-orienting surfaces.

There are further approaches connected to our visualization that we discuss in the following. The method of Mattausch et al. [13] provides means for additional seeding of streamlines in regions of interest. As already mentioned, we do not make use of the vector field and hence are able to develop methods that work on arbitrary line input. Hence, we will not discuss the differences of other methods [6, 9, 28].

Another approach developed for pathlines is provided by Shi et al. [24]. They implemented methods for querying of pathline attributes. Interactive brushing and focus+context visualization methods provide new means for investigation of pathlines.

In a two-dimensional setting, a number of methods for enhancing streamline visualization were discussed previously. Most common are methods for streamline clustering [1], or intelligent seeding of streamlines (cf. [27] and [8]). Some newer approaches extend such techniques to 3D (cf. [12]). We will not use any of these approaches in our visualization.

A further application for line visualization is rendering of white matter fibers of the human brain (e. g. [2]). Visual grouping and distinction of homogeneous paths can again be achieved by clustering. Otten et al. [15] used a combination of hint lines, colored halos, silhouettes, and outlines to enhance cluster visualization for illustrative output.

2.2 Ambient Occlusion

Ambient occlusion is a well established area of research. Ray tracing techniques do not need ambient occlusion as calculation of physically accurate lighting already includes shadows. Ambient occlusion mimics some of this behavior for arbitrary illumination models. Most non-real-time approaches use object space techniques, but for real time rendering Screen Space Ambient Occlusion (SSAO) can be applied. A short overview of the results of different approaches can be found in [14].

For rendering of hair or fur there exist no real ambient occlusion methods yet. Usually, the problem can be solved by depth-based approaches [29, 7]: Hair under the surface, for instance, is occluded and hence darker.

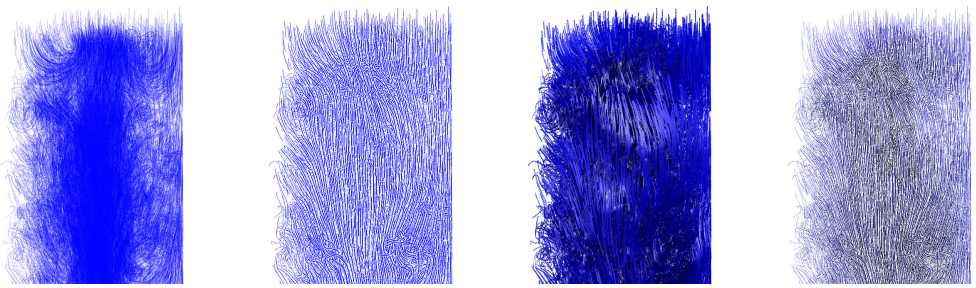
A commonly used approach to accelerate the calculation of ambient occlusion is to voxelize the scene [18]. This approach easily boosts ray tracing for the occlusion calculation by approximation of the scene.

2.3 Flow Features

Flow features have been extensively investigated. A good overview over existing methods is provided by Post et al. [17]. Related to our selection of flow features is the approach by Salzbrunn and Scheuermann [21]. The problem of vortex detection/vortex core extraction has not been solved entirely. Good approaches to this can be found in [20] and [5].

3 Motivation

The visualization of dense integral line data has many challenges. They can be split into the visualization problems and problems concerning the data directly, i. e. extraction of important flow features.



(a) Using transparency of 40% to reduce occlusion of lines: more context, no shape or depth cues.

(b) Halos have been traditionally used to provide depth cues. In this paper they are used for resolving overlaps.

(c) Enhancing illuminated lines with our ambient occlusion technique: good shape and depth cues.

(d) Combining illuminated lines, ambient occlusion intensity mapping, transparency of 60%, and transparent halos into one image.

Figure 1 Comparison of the effects of different line drawing techniques.

3.1 Improvement of Visual Quality

Obvious problems in the visualization are overlapping of lines and occlusion of lines in the back. Common approaches to solve these two problems conflict each other.

Transparency, as discussed in the related work section, is a common method to reduce occlusion (see Figure 1a). But, at the same time it reduces separation of lines at overlaps. On the other hand overlaps are resolved by using halos around the lines (see Figure 1b). This provides a better separation of lines. But again, halos add to the width of the line and occlude more lines in the back.

More important for the understanding of the visualization is the perception of shape and depth of the lines.

Illumination reveals the shape and curvature of the lines. This has been done right from the beginning (see [30]). But, this approach only works well for few lines as there is no solid visual clue for spatial ordering of lines.

Spatial ordering or depth perception has been generally solved by using halos. We already discussed one problem of using halos. Another problem is that in order to keep the spatial cues halos cannot be reasonably combined with transparency.

Our solution to this is that we use halos only for visual separation of lines. In this case halos are still useful when combined with transparency. Instead, we use ambient occlusion to improve visual spatial ordering of lines (see Figure 1c). For this, we had to adapt existing methods for the computation of ambient occlusion to work with lines and allow for partial occlusion.

The combination of these four methods, i. e. illumination, transparency, halos, and ambient occlusion, still does not reduce the information overload when used with dense line data (see Figure 1d). Therefore, we extract curve features to highlight important regions and dim out unnecessary information.

3.2 Feature Extraction

In order to provide better visualizations which focus on interesting and important regions we use transfer functions on curve properties such as length, curvature, and torsion. As we require that we do not need access to the whole simulation data but just the generated lines we use numerical methods to approximate these properties. We show with our results that this is sufficient to highlight regions of interest.

4 Curve Properties

In the following we discuss the computation of curve properties. For the scope of this paper we concentrate on integral lines computed from flow simulation. But, in general we could use arbitrary line data that represents a polyline by a sequence of points.

All the computed curve properties are scalar properties that can be easily mapped to color or transparency by using transfer functions. Beside computing the properties locally for each point of the polyline we additionally compute some global properties by computing minima, maxima, and the average.

Throughout the remainder of this section we use the following conventions for variables of a single curve or polyline:

- $v_i, i = 1, \ldots, n$ are n vertices describing the curve as a polyline
- $s_i, i = 1, \ldots, n-1$ are the segments of the line enclosed by v_i and v_{i+1}.

Local curve characteristics at a given vertex i are given the same index. Contrarily, variables without an index are always global curve parameters.

The segment length d_i is computed by the distance of neighboring vertices

$$d_i = \|v_{i+1} - v_i\|. \tag{1}$$

Assuming a reasonable sampling of the curve we get its length l by summation

$$l = \sum_i d_i. \tag{2}$$

In addition to this we also compute the minimum segment length d_{min}, the maximum segment length d_{max}, and the average segment length of a curve d_{avg}.

4.1 Derivative-based Curve Features

4.1.1 Curvature

For the input we only required that is has to fulfill C^0 continuity. Hence, we approximate derivatives through finite differences. The first derivative is the tangent t_i computed by a central difference scheme (cf. [25]). Accordingly, the second derivative c_i is computed by a first order scheme using the tangents t_i. This yields the curvature κ_i:

$$\kappa_i = \frac{\|t_i \times c_i\|}{\|t_i\|^3}. \tag{3}$$

4.1.2 Torsion

For approximation of the torsion τ_i we need the third derivative w_i along the curve. This derivative again is obtained by a first order finite difference of the second derivatives c_i. Finally, the formula for the torsion τ_i reads:

$$\tau_i = \frac{\|(t_i \times c_i) \cdot w_i\|}{\|t_i \times c_i\|^2}. \tag{4}$$

4.2 Depth

The depth of a point is computed as the distance to the camera plane. In contrast to the previous parameters this one is dynamic and has to be updated each time the camera moves. The depth parameter can be used in visualization to clip unnecessary lines that occlude flow features on the inside of the flow volume.

4.3 Ambient Occlusion

Ambient occlusion is also computed for each vertex of the polyline. Details concerning the methodology and implementation of this technique can be found in Section 6. Ambient occlusion is mostly used to enhance the illumination model to include soft shadows where the intensity of ambient lighting is reduced. Furthermore, with the right choice of parameters ambient occlusion can also encode the density of a flow field. In visualization this can be used for peeling.

5 Curve-based Flow Features

The previous section described how to compute curve properties. In this section we elaborate on the use of these properties to find and extract flow features.

Most visualizations of integral lines have access to the underlying flow field and can exactly determine properties like curvature and torsion. This is especially helpful for extraction of e. g. vortex core lines. Instead, our requirements stated that we do not rely on the underlying flow structures, but we only rely on the line data as input. Hence, we have to find an adequate mapping of flow field features to curve features. Appropriate filtering of the curve data will then provide us with a feature-based visualization.

5.1 Vortex Core Lines

There have been attempts to formalize the description of vortex core lines [16, 19]. Still, there is no agreement in the community on this. Instead, we use the intuition of Jiang et al. [4]: A vortex is characterized by a central core and swirling streamlines surrounding it.

We do not have the information about neighboring lines. But, what we can conclude from this is that the vortex core line has to have a high torsion because of continuity in the flow field and the swirling streamlines surrounding it. Still, it is not possible to extract every vortex core line since for a straight line the torsion computed according to equation 4 is always zero.

Nevertheless, we were able to reliably identify lines in the neighborhood of a vortex core. Our assumption for this is that these lines have high torsion and low curvature at the same time. This assumption works in a lot of cases, but does not guarantee to find all vortices.

5.2 Turbulence

As for vortex core lines it is hard to describe turbulences by a formal definition based on curve properties. Instead, we will again use intuition.

In a turbulent region of a flow streamlines have high rotational components. This is mapped to a high curvature in these regions. In addition, turbulences are small scale feature and particles have a low velocity magnitude. Hence, our adaptive streamline integration used for generation of the test data results in shorter streamlines than in other areas. Combining these two properties, high curvature and short total length, helps finding turbulences in the flow.

6 Ambient Occlusion

The main idea behind using ambient occlusion for line visualization is to add further spatial information through additional shading. Especially for dense line data sets these additional spatial cues will significantly enhance depth perception of the scene.

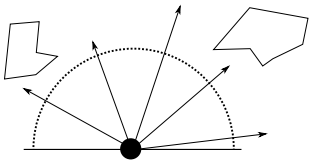
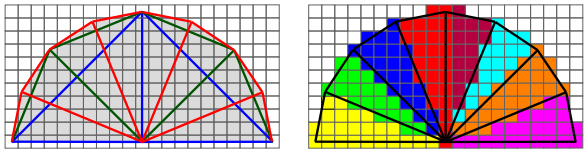


Figure 2 Tracing rays on the hemisphere of a surface for visibility calculations.



(a) (b)

Figure 3 Voxelization for ambient occlusion. (a) Two subdivision steps in 2D for one half-circle resulting in eight bins. Blue: before subdivision, green: first subdivision, red: second subdivision. (b) Putting voxels into bins. Voxels exactly on the boundary are sorted into both adjoint bins.

6.1 Challenges

Compared to existing applications of ambient occlusion the computation of ambient occlusion for lines introduces new challenges: one-dimensional geometry, no geometry normals, SSAO is not suitable, high geometric complexity, and unclear contribution to occlusion. In the following we discuss these challenges and why they pose a problem in further detail.

The common approach to compute ambient occlusion for a specific point in the scene is to use ray casting. On any hardware we can only trace a finite number of rays. Since lines have a one-dimensional geometry for most lines the computer cannot detect a collision of the ray and a line. We will see that this problem is solved by seamlessly partitioning a sphere around a point into so-called *ray bins*. Because of this seamless partition each line that has an influence on the ambient occlusion will be put into a least one ray bin.

As we describe in the background section the original idea behind ambient occlusion assumes a surface normal for the computation. Only objects on a hemisphere in the direction of the normal can have an influence on occlusion. However, one-dimensional objects do not have a surface normal. From this we derive that the ambient occlusion of the line is dependent on the viewing direction and this has to be solved.

Screen Space Ambient Occlusion (SSAO) is commonly used to accelerate the computation of ambient occlusion. As the name already tells this approach works in screen space. For many usage scenarios SSAO is a good approximation for ambient occlusion. But, our dense line data results in high depth-frequencies in the projected 2D image. This is why SSAO as an approximation will yield a completely different result than a more accurate calculation of ambient occlusion.

High geometric complexity is another problem. Since this paper is specifically about the visualization of dense line data there are a lot of objects to be visualized. Hence, tracing rays in such a scene has high computational costs. We use a common approach of voxelization of the scene to speed up computation. However, this raises the question how much a line contributes to occlusion and how to handle transparency.

The contribution of a line to occlusion is unclear due to its one-dimensionality. Intuitively a thin line inside a voxel should not set the voxel's occlusion value to one. Our approach attacks this problem and suggests a solution that we show works well with our visualization.

6.2 General Background

The general idea of ambient occlusion is to map the visibility of an object to soft shadows. Together with common shading and lighting formulas such as the Phong illumination model

this provides a more realistic and intuitive visualization.

For the computation of ambient occlusion for a certain point on a surface we trace rays in every direction of the hemisphere pointing away from the surface (see Figure 2). Then the occlusion is obtained by integrating the visibility for each ray. The general equation for this reads:

$$AO = \frac{1}{\pi} \int_{\Omega} V_{\omega}(n \cdot \omega) d\omega \tag{5}$$

where Ω is the hemisphere, ω is the direction of the ray, and $V_{\omega}(\cdot)$ the binary visibility function for this ray.

Actual implementations usually perform an approximation of the integral described in equation (5). A very common approach is using Monte Carlo Integration as a numerical method (e. g. see [18]). Another simplification is sampling the geometry on a voxel grid. This still yields good results for small voxels, but usually is faster.

6.3 Ambient Occlusion for Lines

The previous paragraph described how ambient occlusion is implemented for general geometry. As we mentioned before this can only be a basis for occlusion computations with lines. For our adaption of ambient occlusion we use a voxel based method. Instead of tracing rays, we sample hemispheres in voxel space. In the following we explain the details of our approach.

First, we give an overview over all steps needed for the computation of ambient occlusion implemented by our method:

1. Rasterize lines into voxels and count the number of lines per voxel
2. Create a subdivision of a hemisphere into ray bins for each axis
3. Create a voxel stencil for each hemisphere
4. Sort voxels of each ray bin according to the distance to the center of the hemisphere
5. Compute occlusion for each voxel
6. Store the occlusion values for each vertex of a curve

6.3.1 Rasterization

As we mentioned before we use a voxel based approach for the computation of ambient occlusion. In general we use a $128 \times 128 \times 128$ voxel grid on a unit cube – a justification for this can be found in the evaluation section. The line data is then scaled to fit the volume. As most simulations are run on boxes that are not cubes in most case we have a lot of empty space.

In this first step we take the segments of our lines and rasterize them using the 3D Bresenham line algorithm. For each traversed voxel a counter is incremented. After that the line count per voxel is scaled down to the range $[0, 1]$. This results in a discretized three-dimensional line density map. It is important that the values are normalized because in a later step we use them for our own adaption of occlusion blending.

6.3.2 Subdivision

As we already discussed it is problematic to find collisions of rays and lines. To compute the occlusion of a voxel, we have to traverse a spherical neighborhood. We therefore partition a sphere around the voxel into so-called *ray bins*. As we will see later it does not make sense to compute the overall occlusion of a point. We rather compute the occlusion for hemispheres pointing in six different directions – one along each axis and its opposite direction. Later, for

visualization three of these occlusion values are blend together depending on the viewing direction.

In order to generate the ray bins we use a subdivision scheme to approximate the spherical hull by a set of tetrahedra. We start off with a pyramid divided into four tetrahedra as a rough approximation of a hemisphere. The general concept is shown in Figure 3(a) for half a circle in 2D. Each tetrahedron (or triangle in 2D) is recursively split into four new tetrahedra (two triangles). For most visualizations two of these subdivision steps already proved to be sufficient.

6.3.3 Stencil

Each of the previously generated tetrahedra corresponds to a ray bin. In this next step we create a stencil that can be applied to any voxel to find voxels belonging to its ray bins. So, we take the relative offset of voxels to the current voxel and put them into the ray bins according to their relative position (see Figure 3(b)).

It is only feasible to use such a stencil if its size is significantly smaller than that of the entire voxel grid. The physical explanation for this is that the influence of occlusion is attenuated over distance. In our implementation we use the formula

$$w(d,r) = \begin{cases} \left(1-\frac{d}{r}\right)^4 \left(\frac{4d}{r}+1\right) & 0 \le d \le r \\ 0 & d > r \end{cases} \tag{6}$$

which has been successfully used in ambient occlusion [18]. The attenuation function $w(d,r)$ reaches zero at a previously defined maximum distance r. An evaluation of different radii r can be found in Section 8.1. Generally, voxels outside of this radius have a weight of zero and hence do not have to be included into the stencil.

6.3.4 Sorting

In this step all voxels of a ray bin are sorted into a single list with increasing distance from the center of the hemisphere. This sorting is needed for efficient blending of occlusion values. This step still operates on the stencil, thus the sorting has to be done only once.

6.3.5 Occlusion Computation

The previous steps, i. e. sphere subdivision, stencil generation, and voxel sorting for a hemisphere, is a one-time process and independent from concrete properties of a data set. Thus, this information can be stored to disc and loaded on demand for ambient occlusion computation. Nevertheless, setup time in the tables 3 and 2 show that this is not necessary.

Now, the setup for the actual occlusion computation is finished. In the actual occlusion computation pre-computed stencil information is applied to a given data set. At first, the occlusion calculation is done for each ray bin separately. Voxels are successively taken from the sorted list of the stencil. Their occlusion values are blended together from front to back just like alpha blending is used to emulate transparency. Additionally, the occlusion values are weighted according to equation 6.

For the final occlusion value of a voxel in one of the six directions we use the average of the occlusion values of all the ray bins of the corresponding hemisphere. To speed up the computation ray bins are first combined into octants of the sphere. Then, four octants are combined into the actual hemisphere to contain the occlusion value.

Finally, for faster access we store the six occlusion values for each vertex of a curve. For this, we just have to find the corresponding voxel for a vertex and transfer its occlusion values.

Our actual implementation of this method computes the occlusion for a voxel at most once and only if there is a least one vertex inside the voxel. Furthermore, each voxel has a list of all vertices that are located inside the voxel. Then, we get an additional speed up because we can just transfer the occlusion values from the voxel to all corresponding vertices.

7 Visualization

The main purpose of this paper is to find a method which provides a feasible visualization option for dealing with dense line data. The obvious solution to use transparency which allows us to show previously hidden lines. With the help of transfer functions based on curve features we can even extract flow features. However, as we discussed in the motivation, Section 3, transparency introduces new problems. Hence, we combine existing methods like illuminated lines and halos and add ambient occlusion for lines to address these problems.

In the following we discuss the basics that are needed for expedient rendering of integral lines. Since the common method to emulate transparency on the GPU is alpha blending we need depth sorting of the line segment. This computation is quite slow for the high number of lines we are using. And a pre-computation for every possible viewing direction is not feasible.

Instead, line segments are pre-sorted along the coordinate axes. During visualization the sorting that corresponds most to the current viewing direction is used. According to [30] this induces an error for at most 1% of the rendered pixels. Looking at our pictures rendering artifacts are very rare.

Many existing methods use a tube-like geometry for rending streamlines (for example see [10]). In our implementation we use OpenGL's line primitive to draw the line segments. The line width is usually given in screen space and does not scale with depth by itself. Thus, for some renderings we use the depth information to adjust the line width per segment. This approach allows for a more natural depth perception while reducing the amount of vertex information compared to rendering a more complex geometry like tubes.

Halos are most commonly used to help the viewer with depth perception. However, we figured out that combining halos with transparency destroys this perception. Nevertheless, halos are useful to visually separate lines from each other. Our implementation draws the original lines first and in a second run draws halos as a thicker line using the background's color. Using the depth buffer of the graphics card the halo line is only drawn along the sides of the actual line segment. The halo width is set as ratio to the original line's thickness. This combines well with the depth dependent scaling of the line width if needed.

7.1 Lighting

Since the first rendering of 3D streamlines [30] the Phong illumination model has been used to support perception of the line's shape. In order for the Phong illumination to work we need a surface normal that is not available for a one-dimensional geometry like a line. Hence, the normal is calculated such that it is in one plane with the tangent t_i at the position of the vertex v_i and the light vector L_i. The corresponding formula reads:

$$n_i = \frac{(L_i \times t_i) \times t_i}{\|(L_i \times t_i) \times t_i\|}. \tag{7}$$

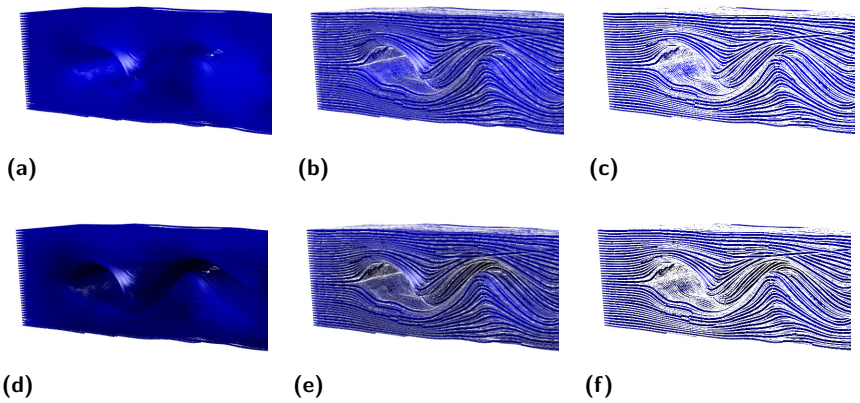


Figure 4 Flow around a cylinder. Left: no halos. Middle: transparent halos. Right: opaque halos. Top row: no ambient occlusion. Bottom row: ambient occlusion mapped to intensity.

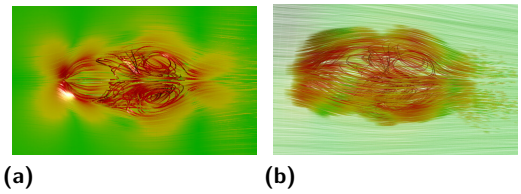


Figure 5 Flow around an ellipsoid. (a) Focusing on a region of interest by cutting off lines in front. (b) Selecting lines with higher curvature through transparency. Coloring shows the magnitude of curvature per segment.

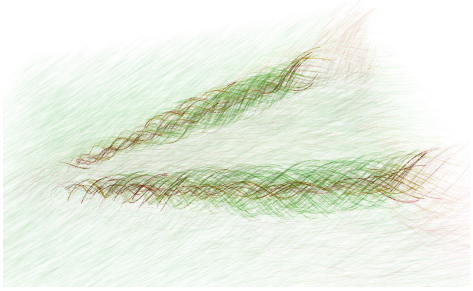


Figure 6 Extraction of vortex cores for the delta wing data set [26]. Transparency selects regions with high torsion and low curvature. Red streamlines on the center of the vortex have a high torsion, green streamlines have a low torsion and are only provided for context.

With this information we compute the illumination of each line segment.

Ambient occlusion is mostly implemented to assist with illumination. It feels natural to the human eye that objects that are partially occluded have soft shadows which means that they are darker. So, the occlusion value is used to reduce the intensity of a line segment, i. e. the influence of ambient lighting is reduced for hidden lines. The result feels more natural and enhances depth perception.

7.2 Transfer Functions and Color Maps

Transfer functions, both on transparency and color, are heavily used in our visualizations. Simply applying the same transparency to all lines reveals more lines, but does not reduce visual clutter. Hence, we are using transfer functions on line properties like e. g. curvature and torsion to focus on important flow features. There is no automated process for this. But instead, there are some general rules as described in Section 5 which combined with basic expert knowledge yield good results.

For illustrative purposes or deeper insight into the data transfer functions can also be applied for colors. Coloring is often used for visualization of streamlines. It shows how flow

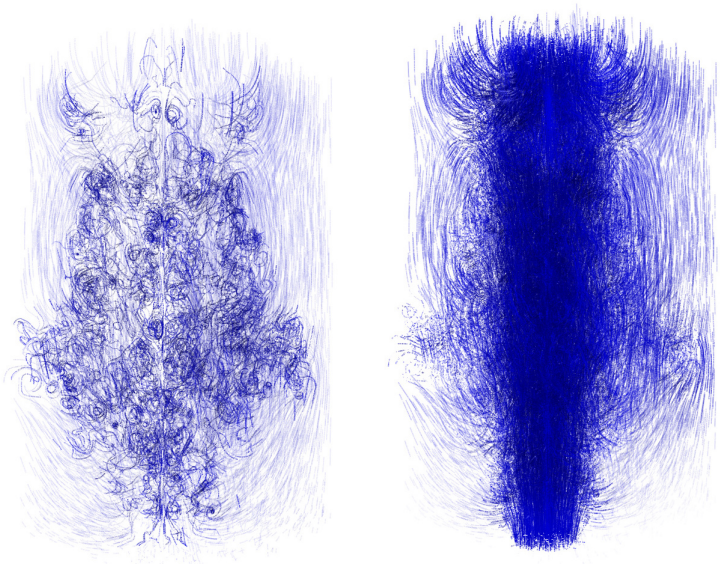


Figure 7 Extraction of turbulences in the jet stream data set. Streamlines are selected by high curvature and short total length. The right image shows the inverse selection.

properties change inside the simulation domain. With the dense line data that we have it is only useful in combination with transparency highlighting important regions.

8 Results

In a first part, we shortly discuss results using ambient occlusion for line rendering. Then, we focus in depth on the use of transfer functions on transparency to highlight important flow features. In a separate subsection we evaluate different parameter settings for the calculation of ambient occlusion.

Figure 4 shows differences for halos and ambient occlusion used for highlighting spatial relationships and separation of lines. Within the dense data set it is not possible for the viewer to clearly separate lines by just using illuminated lines (see Figure 4a). Ambient occlusion clearly improves depth perception in all cases. But without halos it only weakly separates lines in the foreground from those in the background (see Figure 4d). Opaque halos clearly give a nice illustrative visualization (Figure 4f) that can be further improved with ambient occlusion (Figure 4c). Still the insight gained from this visualization is questionable. In general we conclude that we can use intensity mapping based on ambient occlusion for improved depth perception, and transparent halos for separation of lines.

In the following, we show examples of how transfer functions on curve properties can be used to extract or highlight important flow features. We start with a generally known example.

Figure 5a shows the flow around an ellipsoid by cutting off lines in front of the region of interest. The problem here is that in order to get close to the interesting flow features we also cut off streamlines with important flow features. This focusing technique can be improved by using different curve properties. In Figure 5b we use transparency transfer functions on curvature that only select streamline segments that have a high curvature. This removes all straight lines that occluded the view before.

Another well known example for streamline visualization is the delta wing data set. Here, we extract streamlines near the vortex cores according to our description in Section 5.1. In Figure 6 we see that the extraction works well showing streamlines near the center of vortices in red. Streamlines with lower torsion are colored in green. Rendering them with higher transparency gives some context for the overall data set.

In a last example, we show the extraction of turbulent regions in a jet stream data set. As for the vortex core lines we apply the description from Section 5.2. This means that we can find turbulent streamlines by looking for high curvature and a short total length of the streamline.

8.1 Evaluation of Ambient Occlusion Computations

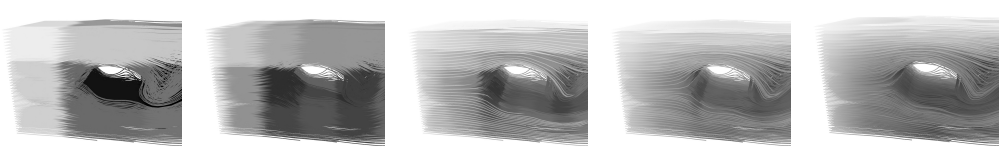
In this section we evaluate the choice of different parameters for ambient occlusion. There are three parameters that can be adjusted: the number of voxels, the maximum distance of the sphere around a voxel, and the number of subdivision steps used as approximation of the sphere. The last parameter directly influences the number of ray bins – two subdivision steps already yield 64 ray bins per hemisphere.

Although the computation of the ambient occlusion is done as a pre-processing step prior to the visualization the right choice of parameters is a trade-off between computation time – which is $\mathcal{O}(n^3)$ – and visual quality. The latter one in many cases is specific to ones visual preferences.

In a first experiment we adjusted the number of voxels used for rasterization. At the same time we linearly scaled the maximum radius of the sphere, given as number of voxels, such that the same lines have an impact on occlusion. This is the only way to make a visual comparison of the results. Our benchmark machine is a notebook with an Intel Pentium T2330 Dual Core CPU running at 1.6 GHz. Our algorithm is completely parallelized for the actual calculation of the ambient occlusion and entirely uses up the two cores. Table 1 shows the runtimes for different parameters averaged over five runs and Figure 8 shows the corresponding visualization for comparison.

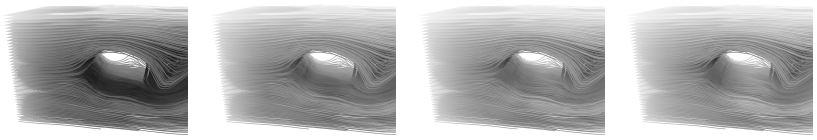
For a second test we varied just the maximum distance. Low values are especially interesting when using ambient occlusion as some measurement for density. Then, the influence on lines to occlusion is restricted to the immediate neighborhood. In visualization the interpretation as density can be used for some sort of peeling. From the results in Table 3 we can derive that the maximum distance already has a high impact on runtimes.

Finally, we tested the influence of subdivision steps of the hemisphere. From the runtime results in Table 2 we see that number of subdivisions has only a minor impact on the overall runtime. The reason for this is that we have the same amount of voxels that have an influence on occlusion regardless of the subdivision. There is only a slight overhead for descending to the next subdivision and additional recursive function calls.



(a) $8 \times 2 \times 2$ **(b)** $16 \times 4 \times 2$ **(c)** $32 \times 8 \times 2$ **(d)** $64 \times 16 \times 2$ **(e)** $128 \times 32 \times 2$

Figure 8 Grayscale maps of occlusion values. The number of voxels used for AO computation increases from left to right (compare Table 1). Low resolutions show voxelization artifacts, but 128 voxels already provides smooth results.



(a) $128 \times 20 \times 0$ **(b)** $128 \times 20 \times 1$ **(c)** $128 \times 20 \times 2$ **(d)** $128 \times 20 \times 3$

Figure 9 Grayscale maps of occlusion values for increasing subdivision steps. There are only minor differences for one and two subdivision steps, but no visible changes for two and three subdivision steps.

The images in Figure 9 show that no or only one subdivision step do not give satisfactory visual results. For a lot of data sets two subdivisions are sufficient and only for very dense data sets there is a visible difference compared to three subdivision steps. From these three tests we derived that the parameter setting of $128 \times 20 \times 2$ is a good trade-off between computation times and visual quality for most cases. Our images including a mapping of ambient occlusion to intensity enhancing the lighting model were generated with these settings.

9 Conclusions and Future Work

In the future there is clearly a need for good visualizations of flow simulations. Integral lines build a good basis for this. In contrast to previous methods we do not rely on the underlying flow field. Instead, we solely use integral lines that can be generated as output by most simulation software. Another difference is that we require a dense sampling of integral lines: without the flow field it is impossible to add information in the visualization step if you have too few lines. On the other hand, if there are too many lines, we provide a solution based on transfer functions to reduce visual clutter and highlight important flow features. To sum it up, this paper has two major contributions. Our results show that our approaches yield insightful pictures.

First, our approach improves rendering of lines, especially for dense line data. For the rendering part we have two major contributions. For one, transparency can be used while maintaining separation of lines and their spatial relationships. Transparency is really helpful for focusing on important flow features. Then, we contribute to ambient occlusion computation by extending existing methods to include AO for lines. With the combination of this and existing techniques – sometimes combining only some of them – we can provide better visualizations than before.

Second, we provide a new method to find important flow features. For this, we do not

Table 1 Runtimes for different voxel grid resolution. The increasing maximum distance has a high impact on increasing computation times.

$n_{voxels} \times max_{dist} \times n_{subdiv}$	runtime
$8 \times 2 \times 2$	0.14 s
$16 \times 4 \times 2$	0.16 s
$32 \times 8 \times 2$	0.88 s
$64 \times 16 \times 2$	27.93 s
$128 \times 32 \times 2$	1313.24 s

Table 2 Runtimes for different numbers of subdivisions of the hemisphere.

$n_{voxels} \times max_{dist} \times n_{subdiv}$	runtime	setup time
$128 \times 20 \times 0$	374.27 s	0.88 s
$128 \times 20 \times 1$	420.63 s	0.89 s
$128 \times 20 \times 2$	490.62 s	0.90 s
$128 \times 20 \times 3$	530.42 s	0.88 s

Table 3 Runtimes for increasing maximum distance. Additionally, we include the setup time for the stencil and rasterization, i. e. steps 1-4 of our method. The setup is only single threaded and the setup times are already included in the overall runtimes.

$n_{voxels} \times max_{dist} \times n_{subdiv}$	runtime	setup time
$128 \times 1 \times 2$	1.47 s	0.66 s
$128 \times 2 \times 2$	2.55 s	0.66 s
$128 \times 4 \times 2$	8.76 s	0.66 s
$128 \times 8 \times 2$	46.22 s	0.68 s
$128 \times 16 \times 2$	282.06 s	0.78 s
$128 \times 32 \times 2$	1314.10 s	1.72 s

rely on the underlying flow field information. This makes it easier to integrate with existing simulation software. Instead, we use only integral lines for visualization. In order to have all the information that is needed we require a dense sampling of integral lines. For the flow expert we provide the steps that are necessary to find vortices and turbulent regions.

Still, there is a lot of improvement for future research. For one, ambient occlusion can be used with a small surrounding sphere to compute the local density. These regions might incorporate interesting flow features as well. For example, regions with a high density can point to sources, sinks, or separation lines.

Furthermore, in this paper we just used properties on single curves. Combining curve properties with neighboring integral lines can give even more insight into the data. Especially, the definition of vortices is easier if the neighborhood is included. Then, it might also be possible to detect vortex cores that are straight lines and hence do not have a torsion according to our calculation from Section 4.

References

1 Bernardo Silva Carmo, Y H Pauline Ng, Adam Prügel-Bennett, and Guang-Zhong Yang. A data clustering and streamline reduction method for 3d mr flow vector field simplification. *Lecture Notes in Computer Science*, 3216:451–458, 2004.

2 Frank Enders, Natascha Sauber, Dorit Merhof, Peter Hastreiter, Christopher Nimsky, and Marc Stamminger Stamminger. Visualization of white matter tracts with wrapped streamlines. *Visualization Conference, IEEE*, 0:51–58, 2005.

3 Maarten H. Everts, Henk Bekker, Jos B.T.M. Roerdink, and Tobias Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics*, 15:1299–1306, 2009.

4 Ming Jiang, Raghu Machiraju, and David Thompson. A novel approach to vortex core region detection. In *Proceedings of the Symposium on Data Visualisation 2002*, VISSYM '02, pages 217–ff, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

5 Ming Jiang, Raghu Machiraju, and David Thompson. Detection and visualization of vortices. In *The Visualization Handbook*, pages 295–309. Academic Press, 2005.

6 Bruno Jobard and Wilfrid Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proceedings of the 8th Eurographics Workshop on Visualization in Scientific Computing*, pages 45–55, 1997.

7 Tae-Yong Kim and Ulrich Neumann. Opacity shadow maps. In *In Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 177–182. Springer-Verlag, 2001.

8 Liya Li, Hsien-His Hsieh, and Han-Wei Shen. Illustrative streamline placement and visualization. In *PacificVis '08: Proceedings of the IEEE Pacific Visualization Symposium 2008*, pages 79–86, 2008.

9 Zhanping Liu, Robert Moorhead, and Joe Groner. An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 12:965–972, September 2006.

10 Kwan-Liu Ma, Greg Schussman, Brett Wilson, Kwok Ko, Ji Qiang, and Robert Ryne. Advanced visualization technology for terascale particle accelerator simulations. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, Supercomputing '02, pages 1–11, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

11 Ovidio Mallo, Ronald Peikert, Christian Sigg, and Filip Sadlo. Illuminated Lines Revisited. In *IEEE Visualization*, pages 19–26, 2005.

12 Stephane Marchesin, Cheng-Kai Chen, Chris Ho, and Kwan-Liu Ma. View-dependent streamlines for 3d vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 16:1578–1586, November 2010.

13 Oliver Mattausch, Thomas Theußl, Helwig Hauser, and Eduard Gröller. Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. In *Proceedings of the 19th spring conference on Computer graphics*, SCCG '03, pages 213–222, New York, NY, USA, 2003. ACM.

14 Morgan McGuire. Ambient occlusion volumes. In *Proceedings of High Performance Graphics 2010*, pages 47–56, June 2010.

15 Ron Otten, Anna Vilanova, and Huub van de Wetering. Illustrative white matter fiber bundles. *Comput. Graph. Forum*, 29(3):1013–1022, 2010.

16 Luis M. Portela. *Identification and Characterization of Vortices in the Turbulent Boundary Layer*. PhD thesis, Stanford University, December 1998.

17 Frits H. Post, Benjamin Vrolijk, Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. Feature extraction and visualisation of flow fields. In *In Eurographics 2002 State-of-the-Art Reports*, pages 69–100, 2002.

18 Christoph Reinbothe, Tamy Boubekeur, and Marc Alexa. Hybrid ambient occlusion. *EUROGRAPHICS 2009 Areas Papers*, 2009.

19 Stephen K. Robinson. Coherent motions in the turbulent boundary layer. *Annual Review of Fluid Mechanics*, 23(1):601–639, 1991.

20 I. Ari Sadarjoen and Frits H. Post. Detection, quantification, and tracking of vortices using streamline geometry. *Computers & Graphics*, 24(3):333–341, 2000.

21 Tobias Salzbrunn and Gerik Scheuermann. Streamline predicates. *IEEE Transactions on Visualization and Computer Graphics*, 12:1601–1612, November 2006.

22 Greg Schussman and Kwan-Liu Ma. Scalable self-orienting surfaces: A compact, texture-enhanced representation for interactive visualization of 3d vector fields. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, PG '02, pages 356–365, Washington, DC, USA, 2002. IEEE Computer Society.

23 Greg Schussman and Kwan-Liu Ma. Anisotropic volume rendering for extremely dense, thin line data. In *Proceedings of the conference on Visualization '04*, VIS '04, pages 107–114, Washington, DC, USA, 2004. IEEE Computer Society.

24 Kuangyu Shi, Holger Theisel, Helwig Hauser, Tino Weinkauf, Kresimir Matkovic, Hans-Christian Hege, and Hans-Peter Seidel. Path line attributes – an information visualization approach to analyzing the dynamic behavior of 3d time-dependent flow fields. In *Topology-Based Methods in Visualization II*, pages 75–88, 2009.

25 John C. Strikwerda. *Finite difference schemes and partial differential equations*. Wadsworth Publ. Co., Belmont, CA, USA, 1989.

26 X. Tricoche, C. Garth, T. Bobach, G. Scheuermann, and M. Rütten. Accurate and efficient visualization of flow structures in a delta wing simulation. In *Proceedings of 34th AIAA Fluid Dynamics Conference and Exhibit*, number AIAA Paper 2004-2153, June 2004.

27 Vivek Verma, David Kao, and Alex Pang. A flow-guided streamline seeding strategy. In *In Proceedings IEEE Visualization 2000*, pages 163–170.

28 Xiaohong Ye, David Kao, and Alex Pang. Strategy for seeding 3d streamlines. *Visualization Conference, IEEE*, 0:471–476, 2005.

29 Cem Yuksel and John Keyser. Deep opacity maps. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2008)*, 27(2):675–680, 2008.

30 Malte Zöckler, Detlev Stalling, and Hans-Christian Hege. Interactive visualization of 3d-vector fields using illuminated stream lines. In *IN PROCEEDINGS OF IEEE VISUALIZATION '96*, pages 107–113, 1996.

Texture-based Tracking in mm-wave Images

Peter Salz[1], Gerd Reis[2], and Didier Stricker[2]

1 **University of Kaiserslautern**
Kaiserslautern, Germany
salz@rhrk.uni-kl.de
2 **Augmented Vision Group**
DFKI GmbH, Kaiserslautern, Germany

Abstract

Current tracking methods rely on color-, intensity-, and edge-based features to compute a description of an image region. These approaches are not well-suited for low-quality images such as mm-wave data from full-body scanners. In order to perform tracking in such challenging grayscale images, we propose several enhancements and extensions to the Visual Tracking Decomposition (VTD) by Kwon and Lee [6]. A novel region descriptor, which uses texture-based features, is presented and integrated into VTD. We improve VTD by adding a sophisticated weighting scheme for observations, better motion models, and a more realistic way for sampling and interaction. Our method not only outperforms VTD on mm-wave data but also has comparable results on normal-quality images. We are confident that our region descriptor can easily be extended to other kinds of features and applications such that tracking can be performed in a large variety of image data, especially low-resolution, low-illumination and noisy images.

1998 ACM Subject Classification I.4.8 Scene Analysis – Time varying imagery, Tracking

Keywords and phrases Visual Tracking Decomposition, low-quality images, texture features, mm-wave imagery

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.89

1 Motivation

To improve airport security, the detection of threats (especially non-metallic ones) attached to a person's body is very important. Full-body scanners that use mm-wave radiation are currently in development with the goal of automated threat detection. Smith Heimann GmbH provided us with several image sequences obtained from such scanners that show a person (carrying one or more threats) with raised arms performing a full rotation around the vertical axis. Our goal for the master's thesis by Salz [13] was to achieve an automated tracking of a threat through the sequence using a manually selected initial bounding box of the threat.

The grayscale images are very challenging with respect to the tracking method since they have a low resolution (with even smaller object sizes) and low contrast while several artefacts further reduce quality: Even between a very small number of frames we observe severe appearance and illumination changes, whole body parts are shadowed for some time, metal objects produce bright responses, and flares decrease the already small SNR.

Since these image sequences are confidential property of Smith Heimann GmbH, we are not able to provide any graphical descriptions. The sample images of Figure 1[1] should give a

[1] http://upload.wikimedia.org/wikipedia/commons/d/d8/Mmw_large.jpg

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 89–101
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
IRTG 1131

rough overview of the image quality, although no threats are shown and real-world scanner data are usually much noisier with less contrast and illumination.

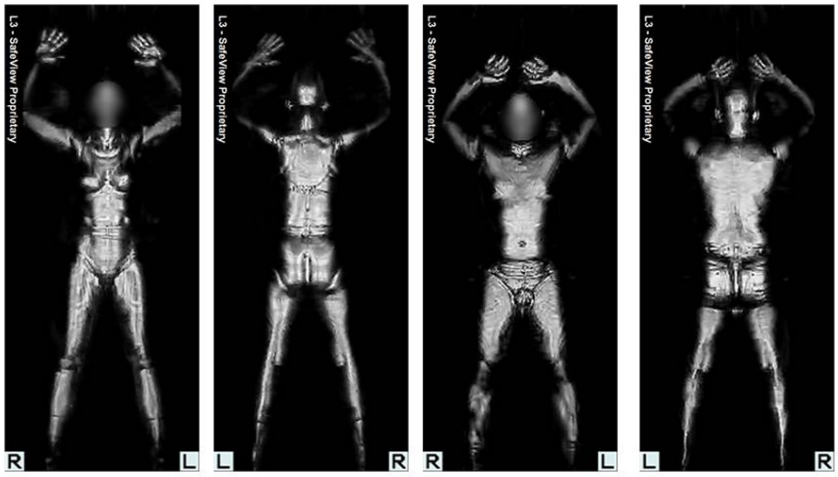


Figure 1 Sample mm-wave images of a female and a male person[1].

2 Overview

2.1 Contributions

We conducted a thorough analysis of several detection and tracking algorithms with respect to their applicability to mm-wave sequences (as is described in detail in [13]). Since we discovered several shortcomings of Visual Tracking Decomposition (VTD), we propose some enhancements (with respect to general tracking) and extensions (to apply it to other kinds of image data), such as a more sophisticated observation and motion model, better sampling and tracker interaction and integration of our own region descriptor.

This region descriptor is based on texture features as known from ultrasound texture discrimination and is computed using a patch decomposition of an image region, taking a carefully designed weighting scheme into account. We adapted the Förstner distance to compare sample and model covariance matrices computed from these patches in a robust way.

2.2 Results

As presented in Section 5, we achieved quite promising results on mm-wave sequences, especially compared to other approaches like VTD. When applied to the same data Kwon and Lee used for VTD, our method is comparable to it in terms of location error and sometimes even has better results.

In other low-resolution, low-light grayscale image data we expect our method to be superior to VTD as well, since the texture-based region descriptor is generally well suited for objects that are not easily represented by color and edge features.

2.3 Paper Organization

This paper is organized as follows: Section 3 introduces all related publications that were the basis for the proposed algorithm. Section 4 describes our algorithm in detail, presents all the sub-parts and gives an overview of the whole algorithm. In Section 5, some results are presented, including a report on mm-wave tracking and a comparison to Visual Tracking Decomposition on normal data. Finally, Section 6 concludes the paper, outlining our contributions and future work.

3 Related Work

3.1 Introduction

There are two dominant approaches for matching an image region to a template. In detection methods, the target object's position, scale, rotation, and other parameters can be almost arbitrary, i.e. the relation between the template and the target image is not known. This is especially helpful when dealing with images of the same scene, taken in different camera configurations, at a different time or in different circumstances (e.g. lighting). One of the major drawbacks is the global and usually exhaustive search in the target image. This is very expensive and often constrains the choice of a region descriptor, but more importantly, it might cause confusion of similar looking objects.

In video sequences on the other hand, tracking is performed in adjacent frames where only small changes both in appearance and in location are expected. This leads to the use of a motion model to predict the movement from frame to frame and an observation model to predict and cope with appearance changes.

Please refer to [13] for a much more extensive description and analysis of the presented methods.

3.2 Region Covariance

Region Covariance is a detection method proposed by Tuzel, Porikli and Meer [14] that uses the covariance matrix of pixel-based statistics as a region descriptor. It incorporates different features like position, color, and partial derivatives, but can easily be extended to other kinds of features. To make an exhaustive search in scale-space feasible, the covariance matrix of a region is computed using Integral Images, proposed by Viola and Jones [15]. This intermediate representation can be pre-computed for the whole image such that a covariance matrix for a region can be computed in constant time. To compare covariance matrices of a template and target region, the Förstner distance (Förstner and Moonen [4]) is used. This distance is based on generalized eigenvalues and assumes that all eigenvalues are positive, which is usually not valid in noisy, real-world data. Therefore, we propose an adapted Förstner distance computation that only considers a few dominant eigenvalues, as presented in Section 4.4.

3.3 Incremental Learning

In contrast to Region Covariance, Incremental Learning is a multi-template tracking method proposed by Ross et. al. [12] that maintains an appearance model based on previous observations. Their algorithm starts with a single template and incrementally updates the appearance model while down-weighting older observations with a forget factor. An intensity vector for all pixels in the image region is used as a region descriptor ($\mathbf{I}$), while the

appearance model consists of the eigenbasis U from $A = U\Sigma V^T$ with $A = [\mathbf{I}_1, ..., \mathbf{I}_n]$. Their update procedure incorporates new observations without re-computing the whole eigenbasis. A Gaussian motion model and a particle filter with the state consisting of translation, rotation, scale, aspect ratio and skew are used for tracking. See for example the tutorial by Arulampalam, Maskell and Gordon [1] for an overview of particle filters.

The method has some disadvantages, such as the high-dimensional feature vector that only includes raw intensity and requires a scaling of each target region to a common size. Furthermore, the Mahalanobis distance is used as the dominant distance measure for the observation model, utilizing a distance from the mean observation vector. This mean might not be very expressive, since the implicit assumption of a normal distribution of observation templates is not always valid in low-quality, noisy image data. The eigenbasis computation using Principal Component Analysis (PCA) allows for fast frame rates, since only a partial update of the covariance matrix is necessary for new observations. On the other hand, since we use a sparse variant of PCA (SPCA, see below), this is not feasible and a full covariance matrix computation is needed for each model update.

The incremental update procedure, the forget factor, and the subsequent implicit limitation of observation model size are valuable contributions to multi-template tracking and are also utilized in our method (see Section 4.3).

3.4 Visual Tracking Decomposition

Visual Tracking Decomposition (VTD) is a multi-template, particle-filter based tracking method proposed by Kwon and Lee [6]. They partition the observation model into different sub-models which are then utilized by basic trackers with different motion models. As state, the position and a single scale parameter are used, while the HSV representation (for grayscale images: intensity) and Sobel-based edge strength serve as features. To compute similarity of an observation to a sub-model, the Diffusion Distance by Ling and Okada [7] is used. Motion models consist of Gaussians with different variances (representing small and abrupt displacements between frames) and the partitioning of the observation model is computed by Sparse Principal Component Analysis (SPCA, d'Aspremont et. al. [3]) of the (non-centered) covariance matrix. SPCA produces sub-models that capture the most severe appearance changes, have a compact representation, and are almost complementary.

The eight basic trackers (with four observation and two motion models) generate Markov chains of samples using the Metropolis Hastings algorithm [5]. Interactive Markov Chain Monte Carlo (IMCMC) by Campillo et. al. [2] is used to fuse those basic trackers. One tracker proposes a state and the others decide whether to leap to this state or maintain their own Markov chain.

The authors report that their method can cope with severe appearance changes (because of the multi-template model), occlusion and illumination changes (robust features), and abrupt motion (different motion models). However, variance parameters for the motion models and the scaling parameter are adapted to the specific image sequence and the implementation deviates from the description in the paper. First, the motion model is centered at the original object position, not the current one, which is helpful for most of the sample sequences where the camera follows the object of interest. Second, basic trackers propose new states in each iteration, although in the paper this sampling is only done with a certain probability or an interaction step is performed. Furthermore, VTD has several other drawbacks that are addressed in our method. The features are not applicable to low-quality images. The old position of a basic tracker is ignored when sampling new states and Metropolis-Hastings sampling yields an acceptance rate close to 100% (while theoretically it should be around

23%, see Roberts et. al. [11]). The entries of sparse principal components are not utilized to weight their influence, although their variance contribution differs significantly.

3.5 Texture Features

In most tracking applications, a combination of color or intensity and edge information is used to compute feature vectors of an image region. For low-quality, noisy and colorless images, these region descriptors perform very poorly. In the field of ultrasound texture classification, another set of features has been proposed that relies on intensity and several intensity transformations (gradient magnitude, difference to mean, horizontal and vertical residuals), as described by Muzzolini et. al. [10] and Liu and Jernigan [9].

The following set of features both from the spatial as well as from the frequency domain has been chosen for our method: The Kolmogorov-Smirnov distance between the cumulative distributions (approximated by histograms) of each target patch and a representative sample (which is known to belong to the texture class of interest) and the standard deviation of each patch are spatial features. Frequency features include the energy at the major peak of the normalized power spectrum (given by the Fourier transform), the Laplacian of major and secondary peak, relative orientation of major and secondary peaks, isotropy of the power spectrum, and relative energy and entropy of inner regions of the power spectrum.

3.6 Sparse Principal Component Analysis

One disadvantage of Principal Component Analysis (PCA) is the non-sparsity of entries. Just thresholding (such that small entries are ignored) is not sufficient for dimensionality reduction. Therefore, a variant of PCA is used, which captures most of the variance in the data, but with a desired sparsity of principal components. It is called Sparse Principal Component Analysis (SPCA). Please refer to [13] for a detailed description.

In our method, the sparse principal components (SPCs) are computed using an iterative elimination strategy (Wang and Wu [16]) with project deflation to eliminate the influence of a pseudo-eigenvector. A single parameter controls the percentage of variance that this SPC should preserve.

4 Texture-based Tracking using Visual Tracking Decomposition

4.1 Model Representation

The tracking result for a single frame consists of a state and the corresponding observation or feature vector. In the original VTD approach, only a single scale parameter was used, but in many applications an object might change its scale in both dimensions, so we propose to incorporate a total of four state variables: x,y position of the region center and x,y scaling parameters for the scale change with respect to the first template. VTD also has a fixed observation model size of ten templates, while older templates are discarded and all current templates are treated equally, relying on SPCA to sort out features and time steps that do not capture important appearance changes. Instead, we make use of the forget factor, introduced in [12], to down-weight older samples and thus implicitly restrict the template size.

Since the observation model is supposed to capture appearance changes, a representative sample has to be chosen during model update (using the terminology introduced in [10]). In our approach, a manually selected template is initially available, which serves as the representative until the first model update. This representative is supposed to replace the

model mean, since it is a true part of the model (describing the object best in a certain point in time) and thus is more useful for computing the variance in the template dataset. The representative is described in Section 4.3.

The motion model is very similar to the one in VTD, although some of its shortcomings have been addressed. As proposal density functions for the Metropolis-Hastings algorithm, Gaussian functions with different parameters have been commonly used and are easily sampled. In an initialization phase, basic trackers are spread out from the current location. If they encounter a better state than the old one, they move to this state and continue sampling from there. In contrast to VTD, our motion models try to predict the movement in the current frame based on previously detected movements. The first motion model is centered at the current location of the basic tracker and has a location variance that depends on the absolute difference of the current and previous position (the larger this difference, the larger the variance). The second motion model assumes that an object continues its movement from the previous frame, so the mean of the Gaussian is shifted in that direction, using the same variances as the first motion model.

The eight basic trackers are constructed from combinations of those two motion models and the four basic observation models.

4.2 Feature Computation

Let R be an image region of size $W \times H$. In VTD, the different features were computed as a matrix of all features and pixels. For our texture-based approach the whole region is not discriminative enough since the texture features are supposed to be constructed from small patches. Therefore, we partition the region into five smaller, overlapping patches. These patches are evenly sized, and four of them fill the whole region while overlapping each other slightly, thereby ensuring that structures near patch borders are not omitted. The fifth patch is placed in the center of the region, overlaps with all other patches, and usually does not contain any irritating non-object pixels from the borders of the region.

Since the center patch is most likely to contain only parts of the object of interest, it is assigned a dominant weight of 0.5, while the remaining four patches get a weight of 0.125 each. For the patch decomposition we chose an overlap parameter $\omega \in [0, 1]$, which we set to $\omega = 0.1$. The patch width (for all patches) is $W_\omega = [0.5 \cdot (1 + \omega) \cdot W]$, with $[\cdot]$ being the round operator, and the patch height is computed accordingly.

For each of these five patches, the $d = 21$ features (see Section 3.5) are computed, resulting in a 21×5 feature matrix $F(R)$ for the region R. Note that this matrix is independent from the region size, so no scaling is necessary compared to the Incremental Learning approach in [12].

4.3 Model Update using SPCA

Observation model size

The observation model is initialized with the manually selected reference template, so for the first few frames our method is equal to a single template region matching. After m frames, we therefore gathered $m + 1$ templates from which the new observation model is computed. With a forget factor $f \in [0, 1]$ and a total number of templates t, the new effective database size after the update becomes $t \leftarrow f(t - m) + m$. In VTD, m is set to two and $f = 1$, while the database size is restricted to ten templates. In Incremental Learning, $m = 5$ and $f = 0.95$ with an effective database size of 100 templates. For our approach we set $m = 3$ and $f = 0.8$ because of the severe appearance changes in mm-wave sequences, resulting in a maximum

database size of $\frac{m}{1-f} = 15$. Therefore, if the database size is larger than the effective size, the oldest templates are discarded (because they do not contribute to the model anymore) and the weights of remaining templates are re-normalized.

Representative sample

At the beginning of each model update, a new representative template is computed as follows:

1. Find median m_k^p for each feature $1 \leq k \leq d$ and patch $1 \leq p \leq 5$, resulting in a $d \times 5$ matrix of median values.
2. Compute the $d \times d$ covariance matrix for this median and for each template from their patches.
3. Compute Förstner distance between covariance matrices of median and each template.
4. The template with smallest distance to the median is chosen as new representative F_{Rep}.

This representative template is a true part of the model which describes the object best at a certain time step, and it also has a minimal distance to the median, so any template with a large variance with respect to this representative describes a certain appearance change of the object.

Template covariance matrix

Each template has a feature matrix $F_l, l = 1, ..., t$ and a weight w_l (the maximum a-posteriori value from the tracking process), which gets down-weighted by powers of the forget factor. For $l = t-1, ..., t-m$ for example, $f_l = f^0 = 1$, while the exponent is increased by one for every m weights.

The template covariance matrix C_M is constructed from the $d \times 5t$ data matrix $D_M[k, 5 \cdot l + p] = \sqrt{w_l \cdot f_l}\,(F_l[k,p] - F_{\text{Rep}}[k,p])$ with $1 \leq k \leq d$ features, $1 \leq l \leq t$ time steps, $1 \leq p \leq 5$ patches, and F_l the l-th template's feature matrix. D_M is normalized by $\frac{1}{\sqrt{\sum_l f_l \cdot w_l}}$, resulting in the template covariance matrix

$$C_M = \frac{1}{1 - \frac{\sum_l (w_l f_l)^2}{\left(5 \cdot \sum_l w_l f_l\right)^2}} D_M \cdot D_M^T. \tag{1}$$

SPCA

The purpose of SPCA is to reduce the number of features and time steps that contribute to a submodel to only those that explain severe appearance changes (with respect to the representative), and to split the observation model into different submodels for each basic tracker. To compute sparse principal components from the template covariance matrix, we use the Iterative Elimination algorithm by Wang and Wu [16] since it is easy to compute and is controlled by a single parameter that controls the amount of variance a sparse principal component (SPC) should contain.

For the $r = 4$ observation models, we need the first four SPCs of C_M, where the first SPC captures 40% percent of the whole variance and the remaining SPCs 24%, 9.6% and 3.84%, respectively, so 77.54% of the total variance is captured by the basic observation models. The corresponding eigenvalues serve as weights for each SPC (normalized such that they sum up to one) and the first SPC u_1 is obtained from the full covariance matrix C_M. For the other SPCs u_c, a projection deflation $C_M \leftarrow \left(I - u_{c-1}u_{c-1}^T\right) C_M \left(I - u_{c-1}u_{c-1}^T\right)$ is used to remove the influence of other pseudo-eigenvectors u_{c-1}.

Feature reduction

For each basic observation model $1 \leq c \leq r$, we obtained a sparse principal component u_c. If its k-th entry is considerably larger than zero (where $1 \leq k \leq d$), feature k is added to the sub-model c with weight $u_c[k]$. Sometimes only one feature is dominant with a weight close to one. Unfortunately, in that case the time step selection described in the next section does not work, so we add the next feature $k + 1 \bmod d$ with weight 0.1.

After the feature selection, each submodel c contains d_c features, while some features may contribute to more than one model since the SPCs are not completely orthogonal (in contrast to true principal components).

Time step selection

To further reduce the complexity of the basic observation models, we use only those time steps that exhibit a large variance with respect to the representative. This differs from the VTD approach where feature reduction and time step selection are performed in a single application of SPCA. In contrast, due to the complex but more realistic weighting scheme in our method, we apply an additional SPCA to select relevant time steps.

For each remaining feature of a basic observation model we compute the $t \times t$ covariance matrix from the five patches of templates. From SPCA we obtain the first SPC that captures 75% of variance in all time steps and store it in a $t \times d_c$ weight matrix T_c.

For each time step we decide to keep or discard it:

1. $t \times 5$ data matrix D_t for each feature $1 \leq k \leq d_c$, weighted as above, each row $1 \leq l \leq t$ is multiplied by $w_l \cdot f_l$ (w_l is the weight of the time step, f_l influence of the forget factor) $\Rightarrow t \times t$ covariance matrix $C_t \Rightarrow$ SPC $u_k \Rightarrow k$-th column of T_c.
2. For each row of T_c take the average (weighted with feature weights from feature reduction), if larger than 0.15, add time step to basic observation model c with weight w_l multiplied by weighted average.
3. Also add the representative time step to each model such that each basic model contains a representative observation and appearance changes.

Model update

For each basic observation model c we now have d_c features and t_c time steps, so all weights are normalized to sum up to one and the normalized eigenvalue from the c-th SPCA is the weight of that model. This leads to a covariance matrix C_c for that basic model, which is positive semi-definite, so in practice not all singular values are non-zero. For the sample weight computation below, we eventually need the inverse of the covariance matrix, so using a Singular Value Decomposition $C_c = U\Sigma V^T$ we construct a pseudo-inverse $C_c^+ = V\Sigma'^T U^T$, where Σ' contains the reciprocal of each diagonal element that is larger than some $\epsilon > 0$. This pseudo-inverse is stored in the observation model since it is computed only during the model update but is used heavily during the sample weight computation.

4.4 Sample Weight Computation

Förstner distance

To determine the confidence that a region contains the object of interest (i.e. belongs to the observation model) we need a weight $w \in [0, 1]$ that is proportional to some distance that measures similarity between a sample and the model. For VTD the Diffusion Distance [7] was

chosen that compares histograms of the feature vectors. Our region representation instead consists of five feature vectors, one for each patch, that are stored in a feature matrix. Since these features have very different ranges (their theoretical bounds are known, for example intensity is in $[0, 255]$, but practical upper bounds are difficult to define), a normalization is applied based on empirical bounds. Note that sometimes a patch has a very low mean intensity (i.e. is almost black) such that the feature computation would not be robust. In such a case we discard the sample immediately by setting the weight to zero.

Each basic observation model c has t_c templates, each with the same d_c features, including the representative. The covariance matrix of feature matrix F for a sample region is computed using the patch-specific weights w_p (0.5 for the central patch and 0.125 for the others) and the mean (for feature k: $\mu_k = \frac{1}{5}\sum_{p=1}^{5} w_p \cdot F[k,p]$) from the data matrix $D[k,p] = \sqrt{w_p}(F[k,p] - \mu_k)$, resulting in $C_S = \dfrac{1}{1 - \sum_{p=1}^{5} w_p^2} D \cdot D^T$ (of size $d_c \times d_c$).

For each template $1 \leq l \leq t_c$ the covariance matrix C_l has been precomputed at the model update, so the Förstner distance is $FD(C_S, C_l) = \sqrt{\sum_{i=1}^{\dim_{C_S}} \log^2 \lambda_i}$, where $\dim_{C_S}$ is the number of eigenvalues of C_S that are larger than $\epsilon = 10^{-6}$ and λ_i is the i-th eigenvalue of matrix $C = C_S \cdot C_l^+$. By restricting the distance computation to only the $\dim_{C_S}$ largest eigenvalues we improve robustness since a lot of the smaller eigenvalues are very close to zero and therefore distort the distance function.

Let w_l be the weight from the SPCA, then the overall distance of a sample region to the model is the weighted average $d = \sum_{t=1}^{t_c} w_l \cdot FD(C_S, C_l)$.

Weight map

To transform the distance into a meaningful and expressive weight $w \in [0, 1]$, a weight map needs to be designed. In VTD, a simple exponential function was used to map the Diffusion Distance to a weight. In our low-quality images and with our specific features and distance, we observed that weights decrease too fast to very small values when using an exponential function. We also noted that there is a sharp boundary between samples that should be accepted or rejected. We therefore decided to use a sigmoid function since it should only take positive values, its maximum value should be at zero (because a distance of zero is a perfect match), for values larger than zero we expect a decrease in weights with a sharp transition from large to small weights.

This results in the following weight map with empirically tuned parameters:

$$w(d) = \frac{1}{1 + e^{4(d-2)}} \tag{2}$$

4.5 Tracker Interaction

The $r \cdot s = 8$ basic trackers independently produce samples from the same distribution. The interaction scheme used here is similar to VTD, but with a lower acceptance rate of samples, a more realistic spread of initial samples, a better state evaluation of other trackers, and a better weighting of a tracker's contribution.

In each iteration the interaction step is performed with probability α, which is initially set

to one and linearly decreases to 0.5. The sum of weights of all basic trackers is $s_w = \sum_{i=1}^{rs} \frac{1}{s}\lambda_i w_i$, where λ_i is the normalized eigenvalue (i.e. the significance of contribution) and w_i is the weight a tracker has at the current state.

For each basic tracker $1 \leq o \leq rs$ we have $p_o = \frac{1}{s}\lambda_o w_o \frac{1}{s_w}$, which, in the original VTD approach, is the probability that a proposed state is accepted by tracker c. It is not clear, however, whether this will indeed improve the state of this tracker. Therefore, we decided to compute the weight of tracker c at the state proposed by tracker o with probability p_o and accept it only if the weight is larger than the current one, thereby guaranteeing that c actually improves its state.

We also replace the Metropolis-Hastings sampling algorithm with its multiple-try variant [8] to increase confidence in a proposed sample by drawing more intermediate samples. This requires less iterations (indeed we reduce VTD's 100 iterations to only 20, because the maximum a-posteriori is usually found very fast) and also produces better samples.

First, ten samples y_i are drawn from the motion model (which serves as proposal density function) associated with basic tracker c and compute the weight w_{y_i}. The samples are sorted by their weight, starting with the largest one, and one sample $y = y_i$ is selected with probability w_{y_i}. Then we center the motion model at y, draw nine additional samples x_j and set x_{10} to the current state of c. We compute weights w_{x_j} and the acceptance ratio $\gamma = \frac{\sum_{i=1}^{10} w_{y_i}}{\sum_{j=1}^{10} w_{x_j}}$. The state y is accepted with probability γ and y is added to the list of accepted samples. The accepted sample with largest weight is then selected as the maximum a-posteriori (MAP) estimate. This weight is added to the list of weights of all templates, which are then re-normalized.

Note that the acceptance rate for a Gaussian proposal density function should theoretically be around 23% (compare Roberts, Gelman and Gilks [11]). For VTD we observed acceptance rates of more than 95%, while our method mostly achieves acceptance rates between 17% and 40%.

4.6 Algorithm Summary

For every frame of the sequence:

1. Compute global intensity transforms.
2. After every 3 frames update observation model using SPCA.
3. Perform initial sampling using extended motion model.
4. Repeat for each iteration:
   - Tracker interaction with probability $\alpha \in [0.5, 1]$.
   - Reduce α linearly.
   - State sampling using Multiple Try Metropolis-Hastings algorithm.
5. Select maximum a-posteriori estimate from accepted states.

5 Results

Application to mm-wave data

As mentioned before, we cannot provide any graphical examples for mm-wave data, but we achieved promising results in our experiments, especially in comparison to the original VTD method which only used intensity and edge strength as features. To our knowledge, no other method is capable of dealing with this kind of challenging data.

Figure 2 shows a comparison of absolute pixel errors of the bounding box center with respect to a manually selected ground truth.

Note that in mm-wave sequences threats are only visible for a few frames and severe appearance and illumination changes occur even between a small number of frames. Additionally, threats might be subject to imaging artefacts and their appearances often do not deviate much from surrounding regions.

For more evaluations of tracking in mm-wave sequences, please refer to [13].

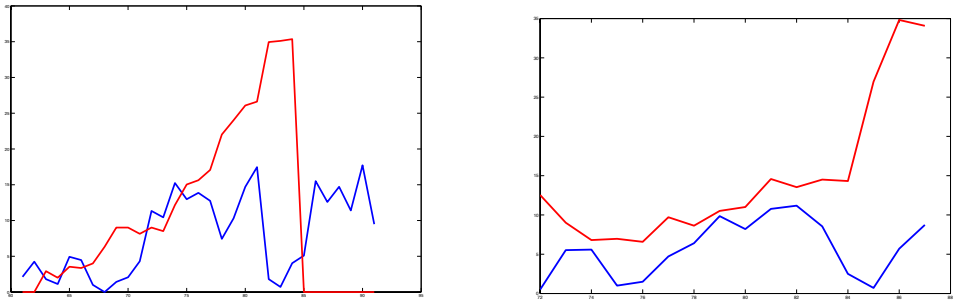
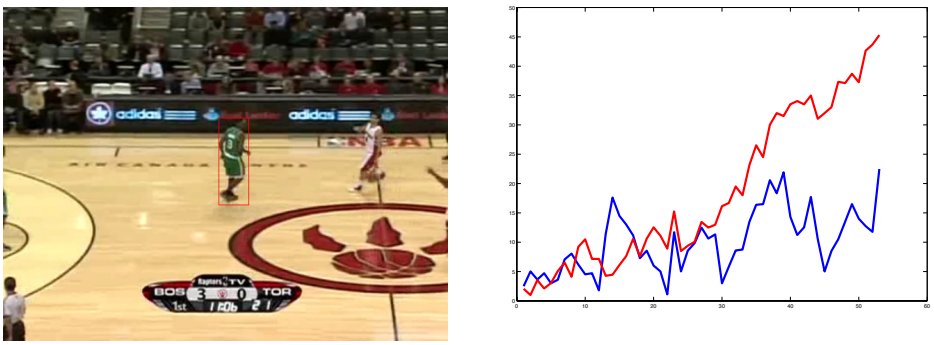


Figure 2 Results for two mm-wave sequences. The blue graph shows the errors of the presented method, while the red graph represents the VTD results. Pixel errors are on the vertical axis, frame numbers on the horizontal axis.

Comparison to VTD on standard data

We also compared our method to VTD using higher-quality images that Kwon and Lee used themselves. It turned out that our method mostly had similar results, sometimes even outperforming VTD, although with a lower framerate. This is depicted in Figure 3.



(a) A sample image

(b) As above, blue represents the presented method and red the VTD.

Figure 3 A sample sequence of real-world data. Note that in this case, our method actually outperforms the VTD.

6 Conclusion

6.1 Summary and Contributions

We presented an extension to the framework Visual Tracking Decomposition [6] that improves some of its shortcomings and enables tracking in low-quality images, especially mm-wave data from full-body scanners. We proposed a new region descriptor that is based on features introduced in ultrasound texture discrimination [10] and applied it successfully to mm-wave data and color images. Our method uses the Förstner distance [4] for region comparison, improves the multi-template observation model of VTD using a forget factor [12] and a more sophisticated weighting scheme, incorporates a better motion model and sampling method [8] for particle filtering, and replaces the model mean with the template closest to the model median.

In comparison with VTD our method is superior on mm-wave data (low-quality and noisy grayscale images), while it has similar results for data with color and strong edges.

6.2 Future Work

The feature set we have chosen consists of 21 features both from the spatial and the frequency domain. A comparative study is necessary to evaluate the usefulness of each feature in order to reduce the number of features to a minimum while keeping tracking quality. Furthermore, feature computation is quite expensive since most computations need to be done locally and more than 16000 Fourier transforms are necessary for each frame. An optimized and parallelized way of feature computation will improve frame rates significantly, although real-time results are not expected to be feasible. Finally, we would like to extend our method to track several objects simultaneously and also incorporate color features to improve tracking results in colored image data.

Since the presented tracking algorithm works well on mm-wave images, a very challenging kind of data that to our knowledge nobody ever dealt with, we definitely expect it to be well-suited for similar applications. Therefore, we plan to test and tune our method to a variety of different imaging techniques, such as infrared imaging and especially ultrasound data. We adapted the texture features from the ultrasound imaging domain, so our region descriptor should be capable of tracking objects or anatomical structures.

7 Acknowledgements

We would like to thank Smith Heimann GmbH for providing us with mm-wave data and discussing our research. We also thank Junseok Kwon and Kyoung Mu Lee for providing us with the source code of their VTD framework.

This work was partially funded by the German Research Foundation's International Research Training Group (IRTG) "Visualization of Large and Unstructured Data Sets, Applications in Geospatial Planning, Modeling and Engineering" and the German Research Center for Artificial Intelligence (DFKI GmbH).

References

1 M. Sanjeev Arulampalam, Simon Maskell, and Neil Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2002.

2 Fabien Campillo, Rivo Rakotozafy, and Vivien Rossi. Parallel and interacting markov chain monte carlo algorithm. *Mathematics and Computers in Simulation*, 79(12):3424–3433, 2009.

3 Alexandre d'Aspremont, Laurent El Ghaoui, Michael I. Jordan, and Gert R. G. Lanckriet. A direct formulation for sparse pca using semidefinite programming. *SIAM Rev.*, 49:434–448, July 2007.

4 W Förstner and B Moonen. A metric for covariance matrices. *Ratio*, 66:113–128, 1999.

5 W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

6 Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, 2010.

7 H. Ling and K. Okada. Diffusion distance for histogram comparison. *International Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, 2006.

8 Liang Faming Liu, Jun S. and Wing Hung Wong. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, March 2000.

9 Song-Sheng Liu and M.E. Jernigan. Texture analysis and discrimination in additive noise. *Comput. Vision Graph. Image Process.*, 49:52–67, January 1990.

10 Russell Muzzolini, Yee-Hong Yang, and Roger Pierson. Texture characterization using robust statistics. *Pattern Recognition*, 27(1):119 – 134, 1994.

11 G.O. Roberts, A. Gelman, and W.R. Gilks. Weak convergence and optimal scaling of random walk metropolis algorithms. *Ann. Appl. Probab.*, 7(1):110–120, 1997.

12 David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. 2008.

13 Peter Salz. Automated tracking of threats in imagery from mm-wave scanners. Master's thesis, University of Kaiserslautern, Germany, 2011.

14 Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In *Proc. 9th European Conf. on Computer Vision*, pages 589–600, 2006.

15 Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, 1:511–518, 2001.

16 Yang Wang and Qiang Wu. Sparse pca by iterative elimination algorithm. *Accepted in Advances in Computational Math.*, 2010.

Evaluation of Mobile Phones for Large Display Interaction

Jens Bauer, Sebastian Thelen, and Achim Ebert

Computer Graphics and HCI Group
University of Kaiserslautern, Germany
{j_bauer,thelen,ebert}@cs.uni-kl.de

Abstract

Large displays have become more and more common in the last few years. While interaction with these displays can be conducted using standard methods such as computer mouse and keyboard, this approach causes issues in multi-user environments, where the various conditions for providing multiple keyboards and mice, together with the facilities to employ them, cannot be met. To solve this problem, interaction using mobile phones was proposed by several authors. Previous solutions were specialized interaction metaphors only for certain applications. To gain more insight into general interaction patterns realizable with smart phones, we created a set of general test cases using a well-known taxonomy for interactions. These test cases were then evaluated in a user study, comparing smart phone usage against the traditional keyboard/mouse-combination. Results (time and user satisfaction) show strengths and weaknesses when using the new interaction with the smart phone. With further evaluations we draw conclusions on how to improve large display interaction using smart phones in general.

1998 ACM Subject Classification I.3.6 Interaction Techniques, H.5.2 Interaction styles, H.5.2 Input devices and strategies

Keywords and phrases User Study, Large Display Interaction

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.103

1 Introduction

The idea to use mobile devices, especially mobile (smart) phones, to control Large Displays is not new. As both Large Displays and mobile phones (and especially smart phones) have become more common in the last few years, this idea is feasible. Large Displays is an umbrella term for various setups. These include tiled display walls, projection screens (with one or multiple projectors), large Liquid Crystal Displays (LCD), Powerwalls, and more. All Large Displays share advantages and issues, as noted for example in [5]. The main benefit is the increased screen real estate, allowing to display more information at once and enabling users to employ their spatial memory for efficient navigation. The authors further identified the following drawbacks:

1. Keeping track of the mouse position
2. It becomes more time consuming to access distant items on the screen
3. Windows may appear in unexpected places, where the user is not focusing at the moment
4. The number of simultaneous tasks a user carries out is probable to increase. This in turn calls for better task management.
5. Problems with the configuration of the different screens, especially with their position relative to each other, may occur.
6. Failure to leverage the periphery of the combined display.

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 103–112

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


These problems can be tackled in various ways (as presented in the next section). Especially the first four problems may be solvable by moving away from 'Windows, Icons, Menus, Pointer' (WIMP) environments and introducing new interaction metaphors. Smart Phones are complex devices with the capabilities of serving not only as an input, but also as an output device. Exploiting these characteristics can help to solve these problems.

1.1 Related Work

1.1.1 Large Displays

Of course, the first three problems only arise in 'Windows, Icons, Menus, Pointer' (WIMP) environments. As these systems are the most common ones used today, they still deserve attention. Robertson et al. [11] came to a similar list of issues, adding the bezel-occlusion problem to the list. To solve the first problem they proposed the mouse-trail and the display of a short animation when the user presses a key. Both functions are implemented in current version of MS Windows. For the distant access problem the paper proposes a number of solutions, like scrolling a single whole screen or selecting a target window using a 2D-ray from the current mouse position. So far the solutions have not been integrated into any modern operating system. Another solution is to use a trackpad with both relative and absolute positioning of the cursor [10]. A solution for problem 3 is not directly proposed. Instead the authors of Robertson et al. [11] propose to work around this problem by using the techniques for problem 2 and the following methods suggested to solve the multi-tasking problem. To do that the authors want to allow grouping of windows and applications both on the desktop and the (Windows) task bar. Wallace et al. [15] show a method to automatically configure tiled display systems, addressing the fifth issue in the list above. By displaying special patterns and evaluating the result through a camera, the authors calculate display distortion an the relative positions of the displays. This information is then used to correct the actual image displayed on the individual displays. This leaves the last problem, which cannot be solved by system designers. Instead this is more a missed opportunity to utilize the capabilities of Large displays. Application designers have to keep this in mind when creating programs for large display environments. One possibility of using this periphery is the focus+context screen [2]. The bezels on tiled displays can be seen as a help for the user to organize his large desktop or as proposed in the Tiled++ approach [7] their effect can be compensated by projecting the missing content onto the bezels.

1.1.2 Mobile Phones

Large displays are often used in collaborative scenarios, where naturally multiple users will want to interact with the application(s). While this can be facilitated by the traditional approach of keyboard and mouse, this does not scale well to the number of users involved. Also mouse and keyboard need to be mounted on some surface to be used without difficulties which imposes additional constraints on the environment of the large display and also the users (who is not able to move around freely). Mobile phones as ubiquitous input devices [1] have the ability to control a large display and may be able to replace mouse and keyboard altogether. Since modern (smart) phones feature multi-touch displays, cameras, Global Positioning System (GPS), accelerometers, compass, connectivity via 3G, WiFi and bluetooth and other capabilities, smart phones are subject of much work done already. This includes research for very special applications, like the 3D Human Brain Atlas [14]. Other work is focusing on data exchange [6][12] or the control of a traditional pointer using the phone [9]. On a more abstract level the design space of mobile phones were researched by

Ballagas et al. [1]. They used the taxonomy introduced by Foley et. al [8] which can still be used today.

1.2 Taxonomy of Foley et al.

This taxonomy uses six categories of input subtasks which are composited to generate actual input tasks. It should be noted, that the different categories are named not using the continuous verb form, but instead the infinitive form (e.g., Position instead of Positioning).

1. *Position* All subtasks asking the user to set the position of an object (including the position of the user) or to move an object around to a new position
2. *Orient* A task similar to position, the user is supposed to set the orientation or rotation of an object
3. *Select* Lets the user pick an object (e.g., an option from a list, select an object in 3D-space, etc)
4. *Ink* (also called *Path* in [1]) This is in effect a combination of multiple Position and/or Orient tasks and is used to define the path of an object and its orientation on this path. As *Ink* has some slightly different requirements it is a task for itself
5. *Quantify* setting a numerical value or some option derived of a numerical value (e.g., setting a volume out of the options "quiet", "normal", "loud" with each option corresponds to a db-value)
6. *Text* entering plain text, text markups are covered by other tasks

Text is already evaluated in detail in the work of Butt and Cockburn [4] and also in the work of Silfverberg et al. [13]. The subtasks *Ink* and *Quantify* can be expressed using the remaining three subtasks (when accepting some limitations), *Position*, *Orient* and *Select* are the most interesting subtasks.

These subtasks are used to further distinguish the tasks given in the test scenarios presented in the next section.

2 Evaluation Basis

This paper focuses on the work published in [3]. For that paper a user study was conducted with four different case studies on a 3x3 tiled high-resolution display. 17 test candidates of various ages and different levels of user experience were asked to complete all test scenarios using a smart phone (HTC Touch Diamond 2) and also using the traditional keyboard or mouse or a combination of both.

2.1 Description of Test Cases

A short summary of the test scenarios follows. For a more detailed description, see [3].

2.1.1 Stacking Cubes

In this three-dimensional test scenario users were asked to stack three cubes one on another in a simple physics driven environment. This involves 3 degrees-of-freedom (DOF) movement in space, without regard to rotation. This could be accomplished by either using a keyboard (using two keys per DOF), a mouse (normal position tracking + mouse wheel) or a smart phone (using the touch screen to perform movement in two dimensions and tilting the phone itself to perform movement in another two dimensions, resulting in 2 ways to control x-direction) as input device. It should be noted that the phone is the only device providing

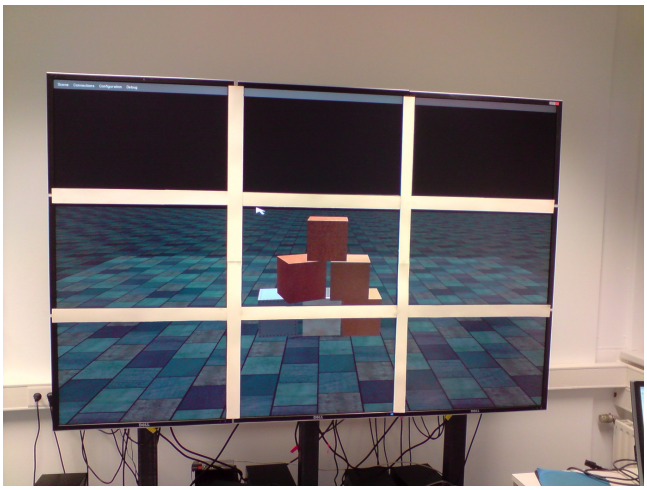


Figure 1 Stacking Cubes, the user has to stack up three cubes in a physics-enabled environment.

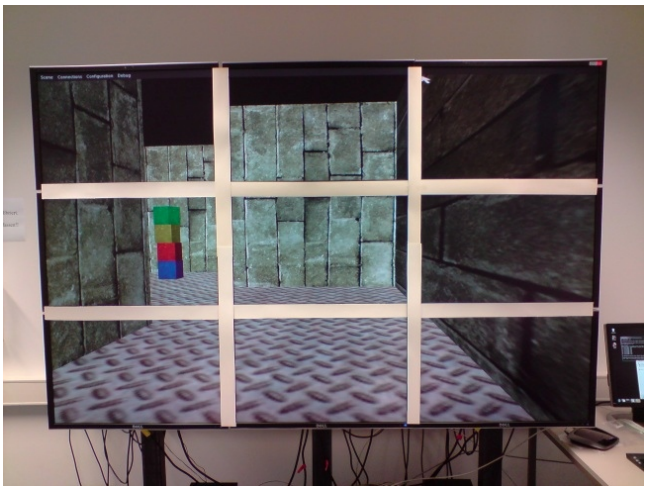


Figure 2 Maze, colored cubes can be found at one spot in the maze.

continuous movement in all dimensions. This scenario uses the subtasks of *Position* and *Select*. Figure 1 shows how the scene looks like in the study.

2.1.2 Maze

The second scenario features the first-person view of a simple maze (as can be seen in Figure 2). The test candidates have to navigate through the maze (with the help of a map) to a certain spot where they can pick up a colored cube by just touching it. Afterwards, the goal is to backtrack the way and to drop the cube into a bin outside the maze. The most commonly used input method for this case is probably the combination of keyboard and mouse, known to a wide audience of first-person computer games. Furthermore, control is possible using only keyboard (arrow-keys) or only the mouse (mouse position orients the view and holding the mouse button accelerates). Using the smart phone as it was a joystick (tilting the phone in the desired direction) was the last input method implemented. This scenario consists of the subtasks of *Position* and *Orient*.

Figure 3 City Map Annotation, the small flags from the upper left corner can be dragged-and-dropped onto the map.

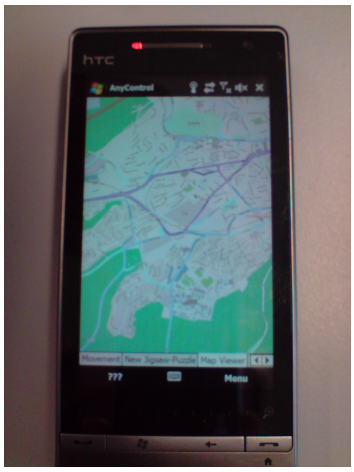


Figure 4 The city map on the phone.

2.1.3 City Map Annotation

On a local city map (Figure 3), the candidates are able to place flag markers. These markers contain some more information: A descriptive text and a picture. The flags can be placed by selecting the flag in the upper left corner in the map view and dragging it to the desired position. Then a small popup-window will appear (the size can be seen in Figure 3 in the lower middle screen), where the user can enter the description and select a picture to be displayed. The smart phone displays a smaller version of the same map, that can be panned and zoomed individually without affecting the main view on the large display (Figure 4). Unfortunately, the smart phone did not support multi-touch interaction, so zooming had to be done by using a menu. With the same menu a selection mode can be activated. Then the next tap on the map will place a flag marker and open a new view on the phone where the user can enter description text and select or even take a photo. As none of these actions will directly affect the main view (besides adding the marker on the map at some point), multiple users can perform this task at once without interfering with each other). In this scenario *Position*, *Select* and *Text* was used.

2.1.4 Jigsaw-Puzzle

One of three simple 5x5 tile jigsaw-puzzle had to be solved in this test case. The puzzle was shown on the tiled wall (Figure 5) and simultaneously a live copy of it on the smart phone (Figure 6). On the phone the tiles can be dragged around with a simple touch and drag. If necessary the user also can zoom in or out of the puzzle. Additionally, each test candidate had to solve another puzzle (for a total of three) with the keyboard and the mouse. *Postion* and *Select* were the subtasks needed for the jigsaw-puzzle.

2.2 Evaluation Setup

The evaluation was done with the already mentioned 17 participants of various ages and levels of computer experience. They were asked to carry out the tasks described above in random order using all available input methods, again in random order. In each case the

Figure 5 Jigsaw-Puzzle, the tiles have to be brought in the correct positions.

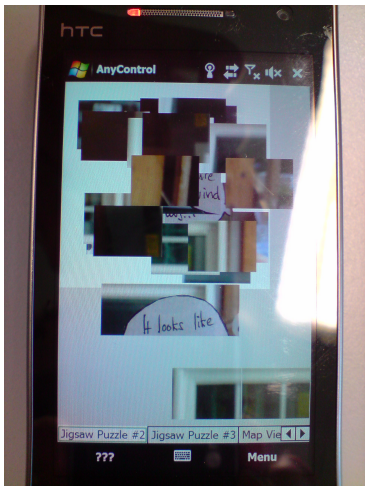


Figure 6 Jigsaw-Puzzle on Phone.

candidate had a short opportunity to get used to the input method, to a point where he or she was able to perform the task. Unfortunately not enough time was available for the candidates get a higher level of proficiency. This probably had a negative effect on the score of the smart phone especially, as no user has used a phone for large display interaction before, but each one had at least a bit of training using a computer mouse and a keyboard. For a quantitative analysis the time to complete each task with each input method was measured. To get comparable times of all candidates, the times were normalized by dividing each time by the accumulated time of all input methods of the observed task. This yields times on a scale from 0 to 1. Another quantitative measure was a grade given by every participant for each input method per test case. Possible grades range from 1 (best) to 6 (worst). To get some qualitative results, each candidate was also asked for his/her comments on the smart phone control and also for improvement proposals. The rest of the paper will now describe the results formally and draw conclusions.

3 Formal Evaluation

The measured times (total and normalized) are on a ratio scale, grades are ordinal. A set of popular descriptive statistics about the normalized times can be found in Table 1. Using a one-way analysis of variance (ANOVA) for each test scenario, the mean times can be shown to be statistically significant different on a confidence level of 5%. The basic requirements of the ANOVA, the mean times being normal distributed and all mean times having the same variance, are assumed to be met. For timed tasks normal distribution can safely be assumed and to be sure about the variances a Levene-Test has been performed for each test scenario. Unfortunately the Levene-Test did not confirm (on a 1% level) that the variances in scene 2 are equal, but as the ANOVA is known to be a very robust test, it was done anyway, but this fact has to be kept in mind. The ANOVA itself showed that the mean-time differences in the input methods are statistically significant on a 5% level (with F-values of 7.109, 21.733, 43.898 and 67.096 for scenes one to four, respectively). For scene 2, the F-value can show significant differences in means for confidence intervals below even 0.1%. This fact and a Welch-Test (also testing for differences in mean values, but also valid for non-equal variances)

Table 1 Descriptive Statistics of Normalized Times. 1st Q and 3rd Q stand for 1st and 3rd quartile, CV is the coefficient of variation and IQR is the interquartile range.

		Min	1st Q	Median	3rd Q	Max	Mean	Std Dev	CV	Range	IQR	Curtosis	Skewnes
Scene 1	Keyboard	0.104	0.172	0.261	0.307	0.570	0.265	0.125	0.471	0.467	0.135	0.346	0.797
	Mouse	0.079	0.190	0.316	0.383	0.576	0.301	0.141	0.469	0.498	0.192	-0.840	0.327
	Phone	0.230	0.348	0.401	0.512	0.740	0.435	0.136	0.314	0.510	0.164	-0.170	0.548
Scene 2	Keyboard	0.145	0.169	0.208	0.228	0.264	0.204	0.035	0.174	0.120	0.059	-1.087	-0.038
	KB + M	0.119	0.177	0.196	0.216	0.328	0.204	0.051	0.248	0.209	0.039	1.591	0.951
	Mouse	0.157	0.226	0.245	0.275	0.372	0.250	0.048	0.194	0.215	0.049	1.531	0.478
	Phone	0.211	0.281	0.335	0.412	0.483	0.343	0.076	0.221	0.271	0.131	-1.112	0.124
Scn 3	Keyboard	0.235	0.295	0.385	0.476	0.543	0.380	0.102	0.269	0.308	0.181	-1.423	-0.062
	Phone	0.457	0.524	0.615	0.705	0.765	0.620	0.102	0.165	0.308	0.181	-1.423	0.062
Scene 4	Keyboard	0.563	0.650	0.697	0.761	0.829	0.701	0.073	0.104	0.267	0.111	-0.654	-0.178
	Mouse	0.171	0.239	0.303	0.350	0.437	0.299	0.073	0.245	0.267	0.111	-0.654	0.178
	Phone	0.263	0.405	0.476	0.558	0.771	0.483	0.128	0.265	0.508	0.153	0.295	0.253

also showing differences in the mean normalized times leads us to the conclusion, that the ANOVA yield correct results even for scene 2. To get deeper insight into which mean times actually differ, a Tukey-HSD test was conducted. This test shows the pair-wise (in-)difference between the normalized mean times. The results are shown in tables 2-4.

The most important fact from this results is that the mean times of the smart phone users always differ significantly from all others. Knowing this, we can safely interpret the normalized times to get an overview of the test results. Since the grades are on an ordinal scale, no ANOVA was conducted for them.

A (Pearson) correlation test shows significant (again on a 5% level) correlation in the different times taken to solve each scenario using the smart phone. An exception for this is scene 3, the *Map Annotation*. A possible explanation for this may be the fact, that in this test case the smart phone's built-in menu had to be used a lot. This was difficult for most users, as nobody had any experience with a Windows 6 smart phone. The correlation for the other test cases show, that there is some kind of taste or distaste for the phone control. This fact is hardly surprising, but still notable. It also hints, that the usage of the native menu of the phone is a very unintuitive way of providing interaction possibilities. This was also mentioned by a few of the test candidates.

Table 2 Descriptive Statistics of Grades. CV is the coefficient of variation.

		Mode	Median	Mean	Std Dev	CV
Scene 1	Keyboard	2	2	2.1	0.96	0.45
	Mouse	3	3	2.5	1.19	0.48
	Phone	4	3	3.3	1.07	0.33
Scene 2	Keyboard	2	2	2.35	0.836	0.36
	KB + M	2	2	2.12	0.963	0.45
	Mouse	3	3	2.88	1.1315	0.39
	Phone	3	3	3.19	0.9656	0.3
Scn 3	Keyboard	1	1	1.4	0.48	0.35
	Phone	2	2	2.3	0.89	0.39
Scene 4	Keyboard	4	4	4	1.6202	0.41
	Mouse	1	1.5	1.69	0.8455	0.5
	Phone	2	2	2.5	1	0.4

Table 3 Tukey-HSD Results for scenario 1.

	KB	CM	SP
KB			✓
CM			✓
SP	✓	✓	

Table 4 Tukey-HSD Results for scenario 2.

	KB	KM	CM	SP
KB				✓
KM				✓
CM				✓
SP	✓	✓	✓	

The ARC-Pad cursor control method [10] was also implemented. This control turns the phone into a trackpad, where a tap causes the mouse cursor to jump to the position on the screen relative to the position the tap happened on the phone, e.g., A one finger tap in the center of the screen lets the mouse cursor jump to the center of the screen. A swipe on the touchscreen moves the mouse cursor normally. The goal of this control is to have fast cursor positioning together with the accuracy of a touchpad control. While the idea sounded very feasible, early tests showed very bad results. Therefore, a formal evaluation of this interaction pattern was not conducted. The main cause of the ARC-Pads issues was the inaccuracy when selecting a cursor position by tap. The resulting corrections of the position took too much time to be comfortable for the users.

The test candidates were also asked for an informal description of their experience using the new input mechanisms. Many of them stated that they had little to no experience with the control of the 3D scenarios. They also pointed out, that the control was a big lagging (Probably caused by the slow CPU on the smart phone). Virtually all users liked the 2D scenarios, where direct interaction was enabled. This was a very intuitive way of solving the tasks at hand. The issues identified here were almost all about the absence of multi-touch and the need to use the clunky system menu to activate selection mode in the Map Annotation Scenario.

4 Conclusion

The smart phone did not get the best grades or the best times. What still makes it a viable input option is the fact that it solves the problem of scaling the interaction against the number of users. Using the improvement suggestions made by the test candidates will further improve the interaction metaphors used in this first study. Together with newer and more capable hardware the smart phone seems to get on par with the other input methods. Unfortunately, there is no formal study at this time to show this, as this is still work in progress. Informal evaluation including the comments made by the test candidates shows, that all candidates liked the interaction metaphors and were also able to understand them. The main complaint was about insufficient hardware capabilities, especially sketchy

Table 5 Tukey-HSD Results for scenario 4.

	KB	CM	SP
KB		✓	✓
CM	✓		✓
SP	✓	✓	

KB = Keyboard, CM = Computer Mouse, KM = Keyboard + Mouse SP = Smart Phone; A checkmark denotes a significant difference in mean times.

accelerometers and missing multi-touch capabilities. This is something to include in further approaches. Informal studies made using the most recent Apple iPhone show much better results. Unfortunately at the time of writing no formal evaluation is was available to be included here. The better accelerometers, combined with filtering of the acceleration sample data provided by the sensors yield very stable results and greatly improve user experience. If this translates into faster solutions of the given tasks is to show in a formal study similar to the one presented here. For the 3D scenarios, *Stacking Cubes* and *Maze* users were mainly burdened with the special hand posture needed to activate the accelerometer on the HTC Touch Diamond 2. Using the touch screen to do the activation causes better results. In *Map Annotation* and *Jigsaw-Puzzle* using the touch screen of the smart phone for direct touch interaction was preferred by most users. What made the tasks in both cases a bit more problematic, was the inclusion of the system menu, as already stated in the last section. For those 2D tasks the multi-touch capabilities of modern smart phones will greatly improve performance of these tasks. Usage of multiple finger to move multiple jigsaw-puzzle tiles and using pinching gestures for map navigation allows for a more intuitive interface and greater user satisfaction.

When grouping the scenarios using the Foley-Taxonomy, *Select* was a task that can be done with the smart phone most easily. The smart phone provides direct touch interaction for selection tasks, while the mouse only provides indirect methods. Using the keyboard either means finding the correct key for the regarded object or instead cycling through all available objects until the object to be selected appears. *Orient* was not performed so well with the smart phone. The main cause may be the use of the accelerometers for this. Maybe better accelerometers, more experience with this kind of interaction or a new metaphor will solve this, but further research is needed. For *Position* the results seem to be mixed at first. But when looking at the different techniques used, one can see, that position with the accelerometer did not perform well, for the reasons already stated. *Position* with the touch screen worked very well and got a high level of user satisfaction.

Future improvements are, as already described, to use better hardware. But besides that, small improvements on the software side can also be done. As a general note, it is very advisable to use a communication protocol with a low memory footprint with smart phones. This can help to reduce lag in the connection between Large Display and phone. While this increases development time, lag decreases the user experience by a large amount. When using the accelerometers, it is also recommended to allow user configurable settings for home positions, dead zones and sensitivity. As seen in the Map Annotation test scenario, menus should be avoided if possible, as tends to break the intuitively of the interface. If really needed menus should only contain the most necessary items and be large enough to be selected with ease.

From the view of an interaction designer, the results show that touch screen input (and output) can be very well used for large display interaction. Even older phones have a touch screen good enough for this task. Employing multi-touch gestures enhances the user experience, but is not needed for basic functionalities. Of course it is necessary to keep an eye on the smaller screen real estate on the phone, but especially for 2D tasks this is the preferred way to go. Using the accelerometer for 3D interaction is a good approach per se, but in reality requires some practice on the user's side. This is therefore only feasible if the same interaction pattern can be reused many times.

References

1 R. Ballagas, J. Borchers, M. Rohs, and J.G. Sheridan. The Smart Phone: A Ubiquitous Input Device. *IEEE Pervasive Computing*, 5(1):70–77, January 2006.

2 Patrick Baudisch and Nathaniel Good. Focus Plus Context Screens: Combining Display Technology With Visualization Techniques. *interface software and technology*, 3(2), 2001.

3 Jens Bauer, Sebastian Thelen, and Achim Ebert. Using Smart Phones for Large-Display Interaction. In *2nd International Conference on User Science and Engineering (I-USEr)*, 2011.

4 Lee Butts and Andy Cockburn. An Evaluation of Mobile Phone Text Input Methods. *Australian Computer Science Communications*, 24(4):55–59, 2002.

5 Mary Czerwinski, George Robertson, and Brian Meyers. Large Display Research Overview. *CHI 06 extended*, page 69, 2006.

6 Raimund Dachselt and Robert Buchholz. Throw and Tilt – Seamless Interaction Across Devices Using Mobile Phone Gestures. *Proc. MEIS 2008*, pages 272–278, 2008.

7 Achim Ebert, Sebastian Thelen, P.S. Olech, Joerg Meyer, and Hans Hagen. Tiled++: An Enhanced Tiled Hi-res Display Wall. *Visualization and Computer Graphics, IEEE Transactions on*, 16(1):120–132, 2010.

8 James Foley, Victor Wallace, and Peggy Chan. Human factors of Computer Graphics Interaction Techniques. *IEEE Computer Graphics and Applications*, 4(11):13–48, 1984.

9 Antonio Haro, Koichi Mori, Tolga Capin, and Stephen Wilkinson. Mobile Camera-Based User Interaction. *Computer Vision in Human-Computer Interaction*, pages 79–89, 2005.

10 D.C. McCallum and P. Irani. Arc-pad: Absolute + Relative Cursor Positioning for Large Displays with a Mobile Touchscreen. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 153–156. ACM, 2009.

11 George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan. Large Display User Experience. *Computer Graphics and Applications, IEEE*, 25(4):44–51, 2005.

12 Khoovirajsingh Seewoonauth, Enrico Rukzio, Robert Hardy, and Paul Holleis. Touch & Connect and Touch & Select: Interacting with a Computer by Touching it with a Mobile Phone. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, page 36. ACM, 2009.

13 Miika Silfverberg, I. Scott MacKenzie, and Panu Korhonen. Predicting Text Entry Speed on Mobile Phones. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00*, 2(1):9–16, 2000.

14 Sebastian Thelen, Joerg Meyer, Achim Ebert, and Hans Hagen. A 3D Human Brain Atlas. *Modelling the Physiological Human*, pages 173–186, 2009.

15 Grant Wallace, Han Chen, and Kai Li. Automatic Alignment of Tiled Displays for a Desktop Environment. *Journal of Software*, 15(12):1786, 2005.

Controlling In-Vehicle Systems with a Commercial EEG Headset: Performance and Cognitive Load

Daniel Cernea[1,2], Peter-Scott Olech[1], Achim Ebert[1], and Andreas Kerren[2]

1 **Computer Graphics and HCI Group**
University of Kaiserslautern, Germany
{cernea,olech,ebert}@cs.uni-kl.de

2 **ISOVIS Group, Computer Science Department**
Linnaeus University, Växjö, Sweden
andreas.kerren@lnu.se

Abstract

Humans have dreamed for centuries to control their surroundings solely by the power of their minds. These aspirations have been captured by multiple science fiction creations, such as the Neuromancer novel by William Gibson or the Brainstorm cinematic movie, to name just a few. Nowadays, these dreams are slowly becoming reality due to a variety of brain-computer interfaces (BCI) that detect neural activation patterns and support the control of devices by brain signals.

An important field in which BCIs are being successfully integrated is the interaction with vehicular systems. In this paper, we evaluate the performance of BCIs, more specifically a commercial electroencephalographic (EEG) headset in combination with vehicle dashboard systems, and highlight the advantages and limitations of this approach. Further, we investigate the cognitive load that drivers experience when interacting with secondary in-vehicle devices via touch controls or a BCI headset. As in-vehicle systems are increasingly versatile and complex, it becomes vital to capture the level of distraction and errors that controlling these secondary systems might introduce to the primary driving process. Our results suggest that the control with the EEG headset introduces less distraction to the driver, probably as it allows the eyes of the driver to remain focused on the road. Still, the control of the vehicle dashboard by EEG is efficient only for a limited number of functions, after which increasing the number of in-vehicle controls amplifies the detection of false commands.

1998 ACM Subject Classification H.5.2 User Interfaces (D.2.2, H.1.2, I.3.6): Evaluation/ methodology, Input devices and strategies (e.g., mouse, touchscreen)

Keywords and phrases Brain-computer interface, EEG neuroheadset, EEG control, driver cognitive workload, in-vehicle systems.

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.113

1 Introduction

The desire of humans to gain the ability to control everyday actions just by using the power of their mind is old. These ideas became very popular in science fiction literature as well as in movies, like the popular Star Wars series. But in fact, research tries to exploit brain-computer interfaces (BCI) to provide support for motionless interaction. For example, Touch Bionics is offering brain controlled prosthetic arms tailored for amputees, aiming to increase the quality of life for handicapped people. In our related work section we will reveal a more detailed view on research related to the field of BCI.

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 113–122

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Figure 1 Image of a participant while interacting with the driving simulator and the EEG-controlled dashboard.

The focus of our current work however is to use a commercial electroencephalographic (EEG) headset in order to support car drivers in controlling secondary dashboard functions of a vehicle. In our case the Emotiv EPOC headset is used. First and foremost we want to emphasize that the user only controls secondary function of the car by the EEG headset. The testing and evaluation of our approach was done in a safe environment (laboratory experiment and evaluation) since we used a driving simulator and implemented secondary function of the dashboard. Our goal is to find out if the test candidates drive more securely when being able to focus on driving the vehicle itself and not having to manually operate secondary (non-driving essential) functions.

In the following sections we will highlight related work, as well as give a brief description of our setup, the evaluation and our test results.

2 Related Work

One of the most common and well-known fields of application for BCI is to control devices by brain waves. As such, research has intensively focused on measuring and increasing the performance of BCI-based consciously operated systems. In this setting, many medical solutions have been explored that would enable patients with physical disabilities to live a normal life. For example, controlling a wheelchair with the help of EEG devices has been a topic in several research papers. Leeb et al. [8] demonstrated that a tetraplegic patient was able to control the movement of a wheelchair in a virtual environment. In the work of Iturrate

et al. [5] a BCI and an autonomous navigation system is used to control a wheelchair. The RIKEN and Toyota Motor Corporation also did research in the field of utilizing brain waves to control a wheelchair in real-time [18]. Stamps and Hamam [12] describe how low-cost BCI devices can be utilized to control prosthetic devices.

Furthermore, the utilization of a commercial headset to control a robotic arm is proposed in the work of Ranky and Adamovich [11]. According to them, after a training period users are able to get used to the control functions and improve the overall performance. Vourvopoulos and Liarokapis compare two commercial EEG headsets in order to control a Lego NXT robot [13].

Controlling vehicles by using BCI also has been subject of research. Zhao et al. [15] did use motor imagery (MI) to control EEG as well as a car in a Virtual Reality (VR) environment. Zhang et al. used a BCI to control an unmanned vehicle [14]. With BrainDriver, a project developed by FU Berlin, Germany, a commercial EEG headset is used along with Laser Range Finder (LRF) and Global Position System (GPS) data to control a car [16]. A quite interesting approach is featured in "Prototype This: Mind Controlled Car" by Discovery Channel [17]. Here, a BCI is used to measure the driver's rage in order to prevent road rage by slowing down the car.

Brain-computer interfaces also have been used to gain knowledge about car drivers to understand what emotional states a driver is experiencing when driving a vehicle. Gugler et al. [20] did monitor attention processes during a monotonous car driving simulation with EEG. The work of Putze et al. [10] and van den Haak et al. [4] utilize BCI to gain knowledge about cognitive workload of drivers during multitasking or under stress.

Osswald and Tscheligi [9] explore the driver distraction when performing secondary tasks during driving. Kyung et al. [7] introduce a wearable in-vehicle device, providing the driver with relevant information and also obtaining the physiological data of the driver.

Relevant to our work is the research of Anderson et al. [1] comparing visualization techniques in terms of cognitive workload by the usage of EEG devices. Cernea et al. [2, 3] also detected facial expressions as well as emotional states during various tasks by using a commercial EEG headset.

3 In-Vehicle Secondary Control Tasks with BCI

Using the previously presented projects as a starting point, our research aims at highlighting the performance of an EEG-based portable BCI headset when used to control in-vehicle secondary, non-vital systems. Additionally, by inspecting the traffic errors of the drivers in multiple circumstances, we hope to capture the influence of such a BCI system on the drivers' cognitive workload, thus further exploring the potential of these devices as in-vehicle interaction systems.

3.1 Design

Driving the vehicle is only one of the many tasks drivers need to focus on. As such, a BCI-powered interaction method could relieve the cognitive workload of the driver and reduce the level of distractions introduced by manually controlling multiple in-vehicle systems. But does EEG-based control have the capacity to reliably execute these commands? To inspect this, we conducted a preliminary study that investigates the performance and accuracy of EEG-based control of in-vehicle devices as well as highlights the error rates for the common tasks drivers have to execute in real-life driving scenarios.

During the experiment, the participants were involved in executing a set of activities that are always or at least sometimes present in everyday driving. The activities were grouped in the following three categories:

- *driving task (highest priority)* – firstly, every driver has to be in control of the vehicle and obey the traffic rules
- *search task (high priority)* – drivers are sometimes involved in search-and-find tasks in the environment surrounding the vehicle (e.g., finding a parking lot, finding a store, etc.)
- *control task (low priority)* – drivers have to interact with the dashboard of the vehicle (in our case via BCI) to activate various devices and vehicle subsystems (e.g., lights, radio, GPS, etc.)

Following this structure, participants were asked to control a vehicle in a simulated environment and obey the traffic rules. Additionally, each driver took part in a set of five search tests, each with duration of five minutes, aimed at simulating a common search scenario that drivers are faced with on a daily basis. Specifically, the tests had the following corresponding search assignments that the drivers had to execute: "Drive around the city, obey the traffic rules, and count the number of X's you see", where X would represent the type of objects that needed counting: Pre-Test – the number of red cars; Test 1 – the number of phone booths; Test 2 – the number of vans; Test 3 – the number of grocery stores; and Test 4 – the number of traffic lights. Instead of simply asking the drivers to search for a place, they were instructed to count them in order to allow all drivers to search for the full five minutes, and also in order to eliminate any competitive aspect from the sessions—e.g., if a driver is trying to find a landmark as fast as possible, he might disregard traffic rules or be unable to concentrate on controlling the BCI.

While the five tests were almost identical in terms of driving and search task, they were mainly introduced to capture the differences and particularities of controlling the in-vehicle systems. An initial pre-test was designed to offer a baseline measurement for the driving performance in the virtual environment and the distraction introduced by controlling the dashboard through normal touch-based interaction. For this, the users drove without the EEG headset and completed a search-and-find assignment during which the number of traffic violations was stored. Additionally, as the sessions were recorded, the answers reported by the users at the end of each search test were compared to the actual number of sought objects that appeared on the screen.

In the following tests (Test 1 to Test 4), the users drove the vehicle while wearing the EPOC headset and trying to control the dashboard with their minds. Each test increased the difficulty of the control task by adding commands or increasing their complexity (Figure 4). In order to better quantify the performance of the drivers, the dashboard controls were grouped into Boolean (turn lights on/off, turn heater on/off) and discrete operations (open the left window a bit, turn the volume up to the middle, etc.). In the initial BCI test (Test 1), the users only have to control two Boolean values (e.g. turn lights on/off, turn heater on/off). Test 2 already involved four Boolean controls, while the last two tests involved two Boolean and two discrete commands, respectively, four discrete commands.

For the control task, a supervisor periodically instructed the subjects involved in a test to execute a command on the dashboard (e.g. "turn on the radio"). While drivers are used to execute a command with touch (Test 0), in the case of EEG control the participants had a 10 seconds window to activate the functions via BCI (Test 1 to 4).

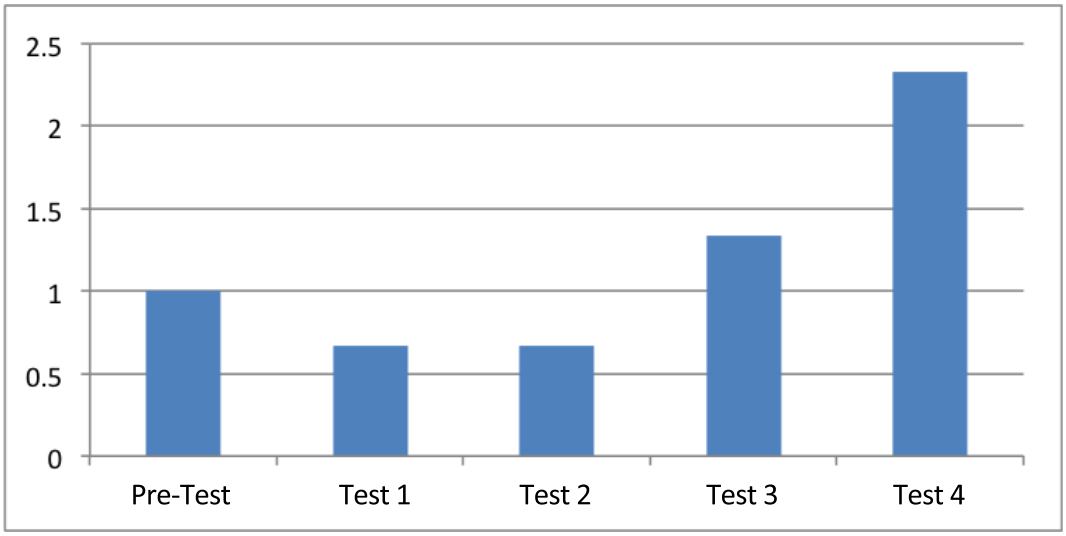


Figure 2 Traffic errors – Average number of traffic violations recorded by the driving simulator for every test.

3.2 Execution

The study was executed on a small sample of 12 participants in order to gather insights about the possibilities of EEG control and inspect the reduction of cognitive load on the drivers. The subjects had to complete an initial questionnaire that reflected an even distribution in terms of gender and age. Furthermore, all participants had a valid driver's license at the time of the experiment and were driving a vehicle on a regular basis.

After the initial questionnaire, the participants took part in an initial training session composed of two parts: getting familiar with the simulator and executing controls with the EPOC neuroheadset. For the simulator training, each user had 30 minutes to drive around and get used to the provided controls and interface. To support a realistic scenario, the controls that needed to be managed included steering, blinkers, direction of movement (forward or reverse), pedals, looking left and right, etc. Note that the virtual environment for the training was different from those used in the experiment, to ensure that no participant would have prior knowledge that would be relevant for the search tasks.

Furthermore, users had up to one hour to learn and train the basics of control with the Emotiv EPOC EEG device. The subjects trained to map various mental activation patterns, such as activating a command when concentrating at a concept or imagining a body movement. This was achieved by employing the EPOC's framework that detects and learns particular mental activations and classifies new activation patterns in existing categories. After training with the BCI, the participants were free to select a set of mental mappings they felt were most efficient and intuitive. These mental patterns were then used to train the EEG system for usage in the experiment.

In terms of the environment, a driving simulator was employed that is commonly used by people preparing for their driving test. The simulator supported traffic rules and was able to detect if a driver violates them. Other features of the simulator included: realistic controls, urban environments with visual and auditory cues, other virtual traffic participants, etc.

Additionally, we implemented a software dashboard to control a set of in-vehicle systems and subsystems. Some of the systems were controlled with Boolean commands (e.g. lights,

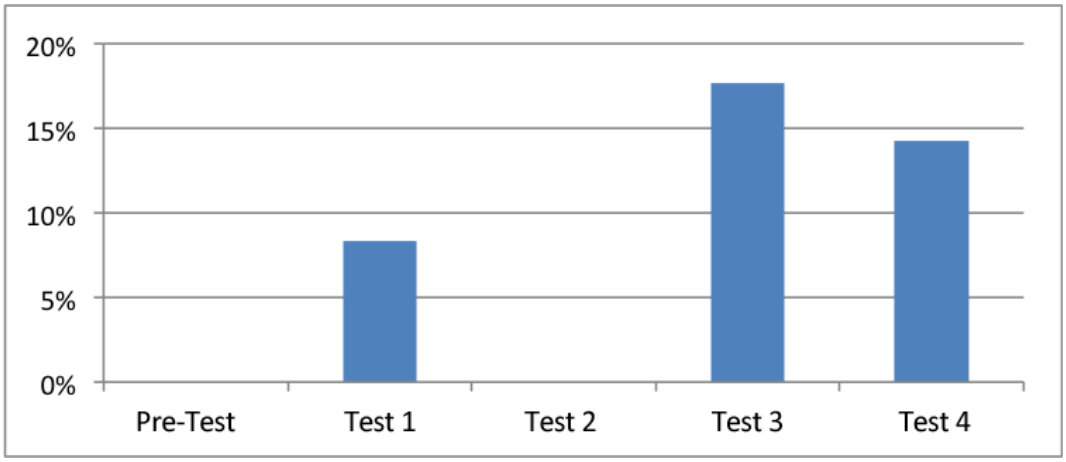


Figure 3 Search task errors – Average percentage of search objects not noticed by the participants during every test.

air-conditioning, seat heaters), while others required a more refined scale of control (radio with volume control, electric windows, etc.). The virtual dashboard could be controlled either through touch by pressing a set of keys or through the EPOC BCI. The dashboard was also conceived to give visual and auditory feedback to the driver about the execution of a certain command.

4 Results and Discussion

Figure 2 highlights the average distribution of traffic violations for each of the five search tasks. One can notice a slight decrease of the errors that the drivers made in traffic in the cases where the EPOC headset was employed for 2-4 simple interaction commands, compared to the touch-based manipulation of the controls. This reduction is relevant, especially when we note that the error rates between the initial training and the baseline search task (Pre-Test) are similar. Also, once discrete commands were considered, the traffic violations increased even passing the baseline level established in the Pre-Test, suggesting that the users had difficulties to execute the different BCI commands. This in turn can be a sign for an increased cognitive load that the execution of multiple complex BCI operations introduces.

The results for the primary search task are highlighted in Figure 3. As the number of search-and-find errors was overall reduced, the results cannot be the based of a thorough analysis. However, we hypothesize that the higher error levels detected for sessions involving discrete control could be again related to an increased cognitive workload.

Furthermore, an element that influences the search task results is the accuracy of the EEG control in each of the sessions (Figure 4). While in the initial tests the control error rates had satisfactory levels, it seems that the error margins increased with the complexity of every session, reaching values of over 40% for Test 4. Such a lack of accuracy can distract and create frustrations for the driver, resulting in higher error rates in the primary tasks. Moreover, we noticed that these control error rates could in some cases be improved with additional training. For the sake of completeness, we would like to mention that no control errors were recorded in the touch-based dashboard interaction (Pre-Test).

Looking at the other side of the coin, Figure 5 highlights the average number of falsely activated commands by the EEG device when the user received no instruction to execute a

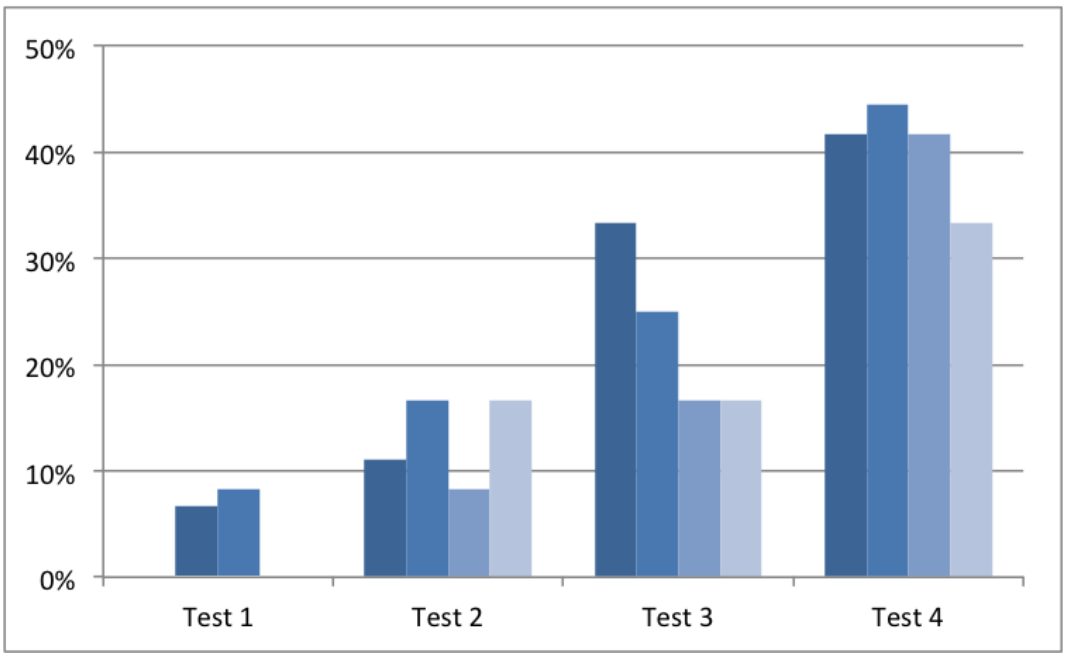


Figure 4 Control errors – Average percentage of not executed EEG-based dashboard commands for every test. The bars in each of the four tests represent: Test 1 – two Boolean commands; Test 2 – four Boolean commands; Test 3 – two Boolean and two discrete commands; Test 4 – four discrete commands.

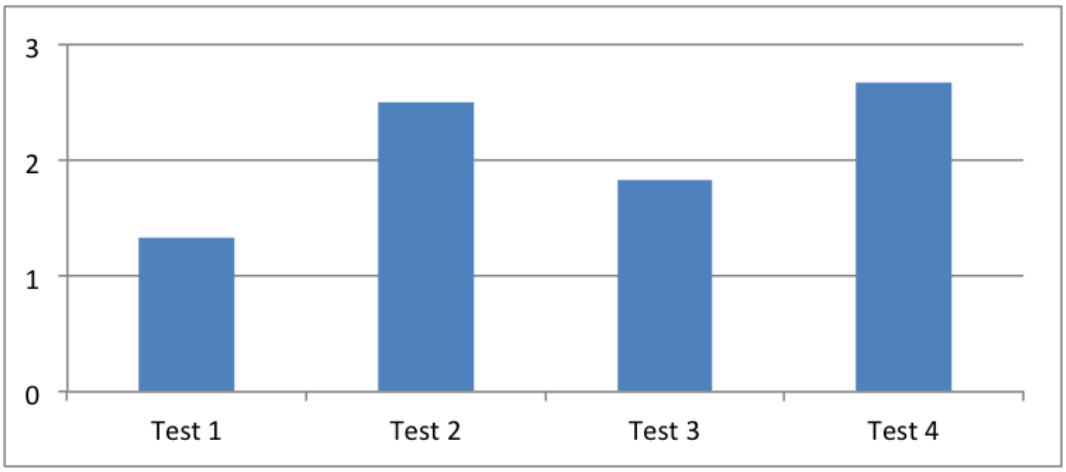


Figure 5 False positives – Average number of false command detections executed by the EEG interface for every test.

function. Considering a number of up to 10 instructions given by the supervisor in every session, the represented error rates can be considered low. Still, in this case, no particular pattern was distinguishable and further research of the topic is planned. Note that Figures 4 and 5 do not include the Pre-Test, as these Figures only reflect information about the EEG-based control, and the BCI headset was not employed in the initial touch-based test.

Besides the previously mentioned experiment, the participants were asked to complete a short questionnaire about their experience with the EEG headset and their impression about the use of BCIs in vehicles. Most users considered the neuroheadset as a viable alternative to touch-based controls, merely suggesting the inconvenient nature of employing non-dry sensors. Also, based on their experiences in the one-hour pre-experiment training, 83% of the participants decided to employ mental mappings that involved mostly imagined body movements (e.g., imagining to move a finger, the eyebrows, the shoulder, etc.) for executing commands.

When looking at in-vehicle usage, 66% of the participants expressed a positive attitude towards using a BCI device with such a functionality. This percentage has the potential to increase, as over 50% of the testers expressed their concern with the current level of accuracy, suggesting that they expect commands to be executed immediately and without repetition, even in the case of non-vital systems (e.g., "I don't want to try this [*turning on the radio*] three times").

5 Future Work

While simulating an environment is a relatively simple and inexpensive solution for many tests, in a next stage of this research we plan to evaluate the EEG headset control with an actual in-car dashboard. This would of course imply the measurement of the EEG headset's performance in control tasks, as well as the comparison of cognitive workload levels when simultaneously driving and accessing the dashboard functionality by touch and by BCI. For the recognition of the cognitive load level we plan to apply the widely accepted NASA Task Load Index (NASA-TLX) [19].

Focusing on a slightly different direction, we plan to investigate the detection of emotional states that the driver and other occupants of the vehicle experience. These states could then be influenced or compensated for by adapting the interior lighting and musical ambience inside the vehicle, making the EGG headset an integral part of a human-vehicle feedback system.

6 Conclusion

As in-vehicle systems become increasingly complex and versatile, it is vital to encourage the development of new interaction metaphors that are suitable for in-vehicle devices and do not representing a distraction to the driver. In this paper, we have investigated the performance and accuracy of EEG-based control of a simulated vehicle dashboard as well as captured traces of effects in-vehicle BCI usage might have on the cognitive workload of the drivers.

Our results suggest that traffic errors—and in a wider sense cognitive load—can be reduced when some interaction with in-vehicle devices is outsourced to BCI systems. At the same time, an increased complexity of the commands controlled through the EEG headset can negatively affect the driver's cognitive load level, manifested in our experiments through higher errors for the traffic and search tasks.

Similarly, in terms of accuracy of the EEG-based control, our findings suggest that the

execution error for the BCI commands is within acceptable limits for up to 2-4 simple commands. While this allows for the control of straightforward dashboard elements, the control of complex in-vehicle systems requires further investigation, as accuracy levels decrease when embedding multiple commands that necessitate discretized operations.

Acknowledgements This work was supported by the German Research Foundation (DFG, grant number 1131) as part of the International Graduate School (IRTG) in Kaiserslautern on "Visualization of Large and Unstructured Data Sets".

References

1 Anderson, E., Potter, K., Matzen, L., Shepherd, J., Preston, G., and Silva, C. 2011. *A Study of Visualization Effectiveness using EEG and Cognitive Load*. Proc. of EuroVIS 2011, 30(3), pp. 791-800.

2 Cernea, D., Olech, P.-S., Ebert, A., and Kerren, A. 2011. *EEG-Based Measurement of Subjective Parameters in Evaluations*. 14th International Conference in Human-Computer Interaction (HCII), Orlando, Florida, USA.

3 Cernea, D., Olech, P.-S., Ebert, A., and Kerren, A. 2011. *Detecting Insight and Emotion in Visualization Applications with a Commercial EEG Headset*. SIGRAD Conference, Linköping Electronic Conference Proc., No. 65, Stockholm, Sweden.

4 van den Haak, P., van Lon, R., van der Meer, J., and Rothkrantz, L. 2010. *Stress Assessment of Car-Drivers using EEG-Analysis*. In Proc. of the 11th International Conference on Computer Systems and Technologies (CompSysTech'10), ACM Press, New York, NY, USA, pp. 473-477.

5 Iturrate, I., Antelis, J.M., Kuebler, A., and Minguez, J. 2009. *A Noninvasive Brain-Actuated Wheelchair Based on a P300 Neurophysiological Protocol and Automated Navigation*. IEEE Transactions on Robotics, 25(3), pp. 614-627.

6 Johnson, G., Waytowich, N., and Krusienski, D.J. 2011. *The Challenge of Using Scalp-EEG Input Signals for Continuous Device Control*. Foundations of Augmented Cognition, Directing the Future of Adaptive Systems, Lecture Notes in Computer Science, Vol. 6780/2011, pp. 525-527.

7 Kyung, G., Chae, S., Nam, K.H., Lee, K., and Shin, W. 2011. *Versatile Wearable Computer for Drivers. Design, User Experience, and Usability*. Theory, Methods, Tools, and Practice. Lecture Notes in Computer Science, Vol. 6770/2011, pp. 152-155.

8 Leeb, R., Friedmann, D., Mueller-Putz, G.R., Scherer, R., Slater, M., and Pfurtscheller, G. 2007. *Self-Paced (Asynchronuous) BCI Control of a Wheelchair in Virtual Environments: A Case Study with a Tetraplegic*. Intell. Neuroscience 2007, Article 7.

9 Osswald, S. and Tscheligi. 2010. *Driver Distraction: The Impact of Secondary Tasks on a Touch Display Steering Wheel*. Adjunct Proc. of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications.

10 Putze, F., Jarvis, J.-P., and Schultz, T. 2010. *Multimodal Recognition of Cognitive Workload for Multitasking in the Car*. Proc. of the 20th International Conference on Pattern Recognition (ICPR'10), IEEE Computer Society, Washington, DC, USA, pp. 3748-3751.

11 Ranky, G.N. and Adamovich, S. 2010. *Analysis of a Commercial EEG Device for the Control of a Robot Arm*. Proc. of the IEEE 36th Annual Northeast Bioengineering Conference, New York, NY, USA, pp. 1-2.

12 Stamps, K. and Hamam, Y. 2010. *Towards Inexpensive BCI Control for Wheelchair Navigation in the Enabled Environment – A Hardware Survey*. Proc. Of the 2010 International Conference on Brain Informatics (BI'10), Springer, Berlin, Germany, pp. 336-345.

13 Vourvopoulos, A. and Liarokapis, S. 2011. *Brain-Controlled NXT Robot: Tele-operating a Robot through Brain Electrical Activity*. In 3rd International Conference on Games and Virtual Worlds for Serious Applications, pp. 140-143.

14 Zhang, X., He, C., Yang, Y., and Jia, B. 2011. *A Control Approach of Unmanned Vehicle Driven by BCI*. In Proc. Internet and Multimedia Systems and Applications/747, Human-Computer Interaction.

15 Zhao, Q., Zhang, L., and Cichocki, A. 2009. *EEG-based Asynchronous BCI Control of a Car in 3D Virtual Reality Environments*. Chinese Science Bulletin-English Edition, Vol. 54, No. 1, pp. 78-87.

16 *BrainDriver - A Mind Controlled Car*. http://www.autonomos.inf.fu-berlin.de/subprojects/braindriver. Accessed on 17.10.2011.

17 *Prototype This: Mind Controlled Car*. http://dsc.discovery.com/videos/prototype-this-mind-controlled-car.html. Accessed on 17.10.2011.

18 *Real-Time Control of Wheelchairs with Brain Waves*. http://www.riken.go.jp/engn/r-world/info/release/press/2009/090629/index.html. Accessed on 17.10.2011.

19 Hitt II, J.M., Kring, J.P., Daskarolis, E., Morris, C., and Mouloua, M. 1999. *Assessing Mental Workload with Subjective Measures: An Analytical Review of the NASA-TLX Index Since its Inception*. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, Vol. 43, No. 24.

20 Gugler, M., Sannelli, C., Haufe, S., Schubert, R., Schrauf, M., Kinsces, W.E., Tangermann, M., and Curio, G. 2010. *Intramodal Attentional Processing During a Monotonous Car Driving Simulation: An EEG Study*. In Annual Meeting of the Cognitive Neuroscience Society, Montreal, Canada.

A Hand-held Laser Scanner based on Multi-camera Stereo-matching

Christoph Bender[1], Klaus Denker[1], Markus Friedrich[1], Kai Hirt[1], and Georg Umlauf[1]

1 **HTWG Konstanz, Computer Graphics Lab**
Brauneggerstr. 55, D–78462 Konstanz, Germany
chbender|kdenker|kahirt|mafriedr|umlauf@htwg-konstanz.de

Abstract

Most laser scanners in engineering are extended versions of tactile measuring machines. These high precision devices are typically very expensive and hardware modifications are not possible without impairing the precision of the device.

For these reasons we built our own laser-scanner system. It is based on a multi-camera reconstruction system developed for fast 3D face reconstructions. Based on this camera system, we developed a laser-scanner using GPU accelerated stereo-matching techniques and a hand-held line-laser probe. The resulting reconstruction is solely based on the known camera positions and parameters. Thus, it is not necessary to track the position and movement of the line-laser probe. This yields an inexpensive laser-scanner system where every hardware component can be modified individually for experiments and future extensions of the system.

1998 ACM Subject Classification I.3 Computer Graphics, I.3.1 Hardware Architecture, Input Devices, I.4 Image Processing and Computer Vision, I.4.8 Scene Analysis, Stereo

Keywords and phrases Laser scanner, 3D point clouds, stereo-matching, multi-camera

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.123

1 Introduction

There are two main principles used for laser measurements [11]: time-of-flight and triangulation scanners. Time-of-flight (TOF) scanners measure the time a laser pulse needs from the emitter to the scene and back to the camera. Because they allow a large measuring distance, they are used for airborne 3D scanning in geo-sciences as in [16] and for range sensing in robotics as in [10]. Thus, for hand-held scanning at low distances this technique is not applicable.

Triangulation scanners measure the displacements of a laser line as seen from one or more cameras placed in a known distance to the laser emitter. They usually provide a much better precision than TOF scanners, but can only be used at short distances. In engineering, they are used for reverse engineering and quality measurements [15]. This type of scanners are used for example for hand-held triangulation scanners for real-time meshing as given in [3]. This algorithm simplifies the use of triangulation scanners mounted on measurement arms, which are usually very expensive.

Therefore, we describe in this paper how to build a low cost laser scanner based on the multi-camera 3D-reconstruction system we presented in [4] and a hand-held line-laser probe. This enables us to experiment and modify every individual step and component of the method at low costs and without impairing the complete system.

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 123–133

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


To this end, we first discuss related work and necessary prerequisites in Sections 2 and 3 before we describe the individual steps of our method: calibration (Section 4), line extraction (Section 5) and depth estimation (Section 6). We close with some results of our method and give a brief outlook to our further plans in Sections 7 and 8.

2 Related Work

Since TOF scanners require high-precision time measurements [11] these devices are usually expensive. Thus, for low cost scanning devices triangulation scanners are the most appropriate technology. A low cost laser scanner is described in [17]. Here, a web-cam and a line-laser probe is used. Because neither the laser position nor the intrinsic camera parameters are known, a known background pattern is used for camera calibration. The 3D coordinates of the laser line are approximated based on the plane of the line-laser probe's light fan.

While laser scanning techniques reconstruct only a single laser point or line at a time, e.g. [15], it might be faster and cheaper to use no lasers at all and reconstruct larger regions at a time. Structured light methods project a set of light pattern onto the scene. Similar to triangulation laser scanners they reconstruct the 3D information from the displacement of these light patterns for known camera positions [13, 7, 19]. Because of the light projection, these methods require a dark environment minimizing interfering light sources.

It is also possible to generate depth information without a light source. Stereo matching approaches like [8, 12, 14, 18, 4] use two or more cameras with known positions. They detect similar image regions in multiple images and use the camera positions to triangulate the depth information. However, stereo-matching is a low precision reconstruction method. It is prone to systematical errors from light conditions, reflections, and repetitions in the images.

Our laser scanning system uses traditional triangulation scanner techniques as in [11] as well as low cost scanner techniques, see e.g. [17]. We have no information about the position of the line-laser probe. So, the 3D reconstruction is based solely on the displacements of the laser line in images taken by the cameras in our multi-camera-system. No target markers or background patterns are required, because the camera positions are known a priori. Thus, this approach is similar to the stereo-matching method we used in [4].

3 Prerequisites

A triangulation laser scanner usually consists of at least one camera and a line-laser probe. We use the multi-camera system with four color cameras, see Figure 1 (left), we built in a previous project [4, 9]. This camera system was designed for stereo-matching and 3D face reconstruction and recognition. The cameras are mounted in a planar upside down Y-constellation, see Figure 1 (middle). Thus, each pair of cameras has a different disparity direction to avoid potential problems with features aligned with a single disparity direction. The four cameras are synchronized such that the cameras take the images at the same time.

To extend this camera system to a laser scanner, two line-laser probes are used, see Figure 1 (right). Each of these probes consists of a laser emitter and a cylindrical lens. The lens spreads the laser beam to a fan such that a laser line is projected. An additional lens is used to focus the fan to a certain distance. This results in a sharper projection of the laser line. The two probes differ by the color of the laser and the light intensity: The red probe emits a 15mW laser line, the green probe emits a 5mW laser line. Using multiple laser colors allows to adapt to different material properties of scanned objects. The different light intensities are partially compensated by the sensor of the color cameras, which has twice as

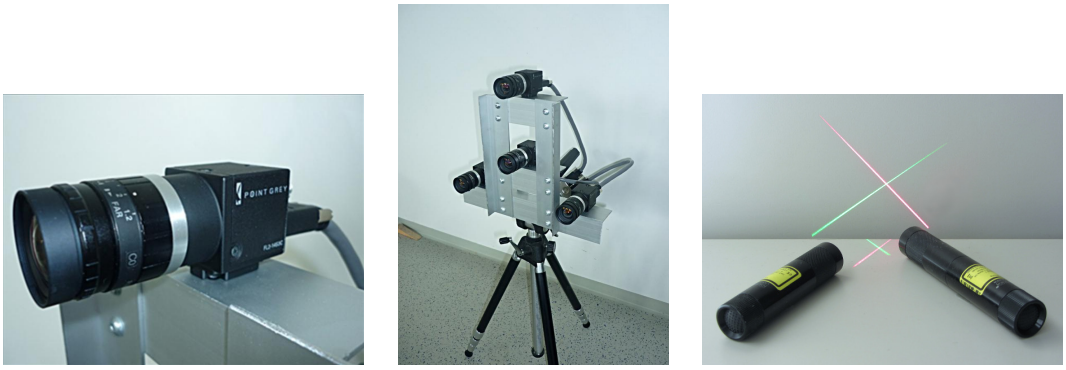


Figure 1 A system of four Point Grey Flea®2 FireWire 800 cameras (left) arranged in an upside down Y-constellation (middle). Red and green line-laser probe (right).

much green as red pixels.

Unlike usual triangulation laser scanners, in our system the position of the line-laser probe is unknown. The operator holds one of the line-laser probes in his hand and points it towards the scanned object. For each camera, the visible 2D laser line is extracted, see Section 5. A specialized stereo-matching algorithm is used to reconstruct the 3D coordinates of all points on this line, see Section 6.

4 Calibration

Since we use a hand-held laser probe, there is no calibration of the laser probes required. So, for the calibration of the scanning system the following parameters are required:

1.1 Camera parameters:

- **a.** Aperture angle α of the cameras.
- **b.** Image height h and width w in pixels of the cameras given by their resolution.

1.2 Relative positions of the cameras.

The aperture angle is computed from the physical width of the area on a planar wall that is visible in the camera image at a one meter distance of the camera to the wall.

The cameras are mounted on a Y-shaped frame of angle plates, see Figure 1 (left) and (middle). Thus, there is one central camera, which will be used as reference for the other three so-called outer cameras. In an ideal camera system the cameras are perfectly co-planar, have parallel view directions, and the central camera has the same distance $\hat{t}$ to all three outer cameras in physical space. The outer cameras are mounted in (normalized) direction $\hat{t}_i \in \mathbb{R}^2, i = 1, 2, 3$, from the central camera, where the angles of $\hat{t}_i$ to the horizontal image direction are 90°, 210°, 330°. In image space the cameras have the relative positions $t_i = t \cdot \hat{t}_i$, where t is the distance measured in pixels of the image centers of the cameras. This can be computed as $t = \hat{t}/s$, where $s = 2\tan(\alpha/2) \cdot z/w$ is the size of one pixel in physical space, if the cameras are placed at a known distance z from a planar wall. This yields the theoretical relative camera positions t_i in image space.

Because of imprecisions in the construction of the Y-frame, the t_i have to be corrected. There is a translational error, because in practice the outer cameras do not have the same distance to the central camera. Due to the construction of the Y-frame, we assume that the relative rotational errors of the cameras around the view direction and the horizontal image

direction are negligible and the relative rotational errors around the vertical image direction are relatively small. Therefore, we assume that the latter can be estimated sufficiently accurate by an additional translational error.

To estimate the translational errors, the true distances in image space of a calibration object captured by all four cameras at the same time are computed. We use as calibration object a simple red laser-point projected onto a planar wall at distance z to the cameras. This laser-point gives a point p_i, $i = 1, 2, 3, 4$, in each camera's image. In an ideal camera system, p_i, $i = 1, 2, 3$, should appear in the image of the i-th outer camera at position $p_4 + t_i$, if p_4 is the point in the central camera. Thus, the translational error is

$$c_i = p_4 + t_i - p_i, \qquad \text{for } i = 1, 2, 3.$$

To measure p_i for each image, the center of the laser-point is detected as the maximum color value after applying a Gaussian blur filter to the image.

5 Line Extraction

For triangulation scanners based on stereo-matching of camera images, the depth of a 3D point can only be computed if it is visible by at least two cameras, see Section 6. Then, stereo-matching is the process of finding corresponding points in the camera images of two different cameras at different perspectives. To accelerate this process we only use points that are on laser lines. So, these laser lines have to be extracted from the images.

For precise depth estimations, the extracted lines must be one pixel wide and the points on the extracted lines need to be at sub-pixel accuracy. Furthermore, the extraction process must be robust to noise and must execute in real time. To satisfy these requirements, we adopt techniques from [2] and [5] in Steps **2.2.** and **2.3.** of our line extraction algorithm below.

Our line extraction algorithm is applied to every image and consists of five steps described below. It takes as input an image, which is given by $I : \{-w/2, \dots, w/2\} \times \{-h/2, \dots, h/2\} \to \mathbb{R}^3$, $(x, y) \mapsto (r, g, b)$, where (x, y) are pixel coordinates. The three coordinate functions I_r, I_g, and I_b of I represent the three color channels of the image.

Line Extraction Algorithm

2.1 Binarize the source image's red channel I_r (analogously for the green channel)

$$B_I(x, y) = \begin{cases} 1, & \text{if } I_r(x, y) > t_I \\ 0, & \text{otherwise.} \end{cases}$$

The threshold t_I is set manually. We use $t_I = 0.165$.
This step removes most of the information from the image that is not necessary for the line extraction. In B_I the laser lines are several pixels wide.

2.2 Convolve the binary image B_I with the phase coded disc O_{PCD}

$$Q(x, y) = \frac{1}{\pi r^2} \sum_{u=-r}^{r} \sum_{v=-r}^{r} B_I(x + u, y + v) \cdot O_{\text{PCD}}(u, v) \in \mathbb{C}, \tag{1}$$

where O_{PCD} is defined in (2) and the radius r of the disc is chosen to be larger than the maximum width of the laser line. Details are discussed in Section 5.1.
This step yields an image with maximal values at the line centers. This line of maxima is exactly one pixel wide.

2.3 A sub-pixel accurate non-maximum suppression NMS is applied to $|Q(x,y)|$ along direction $\mathrm{Arg}(Q(x,y))/2$ to mask the line of maxima from the rest of the image

$$N(x,y) = \mathrm{NMS}\left(|Q(x,y)|, \frac{1}{2}\mathrm{Arg}(Q(x,y))\right) \in \mathbb{R},$$

where $\mathrm{Arg}(z)$ is the principal value of the phase of a complex number z. Details are given in Section 5.2.

2.4 Binarize $N(x,y)$

$$B_N(x,y) = \begin{cases} 1, & \text{if } N(x,y) > t_N \\ 0, & \text{otherwise.} \end{cases}$$

The threshold t_N is set manually. We use $t_N = 0.15$.

This step yields a binary image with sequences of line center points, that are one pixel wide, and have well defined start and end points.

2.5 Subsume the center points in B_N to line segments. Line segments that are shorter than 50 points are ignored. Details are given in Section 5.3.

This line extraction algorithm is implemented in C++ and OpenGL Shading Language GLSL. The GPU is used for

- binarization of the images (Steps **2.1.** and **2.4.**),
- convolution with the phase coded disc (Step **2.2.**), and
- non-maxima suppression (Step **2.3.**).

Only the line tracing in Step **2.5.** is computed on the CPU.

5.1 Phase Coded Disc

In Step **2.2.** we use a convolution with a phase coded disc as in [2], which is defined as

$$O_{\mathrm{PCD}}(u,v) = \begin{cases} \exp(2i\,\mathrm{Arg}(u+iv)) & , \text{if } \sqrt{u^2+v^2} \le r, \\ 0 & , \text{if } \sqrt{u^2+v^2} > r, \end{cases} \tag{2}$$

where $\exp(2i\,\mathrm{Arg}(u+iv))$ is the exponential representation of a complex number whose phase is twice the phase of $u+iv \sim (u,v)$. Note that $\mathrm{Arg}(u+iv)$ is computed by $\mathrm{atan2}(v,u)$, the four-quadrant arctangent function.

Due to the doubling of the phase in (2), the phase angles rotate twice through $[0, 2\pi]$ as (u,v) rotates once around the origin on the phase coded disc O_{PCD}, see Figure 2. This has the effect that points, whose phase differs by 90°, have opposite phases (180° difference) on O_{PCD}. Thus, they attenuate in the convolution (1). On the other hand, points with the same or opposite phases have the same phase on O_{PCD}. Thus, they amplify in the convolution (1).

For the convolution of O_{PCD} with the binarized image B_I this has the effect, that the magnitude $|Q(x,y)|$ is relatively large, if at (x,y) the laser line contains the center of O_{PCD}, see Figure 2 (left). Otherwise, $|Q(x,y)|$ is relatively small, see Figure 2 (right). So, after convolving B_I with O_{PCD}, the maxima of $|Q|$ are located on the center of the laser line. However, lines with 90° corners cannot be detected with this approach.

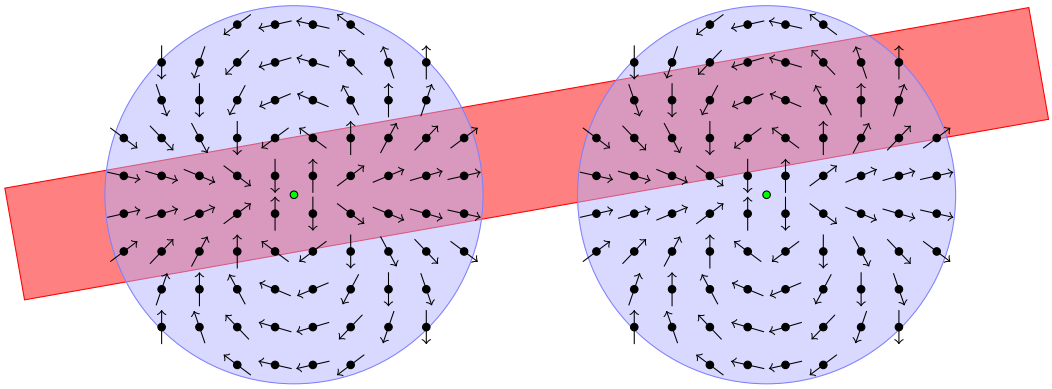


Figure 2 Phase coded discs with red laser line and small arrows visualizing the complex phase angles.

5.2 Non-maxima Suppression

To find sub-pixel accurate maxima in the convolution images, we adapted the approach of [5] in two ways. First, the magnitude of Q is used instead of the gradient's magnitude. Second, the normal $L^\perp$ to the laser line direction L is used instead of the gradient direction. The direction of the laser line $L(x, y)$ is determined by half the phase angle of $Q(x, y)$

$$L(x, y) = \frac{1}{2}\mathrm{Arg}(Q(x, y)).$$

Thus, $L^\perp(x, y)$ is perpendicular to L at (x, y).

Non-maxima Suppression

3.1 Denote by B the intersection of the line through A in direction $L^\perp$ with the line segment through the pixels A_i and A_{i+1} for a $i \in \{1, \ldots, 4\}$, see Figure 3. Analogously, C is the intersection in the opposite direction of A. Thus, the values of $|Q|$ at B and C are linearly interpolated between the values at A_i and A_{i+1} respectively A_{i+4} and A_{i+5}.

3.2 If there is no maximum at A compared to B and C, A is ignored in the result.

3.3 If there is a maximum at A, the sub-pixel accurate position is computed as the position along line $L^\perp$ of the maximum of the quadratic interpolant of the values at A, B, and C, see Figure 4.

Thus, we store per pixel the decimal places of the sub-pixel coordinates of the maximal value and the maximal value of $|Q|$.

5.3 Line Tracing

The extracted line center points in B_N have to be assigned to line segments. Thus, the output of Step **2.5.** is a list of line segments $(s_k)_k$ for each image. Each line segment s_k is a list of line center points $(p_{k,l})_{j=0}^{n_k}$.

Line Tracing

4.1 Identify start points $p_{i,0}$ in B_N by matching with the 3×3 pixel patterns in Figure 5 and generate a new segment s_k containing the start point $s_k = (p_{k,0})_l$. Start points, which are already part of a line segment, are ignored.

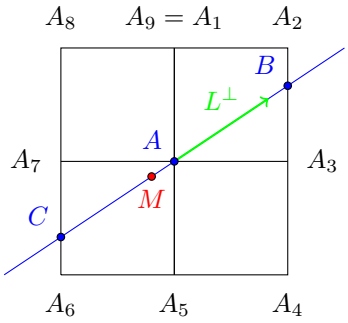


Figure 3 Sub-pixel accurate position of the maximum M on line segment CB [5].

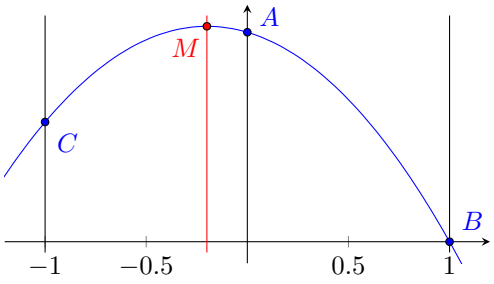


Figure 4 The quadratic interpolation of the values at A, B, C. At M is the maximum of the interpolant.

4.2 For a segment $s_k = (p_{k,0}, \ldots, p_{k,l})$ check the 8-neighborhood of point $p_{k,l}$ in B_N for pixels with value 1. If direct and diagonal neighbors are found, the former are preferred. If a new point $p_{k,l+1}$ is found, it is appended to the corresponding line segment $s_k = (p_{k,0}, \ldots, p_{k,l+1})$. Visited points are tagged to avoid multiple visits in Steps **4.1** and **4.2**.

4.3 Repeat step **4.2** until a point is identified as end-point matching the 3×3 pixel patterns in Figure 5.

4.4 An extracted line segment s_k is ignored, if it contains less than 50 points.

6 Depth Estimation

Similar to the stereo-matching method in [4], the spatial depth of the point data is reconstructed from the disparities between images. This requires a calibrated camera system

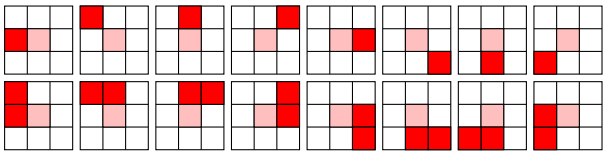


Figure 5 Start point patterns of 3×3 pixels: The white pixels have value 0 in B_N, the other pixels have value 1. Thus, the reddish pixels belong to a laser line. The pink pixel is the query pixel.

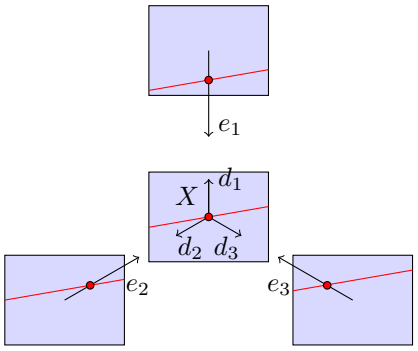


Figure 6 Schematic illustration of the four camera images with projections of epipolar lines e_i of a point X (red point) in the central camera's image on a laser line (red line), see [4].

whose lenses are corrected. For the lens correction we use the method [6, 4] to correct all camera images to a central projection in a pre-processing step. Then, the calibration as described in Section 4 is applied.

For our camera system at every instant of time a quadruple of images is generated containing one image I^i, $i = 1, 2, 3, 4$ of each camera. Accordingly, the above line extraction algorithm yields a quadruple of e.g. B_N^i where the super-script indicates the i-th outer camera for $i = 1, 2, 3$ or the central camera for $i = 4$. From the line extraction step (Section 5) we have three representations of the laser line per camera image: the line segments s_k^i, the binary image $B_N^i(x, y)$, and the sub-pixel accurate laser line center point in $N^i(x, y)$. For each point $p_{k,l}^4$ of the line segment s_k^4 of the central camera image, a depth is estimated using the images of the outer cameras:

5.1 Corresponding points in two camera images from two different cameras are identified along epipolar lines, see Section 6.1.

5.2 For each pair of corresponding points the depth is estimated by an inverse projection, see Section 6.2.

6.1 Epipolar Lines

After the calibration we assume that all four cameras of the camera system are co-planar and have parallel view directions. Therefore, a point $\hat{X}$ at infinite depth in physical space will have the same image coordinates X_i, $i = 1, 2, 3, 4$, in all four camera images I^i. A point $\hat{Y}$ not at infinity with image coordinates $Y_4 = X_4$ has image coordinates $Y_i \neq Y_4$, $i = 1, 2, 3$, in the outer cameras. As $\hat{Y}$ moves closer, Y_i, $i = 1, 2, 3$, moves along the negative camera direction $-d_i$. Thus, $\hat{Y}$ traces out a ray in physical space pointing away from the central camera. This line is called the *epipolar line* of X_4. The central projection of an epipolar line in the outer camera images is a straight line, too. Figure 6 shows the projections $e_i(X)$ of an epipolar line of X for the camera system, schematically.

For every point $p_{k,l}^4$ the epipolar line is computed and projected to the outer camera images. The resulting epipolar line projections $e_i(p_{k,l}^4)$, $i = 1, 2, 3$, are rendered to image I^i using the Bresenham algorithm [1]. If during this rendering a pixel is set that is also set in $B_N^i(x, y)$ an intersection of the epipolar line e_i with a line segment in I^i is detected. The pixel distance between $p_{k,l}^4$ and the sub-pixel accurate laser line position from $N^i(x, y)$ at the intersection is the so-called *disparity* $d_i(p_{k,l}^4)$. So, there are up to three disparities d_i, $i = 1, 2, 3$, for each $p_{k,l}^4$.

To improve the quality of the reconstruction, we use subsequently the average $d(p^4_{k,l})$ of the two disparities that are closest to each other. If there are less than two disparities or the two closest disparities differ too much, the computations for $p^4_{k,l}$ are aborted.

6.2 Inverse Projection

With the average disparity d, the depth c_z in physical space of $p^4_{k,l}$ is estimated by

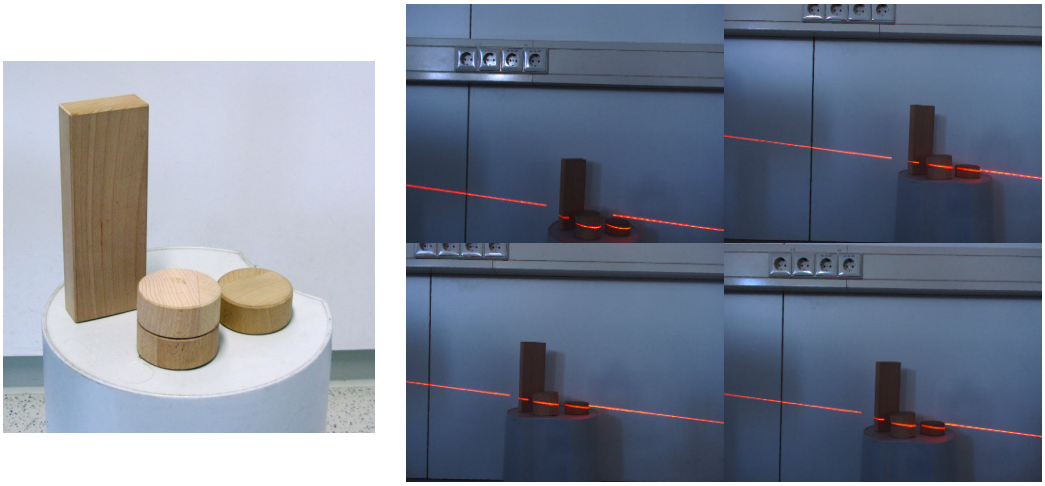
$$c_z = \frac{w\hat{t}}{2\tan\left(\frac{\alpha}{2}\right) d},$$

where α is the aperture angle of the camera, w the image width in pixels, and $\hat{t}$ the camera distance in physical space. The depth c_z together with the pixel coordinates $p^4_{k,l}$ allow an inverse projection of $p^4_{k,l}$ to 3d coordinates $[c_x, c_y, c_z]^T$ in physical space as

$$\begin{bmatrix} c_x \\ c_y \end{bmatrix} = 2\,\frac{c_z \cdot p^4_{k,l}}{w}\tan\left(\frac{\alpha}{2}\right) = p^4_{k,l}\frac{\hat{t}}{d}.$$

7 Results

To demonstrate the effectiveness of our laser scanner system, we scanned a test scene of three wooden bricks shown in Figure 7(a). The exposure of the cameras was reduced to achieve a better contrast between the laser line and the surroundings, see Figure 7(b). During a period of five minutes, we captured ca. 210,000 points at a rate of two image-quadruples per second.



(a) (b)

Figure 7 Test scene consisting of four wooden bricks (a) and the four images of the test scene scanned with a red laser line captured by the camera system (b).

Figure 8 shows a front and a top view of the computed point cloud. The overall quality of the data allows to recognize the shapes of the different wooden bricks, especially the one in the foreground. The reconstruction has a higher quality and is more robust than the pure stereo-matching method in [4]. However, an elaborate comparison with other triangulation methods is doomed by the high costs for other hand-held triangulation devices.

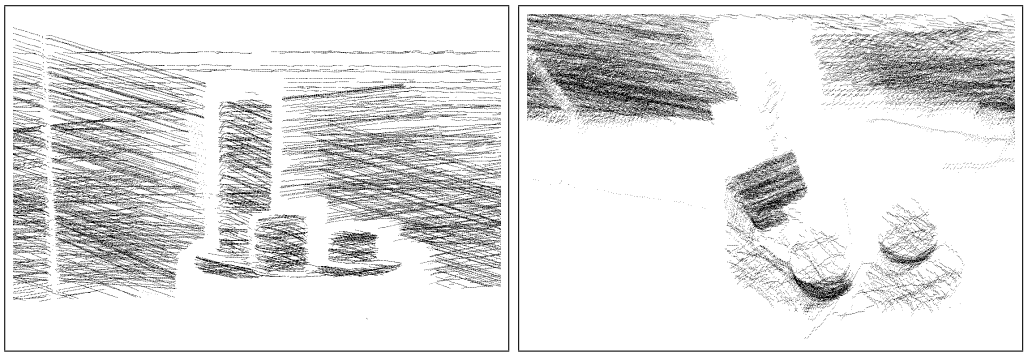


Figure 8 Point cloud from two different perspectives (left: front view, right: top view) of the test scene in Figure 7 scanned with a red laser line.

The scan lines in the point clouds show some small periodic noise in view direction of the cameras. We think this might be caused by three effects:

- Aliasing at the line extraction Step **2.2.**: The binarization $B_I(x, y)$ in Step **2.1.** creates an aliased image of the laser line. In particular, if the aliasing occurs on both sides of the line simultaneously, this may cause small steps in the detected line centers.
- Intersection of e_i with the line segments in I^i in Section 6.1: The laser line position from $N^i(x, y)$ is at sub-pixel accuracy while the projected epipolar line e_i is not. Although this approximation does not affect the disparity d_i very much, a more precise intersection of the line segment with e_i could improve the results.
- Calibration in Section 4: The laser point is not detected at sub-pixel accuracy.

Another effect that we observe in the scan data is that some of the scanned laser lines appear more than once in the data at slightly different depths. This is caused by blurry camera images, e.g. by motion blur. Additional filtering after the detection of the line centers (Section 5, Step **2.2.**) could be used to avoid these artefacts.

8 Conclusion and Outlook

We demonstrated in this paper how to build a laser scanner device at low costs using an existing camera system. The presented reconstruction algorithm runs on the GPU and is fast enough to compute 3D point data during the scan. Despite the remaining problems (see Section 7) the reconstruction yields better quality and is more robust than the pure stereo-matching method in [4].

Our laser scanner is limited to scanning from a single camera position. This only allows to capture one side of an object. For complete scans of objects it is necessary to either move the camera system or rotate the object e.g. on a turn table. Combining such scans from different directions usually requires an *iterative closest point* algorithm.

It is possible to improve the quality of the scanned results in a post-processing step. All points on a laser line lie in the plane of the laser line fan. Thus, fitting this plane to each scan line and projecting the points onto this plane along the view direction of the cameras will probably solve most problems described in Section 7 and will be implemented soon.

Acknowledgments This work was supported by DFG UM 26/5-1 and DFG GK 1131.

References

1 J.E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

2 S. P. Clode, E. E. Zelniker, P. J. Kootsookos, and I. V. L. Clarkson. A phase-coded disk approach to thick curvilinear line detection. In *Proceedings of the 12th European Signal Processing Conference*, pages 1147–1150, 2004.

3 K. Denker, B. Lehner, and G. Umlauf. Real-time triangulation of point streams. *Engineering with Computers*, 27(1):67–80, 2011.

4 K. Denker and G. Umlauf. Accurate real-time multi-camera stereo-matching on the gpu for 3d reconstruction. *Journal of WSCG*, 19(1-3):9–16, 2011.

5 F. Devernay. A non-maxima suppression method for edge detection with sub-pixel accuracy. Technical Report RR-2724, INRIA, 1995.

6 F. Devernay and O. Faugeras. Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments. *Mach. Vision Appl.*, 13(1):14–24, 2001.

7 G. Frankowski, M. Chen, and T. Huth. Real-time 3d shape measurement with digital stripe projection by texas instruments micro mirror devices DMD™. *Three-Dimensional Image Capture and Applications III*, 3958(1):90–105, 2000.

8 Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *CVPR*, pages 1–8, 2007.

9 J. Hensler, K. Denker, M. Franz, and G. Umlauf. Hybrid face recognition based on real-time multi-camera stereo-matching. In G. Bebis et al., editors, *Advances in Visual Computing*, volume 6939 of *Lecture Notes in Computer Science*, pages 158–167, 2011.

10 R. A. Jarvis. A laser time-of-flight range scanner for robotic vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 5(5):505–512, 1983.

11 R. A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):122–139, 1983.

12 A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Proc. of the 18th Int. Conf. on Pattern Recognition*, pages 15–18, 2006.

13 J. L. Posdamer and M. D. Altschuler. Surface measurement by space-encoded projected beam systems. *Computer Graphics and Image Processing*, 18(1):1 – 17, 1982.

14 D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

15 C. Teutsch. *Model-based Analysis and Evaluation of Point Sets from Optical 3D Laser Scanners*. Shaker Verlag, 2007.

16 A. Wehr and U. Lohr. Airborne laser scanning - an introduction and overview. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54(2-3):68–82, 1999.

17 S. Winkelbach, S. Molkenstruck, and F. M. Wahl. Low-cost laser range scanner and fast surface registration approach. In *DAGM-Symposium*, pages 718–728, 2006.

18 R. Yang and M. Pollefeys. A versatile stereo implementation on commodity graphics hardware. *Real-Time Imaging*, 11(1):7–18, 2005.

19 S. Zhang and S.-T. Yau. High-resolution, real-time 3d absolute coordinate measurement based on a phase-shifting method. *Opt. Express*, 14(7):2644–2649, 2006.

A Survey of Dimension Reduction Methods for High-dimensional Data Analysis and Visualization*

Daniel Engel[1], Lars Hüttenberger[1], and Bernd Hamann[2]

1 **Computer Graphics and HCI Group**
University of Kaiserslautern, Germany
{d_engel, l_huette}@cs.uni-kl.de

3 **Institute for Data Analysis and Visualization & CS Department**
University of California, Davis, USA
hamann@cs.ucdavis.edu

Abstract

Dimension reduction is commonly defined as the process of mapping high-dimensional data to a lower-dimensional embedding. Applications of dimension reduction include, but are not limited to, filtering, compression, regression, classification, feature analysis, and visualization. We review methods that compute a point-based visual representation of high-dimensional data sets to aid in exploratory data analysis. The aim is not to be exhaustive but to provide an overview of basic approaches, as well as to review select state-of-the-art methods. Our survey paper is an introduction to dimension reduction from a visualization point of view. Subsequently, a comparison of state-of-the-art methods outlines relations and shared research foci.

1998 ACM Subject Classification G.3 Multivariate Statistics, I.2.6 Learning, G.1.2 Approximation

Keywords and phrases high-dimensional, multivariate data, dimension reduction, manifold learning

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.135

1 Introduction

Contemporary simulation and experimental data acquisition technologies enable scientists and engineers to generate massive amounts of data. Thereby, more and more application domains are producing progressively larger and inherently more complex (multivariate) data sets. These data sets are collections of samples that consist of multiple measured (or simulated) observations of a variable set. Expressed in a space that requires many degrees of freedom, multivariate data present severe problems for data analysis and especially for visualization. Visualization is the integral part of exploratory data analysis, the first stage of data analysis where the goal is to make sense of the data before proceeding with more goal-directed modeling and analyses. Since human perception (and output devices) is limited to three-dimensional space, the challenge of visualizing multivariate data is converting the data to a space of lower dimensionality that is depictable and comprehensible to the user while preserving as much information as possible. This process is called dimension reduction and visualization of multivariate data is one of its traditional applications.

This survey reviews methods of dimension reduction that focus on visualizing multivariate data. That is, they are suitable for a depictable target space. Our aim is not to be exhaustive

* This work was supported by the German Science Foundation (DFG).

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 135–149

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

but to provide an overview of basic approaches, as well as to review select state-of-the-art methods. Thereby, we describe the mathematical concepts and ideas underlying the algorithms. Implementation details, although important, are not discussed. The reader should be aware that there are numerous dimension reduction methods that focus on the various aspects of data analysis. For example, methods for feature reconstruction or classification are closely related to those considered here, but are not discussed because their focus is not visualization. The reader will find that, due to its long history, there are numerous surveys on dimension reduction. For example, authors focus on a specific subset of techniques [23] or investigations [25], provide a broad overview [4], or historical background [16]. This survey provides an introduction to the concepts of visualizing high-dimensional data using dimension reduction and reviews select state-of-the-art methods that share this focus.

The remainder of the paper is structured as follows. Section 2 represents the core of the survey - a detailed introduction to the concepts of dimension reduction. After a formal problem statement is given, we divide the basic approaches in two classes: projection (Section 2.1) and manifold learning (Section 2.2). We also provide a taxonomy for these methods that can act as a classifier for which data the methods are most suited. Section 3 reviews two recently developed but fundamentally different approaches to non-linear multivariate data visualization and offers a qualitative comparison between them. The object of this investigation is to infer common trends between different concepts of dimension reduction. Finally, concluding remarks are provided in Section 4.

2 Dimension reduction

Methods for dimension reduction compute a mapping from high- to low-dimensional space. The formal problem setting can be described as follows. Let $X \in \mathbb{R}^{(n\times m)}$, a set of n points in m-dimensional data space, and two metric distance (or dissimilarity) functions, $\delta_m : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ and $\delta_t : \mathbb{R}^t \times \mathbb{R}^t \to \mathbb{R}$, over data space $\mathbb{R}^m$ and target space $\mathbb{R}^t$ respectively, with $m, t \in \mathbb{N}^*, t \ll m$, be given. A mapping function ϕ that maps the m-dimensional data points ($x_i \in X$) to t-dimensional target points ($y_i \in Y$), i.e.,

$$\begin{aligned} \phi : \quad \mathbb{R}^m &\to \mathbb{R}^t \\ x_i &\mapsto y_i, \text{ for } \; 1 \leq i \leq n, \end{aligned} \tag{1}$$

is defined s.t. ϕ "faithfully" approximates pairwise distance relationships of X by those of $Y \in \mathbb{R}^{(n\times t)}$, thereby mapping close (similar) points in data space to equally close points in target space, i.e., $\delta_m(x_i, x_j) \approx \delta_t(y_i, y_j)$, for $1 \leq i, j \leq n$. In particular, an adequate mapping is designed to ensure that remote data points are mapped to remote target points.

Since the target space usually has lower degrees of freedom than those required to model distance relationships in multi-dimensional space, the mapping ϕ adheres to an inherent error that is to be minimized by its definition. Thereby, ϕ is commonly defined to minimize the least squares error $\mathcal{E}_\phi$:

$$\mathcal{E}_\phi = \sum_{1\leq i,j\leq n} W_{i,j} \; (\delta_m(x_i, x_j) - \delta_t(y_i, y_j))^2, \; \textit{for } W \in \mathbb{R}^{(n\times n)}, \tag{2}$$

where W is a weight matrix that can be used to define the importance of certain data relationships or dimensions. For example, this may be used to disregard outliers by defining $W_{i,j} = 1/\delta_m(x_i, x_j)$ (for $\delta_m(x_i, x_j) \neq 0$).

Formally, the above definitions require both data and target distance functions to be metric. That is, both functions must adhere to the properties of positive definiteness,

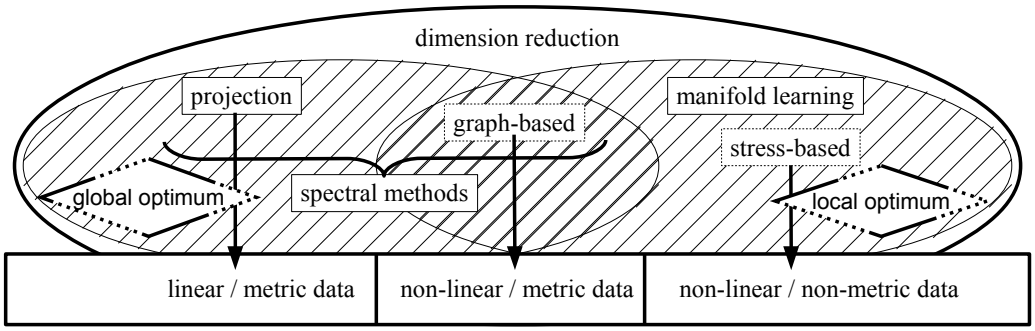


Figure 1 Concepts of dimension reduction.

symmetry, and the triangular inequality. Based on human perception, the most intuitive distance metric is the Euclidean distance, $L_2(p,p') = \sum_{1\leq i\leq q}\sqrt{(p_i - {p'}_i)^2}$ for $p,p' \in \mathbb{R}^q$. Due to its intuitiveness, the Euclidean distance is often chosen as the metric for the target space, $\delta_t = L_2$. However, the distance (or dissimilarity) measure of the application domain, δ_m, is in most cases not Euclidean and may in some cases not even be metric. For example, psychometric dissimilarities can be non-metric. In practice, this formal prerequisite can be relaxed since even an optimal mapping is, at any rate, an approximation of multivariate relationships.

In the following, we review and discuss several algorithms that realize a suitable mapping ϕ as defined above. We divide them into two basic approaches of the following underlying principal geometric ideas. If the data lie within a linear subspace of lower dimensionality, then they can be re-expressed by a linear basis transformation without loss of information. These bases can be ordered according to their contribution to the mapping error $\mathcal{E}_\phi$ and the t bases are used that minimize this error. However, if the data are non-linear and lie on an unknown manifold of lower dimensionality, then distance relationships along this manifold can be learned in an unsupervised manner and used for data mapping.

A careful taxonomy of the methods considered here is formulated in the following and illustrated in Figure 1. Methods that are solely based on linear inner product transformations are defined as projection techniques, while those that are able to ascertain distance relationships in a non-linear data structure are defined as manifold learning techniques. These techniques can be further grouped in two basic approaches. Focusing on metric data spaces, the first approach is graph-based. These methods model the data as a graph and utilize optimizations of graph theory to learn manifold distances in data space. The second approach is stress-based and focuses on the embedding directly, i.e., learning the mapping that minimizes the mapping error in target space. These methods are based on iterative optimizations of the mapping error (stress) and can learn the embedding of non-metric distances.

2.1 Projection-based Methods

Projective techniques display multi-dimensional data by projecting points onto a lower-dimensional space such that distance relationships between points in the projection space reflect specific relationships between the data points in multi-dimensional space. Since these relationships may be too complex to be completely conveyed in lower-dimensional space, projections (and all mappings considered here) are in general ambiguous. We define

a projection by the use of a projection in the geometric sense - projecting the data based on a (linear) inner product transformation. The geometric idea behind this approach is to express the data by a set of "condensed" variables that approximately model the (unknown) underlying factors and reduce redundancies. The two main approaches are to project based on variance or inner product relations and both are, in an Euclidean setting, interchangeable.

2.1.1 Principal Components Analysis (PCA)

As one of the first dimension reduction techniques discussed in the literature, Principal Components Analysis (PCA) [20] conveys distance relationships of the data by orthogonally projecting the data on a linear subspace of target dimensionality. In this specific subspace, the orthogonally projected data have maximal variance. Thereby, PCA defines a "faithful" approximation as one that captures the data's variance in an optimal way. It has been shown [13] that by the maximization of variance, PCA also minimizes the least squares error (2) for Euclidean distances in data and target space, $\delta_m = \delta_t = L_2$, under the constraint of orthogonally projecting the data:

$$\mathcal{E}_{\mathrm{PCA}} = \sum_{1 \leq i,j \leq n} (L_2(x_i, x_j) - L_2(y_i, y_j))^2. \tag{3}$$

Remarkably, PCA achieves this through a computationally efficient linear transformation. The resulting projection is a genuine view that does not distort the data. The only major drawback of PCA is that, due to its linear nature, it does not capture non-linear data well.

For the following considerations, we assume without loss of generality that $X \in \mathbb{R}^{(n \times m)}$ is centered, i.e., the mean of all given data points has been subtracted from all data points. The PCA projection is defined as

$$\begin{aligned} \mathrm{PCA}: \quad \mathbb{R}^m &\to \mathbb{R}^t \\ x_i &\mapsto x_i \, \widehat{\Gamma}, \ \textit{for } 1 \leq i \leq n, \end{aligned} \tag{4}$$

with $\widehat{\Gamma} = (\gamma^{(1)}, ..., \gamma^{(t)}) \in \mathbb{R}^{(m \times t)}$ being the matrix storing columnwise the eigenvectors of the corresponding t largest eigenvalues of the data's covariance matrix $S = n^{-1} X^T X$. The largest eigenvalue of S, λ_1, holds the variance of the data orthogonally projected in the direction of γ_1. $\widehat{\Gamma}$, storing the t mutually orthogonal vectors in which directions the data have the largest variance, define a partial orthonormal basis in data space $\mathbb{R}^m$. The orthogonal projection onto the corresponding rank-t subspace in $\mathbb{R}^m$ is defined by $\widehat{X} = X \, \widehat{\Gamma}\widehat{\Gamma}^T$. Thereby, $\widehat{X} \in \mathbb{R}^{(n \times m)}$ is the best rank-t-approximation of X (under L_2). Using the basis $\widehat{\Gamma}$, data points x_i are projected onto this subspace such that $\widehat{x}_i = \sum_{1 \leq k \leq t} \gamma^{(k)} \mathrm{PCA}(x_i)_k$, for $1 \leq i \leq n$.

Besides its broad applicability to visualization, PCA may be used for many more tasks. For example, a prominent gap in the eigenvalue spectra gives an upper bound for the intrinsic dimensionality of the data. Therefore, it is often used for filtering Gaussian noise or for reducing data size and computation time. PCA is a well-established technique with an extensive history. As such, many variants exist and more information can be found, for example, in [11] or [17].

2.1.2 Metric Multidimensional Scaling (MDS)

Metric Multidimensional Scaling (MDS) [28], also known as classical MDS, is a well-established approach that uses projection to map high-dimensional points to a linear subspace of lower dimensionality. The technique is often motivated by its goal to preserve pairwise distances

in this mapping. As such, metric MDS defines a faithful approximation as one that captures pairwise distance relationships in an optimal way; more precisely, inner product relations.

Metric MDS finds an optimal (least squares) linear fit to the given pairwise distances, assuming the distance used is metric. If Euclidean distances are given, $\delta = L_2$, metric MDS is equivalent to PCA up to scaling and rotation. However, metric MDS finds the best linear fit to any metric dissimilarities. This makes the technique more flexible to use compared to PCA. Its performance is also independent of data dimensionality, however, the method scales poorly with the number of data points.

By the method's design, the mapping error preserves inner product relations:

$$\mathcal{E}_{\text{mMDS}} = \sum_{1 \leq i,j \leq n} (x_i x_j^T - y_i y_j^T)^2. \tag{5}$$

Let a matrix of pairwise metric distances (or dissimilarities), $(\Delta)_{i,j} = \delta_{i,j}$, be given. From these metric distances, the data's Gram matrix of inner products is given by $G = HAH^T$, where $A = -1/2\delta_{i,j}^2$ and H is a centering matrix. The complete eigendecomposition of G requires $O(n^3)$ time which is, in most cases, too expensive for practical problems. However, variants of the method achieve an approximation in $O(nlogn)$ time based on a divide and conquer approach of the eigendecomposition [30]. In addition, increasingly faster solvers are being developed [14].

Metric MDS is defined as

$$\begin{aligned} \text{mMDS}: \quad \mathbb{R}^m &\to \mathbb{R}^t \\ x_i &\mapsto \widehat{\Gamma}_i \, \widehat{\Lambda}, \ for\ 1 \leq i \leq n \end{aligned} \tag{6}$$

with $\widehat{\Gamma} = (\gamma^{(1)}, ..., \gamma^{(t)}) \in \mathbb{R}^{(n \times t)}$ being the matrix storing columnwise the eigenvectors of the corresponding t largest eigenvalues of the Gram matrix of inner products, $G = XX^T$, $G_{i,j} = x_i x_j^T$. $\widehat{\Lambda}$ is the diagonal matrix storing the roots of the t largest eigenvalues of G, $\widehat{\Lambda} = diag(\sqrt{\lambda_1}, ..., \sqrt{\lambda_t})$.

Although metric MDS works in the inner product space, the geometric intuition behind the method is very similar to that of PCA. As such, points are projected into the linear subspace of largest variance. However, this subspace is defined by metric MDS based on the eigenvalue decomposition of an $n \times n$ matrix of inner products. The duality between PCA and MDS becomes clear when considering that G has the same rank and eigenvalues (up to a constant factor) as the covariance matrix $S = n^{-1}X^T X = Cov(X)$ and $G = n^{-1}Cov(X^T)$. Therefore, the Gram matrix is a covariance matrix in $\mathbb{R}^n$ that reflects the same principal relationships of the data as the covariance matrix in $\mathbb{R}^m$, although, expressed in a basis system that reflects linear combinations of data points (instead of dimensions). For more information on metric MDS, the reader is referred to [9] or [5].

Although being both powerful and flexible, Metric MDS leaves two questions unanswered: (1) What if the data are samples from a non-linear manifold and its proximity relationships are unknown? (2) What if these dissimilarities are not metric? In the following, we discuss the essential concepts that solve these two major issues. In particular, metric MDS has brought forth the variants Kernel PCA, Isomap, and non-metric MDS.

2.1.3 Kernel PCA

Kernel PCA [24] is considered a variant of PCA and metric MDS (due to their duality) that is capable of depicting non-linear data. Although distance relationships along a non-linear pattern are unknown, Kernel PCA is based on two assumptions that make the application

of (linear) PCA to non-linear data possible. The first assumption is that in the space of the data's underlying features, the data are linear. The second assumption is that there is a function that approximates the inner product of data points in this feature space. This function is called a kernel and the utilization of a non-linear kernel in a linear setting to capture non-linear data structure is known as the "kernel trick". Formally, this setting is described as follows. Let a kernel k be given that approximates inner product relations of non-linear data in their feature space, such that

$$\begin{aligned} k: \quad \mathbb{R}^m \times \mathbb{R}^m &\to \mathbb{R} \\ (x_i, x_j) &\mapsto \Phi(x_i)\Phi(x_j)^T, \ for\ 1 \leq i \leq n, \end{aligned} \tag{7}$$

where Φ is the mapping to feature space. Kernel PCA is defined as

$$\begin{aligned} \mathrm{K-PCA}: \quad \mathbb{R}^m &\to \mathbb{R}^t \\ x_i &\mapsto \widehat{\Gamma}_i\, \widehat{\Lambda}, \ for\ 1 \leq i \leq n \end{aligned} \tag{8}$$

with $\widehat{\Gamma} = (\gamma^{(1)}, ..., \gamma^{(t)}) \in \mathbb{R}^{(n \times t)}$ being the matrix storing columnwise the eigenvectors of the corresponding t largest eigenvalues of the Gram matrix of inner products *in feature space*, $G^{(k)}_{i,j} = k(x_i, x_j)$. $\widehat{\Lambda}$ is the diagonal matrix storing the roots of the t largest eigenvalues of $G^{(k)}$, $\widehat{\Lambda} = diag(\sqrt{\lambda_1}, ..., \sqrt{\lambda_t})$.

Thereby, Kernel PCA computes the eigenvectors of the covariance matrix of the data in feature space. Although this space, as well as the data coordinates therein, is unknown, the kernel maps to the data's Gram matrix of inner products in feature space. Based on the assumption of the correctness of a kernel k, the eigendecomposition of $G^{(k)}$ captures the non-linear relationships in the data by maximizing variance in feature space. As such, Kernel PCA can be viewed as a generalization of the method metric (classical) MDS by substituting the utilization of Euclidean dot products to generalized dot products.

It is not surprising that the bottleneck of Kernel PCA is finding the "right" kernel. Since distance relationships along the possibly non-linear sub-structures of the data are, in general, a-priori unknown, the definition of a suitable kernel requires explicit knowledge about the data. If this knowledge is not given, methods are better suited that determine distance relationships along non-linear data structures in an unsupervised data-driven manner. This is the concept of manifold learning.

2.2 Manifold Learning

Projection-based methods work well for data that fit approximately to a linear subspace. When this is not the case, the hope for dimension reduction is that the data follow at least a non-linear pattern, i.e., they lie on a manifold. The methods considered in this section are able to learn (and depict) proximity relationships of data points on (non-linear) manifolds in an unsupervised manner. While mappings from projection-based methods can be described by linear transformations that capture known proximity relationships, this is not the case for manifold learning techniques. In particular, these techniques abstract from Euclidean distance relationships and capture distances along a manifold. Figure 2 illustrates the difference between projection and manifold learning based mappings.

There are two distinct approaches to learn unknown proximity relationships. These approaches are based on the data being of metric or non-metric dissimilarity. To model metric distances on a manifold, graph-based techniques are often used that retrieve local distance relationships in a data-driven way and project the data based on these metric distances. However, there are various applications that require the display of non-metric

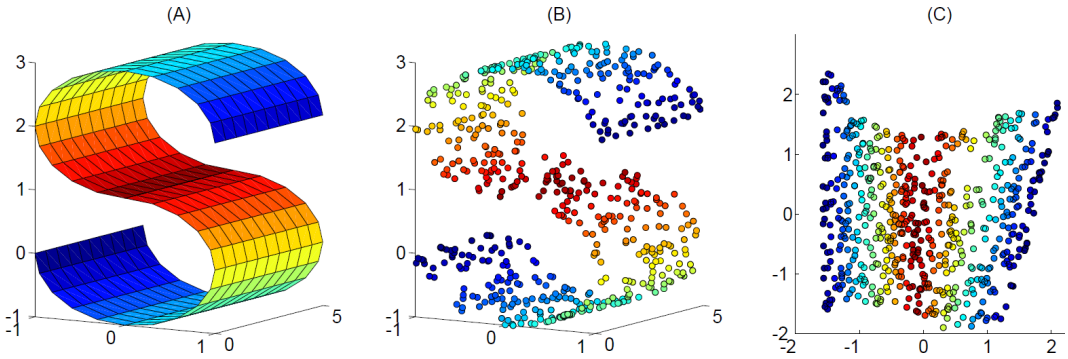


Figure 2 In this example, data sampled from a non-linear three-dimensional manifold (A) are mapped by a projection-based method (B) and by a manifold learning technique (C). In (B), the projection of the data is a linear transformation that optimally captures Euclidean distances. In (C), distance relationships along the manifold are captured by a non-linear mapping of the data. This figure derives from [21].

dissimilarity relationships. This problem cannot be solved by graph-based methods but only through a direct minimization of the mapping error in the embedding. This leads to the optimization of a non-convex stress function. Consequently, stress-based methods are prone to local minima and often slow convergence.

Graph-based methods can be divided into two classes: global and local modeling. Global approaches first learn proximity relationships on a locally low-dimensional sub-manifold and, second, depict these relationships using, for example, projection-based methods like metric MDS. Local graph-based modeling follows a divide and conquer approach. The idea is to divide the data into small groups and to solve this embedding locally. Local systems are then "pieced together" based on overlapping or fixation points. Although the projection step finds the global optimum for the embedding, the initial retrieval of distance relationships is based on optimization problems such as shortest path problems, least squares fits, or semidefinite programming. In this regard, graph-based methods are also prone to local minima or higher computational cost.

2.2.1 Non-metric MDS

The ability of metric MDS to map data relationships from a dissimilarity matrix is based on the key assumption that dissimilarities are approximate squared metric distances. As for all spectral methods, this allows for the computation of a global optimal projection. However, this also limits its application and prohibits non-metric scenarios, for example, stemming from psychometric research where metric postulates do not hold. Instead of this eigendecomposition approach, the idea of non-metric Multidimensional Scaling is to directly minimize the mapping error (2) with respect to a given non-metric dissimilarity matrix and possibly some weighting thereof. Unfortunately, due to the non-metric nature, the resulting stress function is non-convex and optimization thereof is prone to local minima.

For a perfect projection, it holds that $\mathcal{E}_{\mathrm{MDS}}(\Delta, Y, W) = 0$, where Δ is the input, Y the output, and W an optional (arbitrary) weighting. One way to approximate the solution is through a steepest descent approach, for example, with the Euler method [1]. Thereby, a step-wise iteration towards zero, where the $(k+1)^{th}$ iteration has the form $Y^{(k+1)} = Y^{(k)} + \alpha^{(k)} \nabla \mathcal{E}_{\mathrm{MDS}}(\Delta, Y^{(k)}, W)$, converges to a local minimum. The step size $\alpha^{(k)}$ can be constant or can be computed by means of line search. A disadvantage of this

method is its slow convergence near a minimum. An approach to avoid this is to use higher-level, gradient-descent-type methods, for example, Newton's methods [12]. These methods converge more quickly at a higher computational cost.

The exact embedding of non-metric dissimilarities in a metric target space is impossible. However, in non-metric MDS, the rank-order of dissimilarities is assumed to contain the most significant information, and the main goal of the approach is to depict the rank-order in its output configuration. A well-known approach to non-metric MDS is the Shepard-Kruskal algorithm [15]. At its core is a twofold optimization process that optimizes the goodness of fit with regard to the non-metric input. First, an optimal monotonic transformation of the non-metric dissimilarities to metric distances is found that preserves the rank-order of non-metric inputs. After the optimization of the rank-order distances, the output configuration is further improved iteratively, balancing both stress and monotonicity.

MDS is in all respects a hard non-convex optimization problem. Using a good initialization is therefore important. Numerous variants of MDS exist and many other methods are closely related, like Sammon's mapping [22]. Especially multi-level approaches have substantially increased performance [10]. For an overview, reference [2] is helpful.

2.2.2 Isomap

Instead of learning the embedding directly in target space, Isomap [27] attempts to explicitly model non-linear proximity relationships in terms of geodesic distances. As such, it can be viewed as a variant of metric MDS to model non-linear data using its (metric) geodesic distances. In order to retrieve these distances, a global graph-based optimization approach is utilized.

Geodesic distances are learned by linearly approximating the non-linear manifold. Thereby, a network of undirected neighborhood graphs is constructed in which each data point is a node and has edges to its neighbors that are weighted by the points' dissimilarity. The weights represent the local approximation of geodesic distances on the manifold. From these graphs, a square geodesic distance matrix is computed which is used for the metric MDS projection. The essential steps can be summarized as follows:

1. For each data point x_i compute an undirected k-neighborhood graph based on the k points of smallest dissimilarity to x_i and assign this dissimilarity as the edge's weight.[1]
2. The $(n \times n)$ matrix of geodesic distances $\tilde{\Delta}$ is found by computing the shortest paths through the network of neighborhood graphs.[2]
3. Project the data using $\tilde{\Delta}$ and metric MDS, as described in Section 2.1.2.

One problem of Isomap is that after double-centering of the geodesic distances, the Gram matrix of inner products is not guaranteed to be positive semidefinite. One variant that solves this issue is Maximum Variance Unfolding (MVU) [29]. The underlying idea behind MVU is to unfold the manifold under the constraint that local distances between neighboring points are preserved. This is optimized with respect to maximum variance.

Note that the lower-dimensional embedding of geodesic distances by Isomap involves the eigendecomposition of a dense $(n \times n)$ matrix. Like with metric MDS, this leads to significant computational effort. Further variants exist that tackle this problem, for example, by integrating a local approach [25].

[1] Often a threshold is used to model disconnected sub-manifolds.
[2] This can be computed, for example, using Dijkstra's algorithm[7].

2.2.3 Locally Linear Embedding (LLE)

In contrast to modeling a manifold by global geodesic distance relationships, LLE [21] models the manifold by extracting its local intrinsic geometry. Thereby, LLE follows a local graph-based approach. The basic idea of LLE is based on the linear approximation of all data points (in complex non-linear structures) by a convex linear combination of its neighborhood. Formally, this assumption can be described by the following equation which has to hold for all data points $x_i \in X$ and their surrounding neighbors N_i,

$$x_i = \sum_{x_j \in N_i} W_{i,j} x_j \tag{9}$$

with $0 \leq W_{i,j} \leq 1$, $\sum_{x_j \in N_i} W_{i,j} = 1$, and $W_{i,i} = 0$, for $1 \leq i, j \leq n$. The local intrinsic geometry has the appealing property that it stays unchanged under transformations like translation, rotation or scaling. Hence, the local linear relationships of points in data space directly define the intrinsic geometry for the output points to target space. The weights $W_{i,j}$ are approximated by solving a least squares problem based on a k-neighborhood graph. In contrast to Isomap, LLE models nearest neighbors by directed graphs which leads to a more suitable approximation. With these local relationships, LLE constructs a set of global equations for the projection to target space. The method is summarized as follows:

1. For each data point x_i, compute the k neighbors N_i that are nearest to x_i with respect to the distance function δ_m.
2. Compute the weights $W_{i,j}$ that minimize the equation $\sum_{i=1}^{n} |x_i - \sum_{j=1}^{n} W_{i,j} x_j|^2$ and satisfy the constraints, $W_{i,j} = 0$ if x_j is not a neighbor of x_i, $W_{i,i} = 0$ and $\sum_{j=1}^{n} W_{i,j} = 1$ for all $1 \leq i \leq n$.
3. Compute the output points y_i that minimize the equation $\sum_{i=1}^{n} |y_i - \sum_{j=1}^{n} W_{i,j} y_j|^2$.

As with Isomap, the data projection step is done by solving an $n \times n$ eigenproblem that is based on the global weight matrix W. Due to the locality of LLE, this weight matrix is sparse which leads to a significant advantage in terms of computation speed. The projection is defined by the bottom $t+1$ [3] eigenvectors of the matrix $(I-W)^T(I-W)$ that can be computed without a full matrix diagonalization [6].

3 Current State of Research

Having introduced the main concepts of dimension reduction that can be utilized for visualization, this section reviews more recent work. We compare the two dominant and distinct approaches to non-linear dimensionality reduction, namely graph- and stress-based methods. We review one representative paper of each approach, each one being both state-of-the-art and comparable in terms of similar goals and assumptions. Because both methods stem from a different background, it is likely that they have been developed independently from each other. Our goal is to infer common trends, relations, and solutions of these independent research streams that both solve the problem of finding optimal lower-dimensional embeddings for non-linear multivariate data.

[3] The bottom eigenvector is a unit vector and is discarded to enforce the constraint that the embeddings have zero mean. Here, bottom refers to the ordering imposed by largest to lowest corresponding eigenvalues.

3.1 Piecewise Laplacian-based Projection (PLP)

Similar to LLE, PLP [19] makes the assumption that every data point x_i can be approximated by a convex combination of its neighbors $x_j \in N_i$ based on weights $W_{i,j}$. While LLE finds those weights through optimization, PLP uses pre-defined weights according to:

$$W_{i,j} = \frac{1}{\delta_m(x_i, x_j)} \Bigg/ \sum_{x_k \in N_i} \frac{1}{\delta_m(x_i, x_k)} \tag{10}$$

with δ_m being the metric distance function of the data space. Due to those pre-defined weights, the projection has no unique solution. Therefore, a set of global control points is added on a divide-and-conquer basis to solve this problem. PLP divides the data in smaller subsets, each contributing a number of control points that are globally projected to preserve global relationships among subsets. This procedure allows for corrections based on user input, which makes this method interactive. PLP is defined by the following steps:

1. Separate X into $s = \sqrt{n}$ different samples S_j for $1 \leq j \leq s$. [4]
2. For each sample S_j define the neighborhoods $N_i \subseteq S_j$ for each $x_i \in S_j$ and a set of control points $C_j \subseteq S_j$.
3. Globally project all control points $C = C_1 \cup \ldots \cup C_s$ from $\mathbb{R}^m$ to $\mathbb{R}^t$.
4. For each sample S_j, construct and solve a separate local linear system but based only on the local variables C_j and the neighborhoods $N_i \subseteq S_j$.
5. Present the resulting projected data points Y to the user who can redefine the neighborhoods. Based on the new neighborhoods, repeat the method from step three.

Paulovich et. al. [19] set the number of neighbors k to ten and the number of control points in each sample S_i to $\sqrt{|S_i|}$ [5], which ensures that the number of control points of a sample corresponds to its sample size. The set of global control points C can be embedded by any appropriate mapping, for example, Paulovich et. al. use the stress-based *Force Scheme* [26].

After the local linear systems have been solved for each sample, the user can interact with the projected data set through its representation as a k-nearest neighbor graph and adjust neighborhoods or samples by simply moving data points within the embedding. Due to the used multi-level approach, only the linear systems of samples have to be recomputed in which the neighborhoods have been changed. Consequently, PLP can learn the embedding of large high-dimensional data sets in a semi-supervised manner.

If data do not come in a tagged format, partitioning them into samples is done by clustering methods. On the one hand, the multi-level approach leads to significantly smaller total computational cost since the linear systems, which are solved at step four, are now smaller. On the other hand, important global features may be missed due to this approach. Since the control points (randomly chosen) set the frame for global relation of local patches, there is no guaranty that global features can be preserved in all cases. However, the novel option of user interaction likely compensates for this scenario.

3.2 Multigrid Multidimensional Scaling (MG-MDS)

As a variant of multidimensional scaling, MG-MDS [3] is based on the direct optimization of the *weighted* mapping error as a stress function $\mathcal{E}_\phi$ given by (2), although, the method

[4] Note that $\sqrt{n}$ is an upper bound for the number of groups in a data set of size n [18]. More sophisticated estimation schemes may also be used.

[5] Note that the total number of control points amounts to $n^{3/4}$

requires distances to be metric. In contrast to PLP, weights in MG-MDS can be arbitrary and do not represent a convex combination. The basic idea is to re-state the problem of finding $\phi = \arg\min_\phi \mathcal{E}(\phi(X); \Delta, W)$ with respect to the gradient-descent-type method as a problem of finding ϕ with $\nabla \mathcal{E}(\phi(X); \Delta, W) = 0$ and to embed this problem into a multigrid approach, through which substantial performance improvements can be achieved. A simplified view of MG-MDS is that the re-stated problem is first solved for a *core*, a small subset of all data points. But instead of a one-step projection of the remaining data points, each of the remaining data points is projected separately, in a step-by-step projection. Hence, to project one of the remaining data points, not only the projected core but all the so far projected points are used. Obviously, this increases the computational cost, but approximation errors, which occur during a big, one-step projection, can be counteracted.

MG-MDS constructs a hierarchy of grids from the data set X such that for $X = \{x_{i_1}, \ldots, x_{i_n}\}$, the hierarchy is defined by choosing x_{i_n} randomly chosen from X and picking x_{i_k} from $k = n-1$ to $k = 1$ so that the following equation holds:

$$x_{i_k} = \arg\max_{x \in X} \min_{l=k+1,\ldots,n} \delta(x, x_{i_l}) \text{ for } 1 \leq k \leq n-1.$$

In other words, x_{i_k} is a data point with maximal distance to all data points with higher hierarchy level. Each grid level k holds the set of all data points of the hierarchy level equal or higher than k, i.e., $X_k = \{x_{i_k}, \ldots, x_{i_n}\}$. To transfer between grid levels, multi-grid approaches offer *restriction* P_k^{k+1} and *interpolation matrices* P_k^{k-1}, such that $N_{k-1} = P_k^{k+1} N_k$ and $N_{k-1} = P_k^{k-1} N_k$. Additionally, a corresponding stress function $\mathcal{E}_\mathbf{k}$, based on X_k, Δ_k, and W_k, determines the error.

Choosing a maximal grid level R, MG-MDS is summarized by the following steps:

1. If $r = R$, solve $\min_{X_R} s_R(X_R, T_R)$ by using Euler's or Newton's methods which are based on the gradient of s_R.
2. Otherwise, go from grid r to $r+1$, using P_r^{r+1} and ∇s_r, changing also W and Δ.
3. Apply recursively the MG-MDS method to X_{r+1} and use P_{r+1}^r to get from grid $r+1$ back to grid r.
4. During each movement from one grid to the next, a relaxation using an SMACOF-type method [2] is needed to smooth the errors which occur during the movement.

Note that the existence of P_r^{r+1} and P_r^{r-1} for all $R \leq r \leq n$ is a weaker form of the convex neighborhood assumption of LLE or PLP. P_r^{r+1} and P_r^{r-1} can be found if the data points in grid level r belong to the convex combination of the points in $r+1$, and $r-1$ respectively.

3.3 Comparison

Both approaches of stress optimization and spectral decomposition solve the problem of visualizing non-linear multivariate data. However, they achieve this in completely different ways. A comparison between them is difficult because stress optimization solves the much harder problem of embedding non-metric distance relationships, while spectral methods are restricted to metric ones. Nevertheless, such a comparison has the potential of inferring valuable insights on what generic ideas and solutions help with the problem at hand. For this, MG-MDS was chosen as a representative over numerous other state-of-the-art methods that follow the stress optimization approach, because its unique advantages are also restricted to the input being metric dissimilarities. Here, the relations between both methods are qualitatively discussed and their suitability for different scenarios is assessed. This comparison is based on the crucial factors that may delimit their application: online behavior, parametrization, and computational cost.

Assumptions

PLP (like LLE) makes the assumption that each data point can be represented by a convex combination of its nearest neighbors. Thereby, the data are approximated by a set of linear patches. MG-MDS, on the other hand, is based on the minimization of the stress function through gradient methods and uses only a weak form of this assumption.[6] Hence, for data sets where the convex combination property does not hold, no suitable neighborhood can be found, or when the computation of the neighborhoods is too costly, MG-MDS is better suited to solve the problem.

Relations

PLP and MG-MDS are similar in the sense that they do not use the whole data set at once. Instead, they use a small subset for the costly core projection[7] and then project the rest of the data with a faster method which uses the core projection. This is a definite trend and saves a significant amount of time. However, this approach requires the data set to be of sufficient size in order for a good initial core projection to be possible. Hence, for smaller data sets, methods like LLE are preferable.

Online behavior

Considering online scenarios where an existing solution is to be adjusted with regard to new data, PLP is better suited for such purpose than MG-MDS. [8] With PLP, new data points do not chance the global projection but only the local linear system within the sample which can be computed with comparably low computational cost. In this regard, MG-MDS has to be redone for grid levels in which the new data points occur. Although, most likely, the maximal grid level $r = R$ stays unchanged, the overall computational cost is higher. Both methods, however, are based on the dimension of the data points. For online scenarios where, instead of new points, new dimensions are added to the already existing data, methods solely based on local intrinsic geometry (like LLE) are advantageous. In any case, local methods are preferable for online scenarios.

Parameterization

When little is known of a data set, an extensive list of parameters often represents a burden for the analyst. However, in a visual analytics environment, the ability to tweak the mapping based on knowledge and interaction is a definite advantage. Additionally, expert knowledge is utilized that simplifies the problem of embedding. PLP requires knowledge of the "right" clustering technique, the number of clusters in the data set, the number of control points, as well as knowledge for defining the "right" neighborhood. This requires the user to have a good initial assessment on the data's structure and their global features. Therefore, when no expert is available, MG-MDS is the safer choice because it requires less user parameters (maximal grid level and core gradient method). On the other hand, PLP's ability to iteratively refine the mapping based on user interaction makes the method more suitable for visual exploration and allows one to infer this knowledge over time.

---

[6] In MG-MDS, the convex combination is only a sufficient condition but does not have to hold for all data points and also does not include neighborhood relations.

[7] Either the projection of the control points or the calculation at the maximal grid level $r = R$.

[8] It is assumed that the new data points are not taken as control points.

Computational cost

Another limiting factor for the suitability of dimension reduction methods with regard to many applications are their computational costs. The cost for computing a neighborhood graph depends on the form the data are given in. In case of a distance matrix, the cost to construct a k-neighborhood graph amounts to $O(k \cdot n)$ for each of the n data points. If the data are given as an m-dimensional point set, the computational cost to define the neighborhood for each data point is $O(m \cdot n^2)$. Although, in some cases, space partitioning data structures like *K-D trees* [8] can reduce this cost to $O(n \cdot \log n)$, their suitability for higher-dimensional spaces is an open research question. We therefore denote the cost to compute the distance matrix by $O(\text{Distance})$, while the cost to compute a k-neighborhood graph is denoted by $O(\text{Neighbors})$. With these considerations in mind, the computational costs of these two methods are:

PLP $O(\text{Distances}) + O(n^{3/2}) + O(n^{9/4}) + O(s{\cdot}\text{Sample}_{n/s})$ with $s = \sqrt{n}$ being the number of samples and $O(\text{Sample}_{n/s})$ the computational cost for each sample with size n/s. For this, a uniform size over all samples is assumed. $O(\text{Sample}_k)$ is defined as $O(\text{Sample}_k) = O(k^{3/2}) + O(k^{3/2}) +$ the computational costs to solve a linear system of size $k \times (k + \sqrt{k})$, with $\sqrt{k}$ being the number of control points in the sample. The two other terms are the cost to find the samples using a clustering method and the global projection of all $n^{3/4}$ control points using any $O(n^3)$ projection method.

MG-MDS $O((n - R)n^2) + O(2Rn^2) + O(\text{Distances})$, with R being the maximal grid level. The first term is for the core projection of grid level $r = R$ using Euler's Method. The second term is for movement between these r many grid levels. By using more complex methods than Euler's method, the computational cost increases while the value of the stress function decreases. Based on the same considerations as those made by PLP, it seems that $R = n - \sqrt{n}$ is a fair initial guess for the maximal grid level.

Note that these terms are all upper bounds. The actual computational cost can be far smaller. For example, in PLP, much effort is saved since the computation of the samples and control points uses the clustering results for the computation of the neighborhoods. Also, data may already come in a gridded or tagged form that these algorithms can use and take advantage of.

4 Conclusion

Research on dimension reduction continues at a rapid pace. This survey provides an introduction to the main concepts of dimension reduction for visualization: from linear data projection to graph- and stress-based manifold learning. Although being non-exhaustive, the comparison of state-of-the-art methods that follow the graph- or stress-based approach shows that no single method can be preferred over another. On the contrary, the effectiveness of state-of-the-art methods mainly depends on the data and application. However, the comparison also shows that there are similar research directions. At present, especially multi-level approaches show great potential and form one of the dominant research directions in both graph- and stress-based manifold learning.

Motivation for ongoing work includes manifolds of complex non-linear geometry, more flexible and interactive embeddings, better encoding of information, and scalability to data sets of peta-scale sizes. We believe that only through the incorporation of multiple concepts from different research fields, can methods for dimension reduction keep pace with future problems. Due to the increasing complexity of high-dimensional data sets, a two-dimensional

target space is not sufficient for the embedding. There is a major gap to close to the concepts of information visualization that may be used to gain additional degrees of freedom in an embedding. Furthermore, these concepts can help with better interpretability and interactivity in adjusting both view and model of the lower-dimensional mapping. Data analysis also requires the incorporation of level-of-detail approaches for data abstraction and new concepts for visual verification that evaluate the error and ambiguity of a mapping. As we have discussed, the focus of state-of-the-art methods has already changed towards semi-supervised learning that incorporates user knowledge into mapping and visualization, thereby allowing an effective visual exploration. It is likely that these knowledge-based algorithms will continue to evolve and gain in importance.

Acknowledgements The authors gratefully acknowledge the support of the German Science Foundation (DFG) for funding this research (grant #1131).

References

1 U.M. Ascher and L.R. Petzhold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998. 314 pages.

2 I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

3 M. M. Bronstein, A. M. Bronstein, R. Kimmel, and I. Yavneh. Multigrid multidimensional scaling. *Numerical Linear Algebra with Applications (NLAA)*, 13:149–171, March-April 2006.

4 Christopher J. C. Burges. Dimension reduction: A guided tour. *Foundations and Trends in Machine Learning*, 2(4), 2010.

5 T. Cox and M. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 1994.

6 James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

7 Edsger. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

8 J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):290–226, 1977.

9 W. Härdle and L. Simar. *Applied Multivariate Statistical Analysis*. Springer, 2nd edition edition, 2007.

10 Stephen Ingram, Tamara Munzner, and Marc Olano. Glimmer: Multilevel mds on the gpu. *IEEE Transactions on Visualization and Computer Graphics*, 15:249–261, 2009.

11 I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, 2002.

12 A. Kearsley, R. Tapia, and M. Trosset. The solution of the metric stress and sstress problems in multidimensional scaling using newton's method. *Computational Statistics*, 13(3):369–396, 1998.

13 Y. Koren and L. Carmel. Robust linear dimensionality reduction. *Visualization and Computer Graphics, IEEE Transactions on*, 10(4):459 –470, jul. 2004.

14 Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving sdd linear systems. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 235–244, Washington, DC, USA, 2010. IEEE Computer Society.

15 J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29, 1964.

16 John Aldo Lee and Michel Verleysen. Unsupervised dimensionality reduction: Overview and recent advances. In *IJCNN*, pages 1–8. IEEE, 2010.

17 B. F. Manly. *Multivariate statistical methods: a primer*. Chapman & Hall, Ltd., London, UK, UK, 1986.

18 N.R. Pal and J.C. Bezdek. On cluster validity for the fuzzy c-means model. *IEEE TFS*, 3(3):370–379, 1995.

19 F.V. Paulovich, D.M. Eler, J. Poco, C.P. Botha, R. Minghim, and L.G. Nonato. Piece wise laplacian-based projection for interactive data exploration and organization. *Computer Graphics Forum*, 30(3):1091–1100, 2011.

20 K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

21 Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

22 J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.*, 18(5):401–409, 1969.

23 L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. Spectral methods for dimensionality reduction. *Semisupervised Learning. MIT Press: Cambridge, MA*, 2006.

24 Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

25 Vin De Silva and Joshua B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 705–712. MIT Press, 2003.

26 E. Tejada, R. Minghim, and L.G. Nonato. On improved projection techniques to support visual exploration of multidimensional data sets. *Information Visualization*, 2(4):218–231, 2003.

27 J.B. Tenenbaum, V. Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

28 W.S. Torgerson. *Theory and methods of scaling*. Wiley, 1958.

29 Kilian Q. Weinberger and Lawrence K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*. AAAI Press, 2006.

30 Tynia Yang, Jinze Liu, Leonard Mcmillan, and Wei Wang. A fast approximation to multidimensional scaling, by. In *Proceedings of the ECCV Workshop on Computation Intensive Methods for Computer Vision (CIMCV)*, 2006.

A General Introduction To Graph Visualization Techniques

Raga'ad M. Tarawneh[1], Patric Keller[2], and Achim Ebert[1]

1 **Computer Graphics and HCI Group**
University of Kaiserslautern, Germany
{tarawneh, ebert}@cs.uni-kl.de
2 **Software Engineering: Dependability Group**
University of Kaiserslautern, Germany
pkeller@cs.uni-kl.de

Abstract

Generally, a graph is an abstract data type used to represent relations among a given set of data entities. Graphs are used in numerous applications within the field of information visualization, such as VLSI (circuit schematics), state-transition diagrams, and social networks. The size and complexity of graphs easily reach dimensions at which the task of exploring and navigating gets crucial. Moreover, additional requirements have to be met in order to provide proper visualizations. In this context, many techniques already have been introduced. This survey aims to provide an introduction on graph visualization techniques helping the reader to gain a first insight into the most fundamental techniques. Furthermore, a brief introduction about navigation and interaction tools is provided.

1998 ACM Subject Classification A.1 Introduction and Survey, B.7.2 Design Aids

Keywords and phrases Graph Visualization, Layout Algorithms, Graph Drawing, Interaction Techniques

Digital Object Identifier 10.4230/OASIcs.VLUDS.2011.151

1 Introduction

One goal of information visualization is to provide techniques for converting (abstract) information e.g., in form of textual description into visual representations facilitating the perception and handling of hidden structures from underlying data sets [18]. In cases in which corresponding data elements have inherent relationships among each other, graph visualization methods are commonly applied to support the better understanding.

▶ Definition 1. Formally, a graph $G = (V, E)$ is a mathematical structure consisting of two sets, V the set of **vertices** (or nodes) of the graph, and E the set of **edges**. Each edge has a set of one or two vertices associated to it, which are called endpoints [53].

Many application areas use graphs to represent existing structures: For example, in social networks people of a group my represent the vertices of a graph where the different relations among them are represented by a set of edges. In other areas, like biology and chemistry graphs are widely used to represent molecular and genetic maps, as well as protein production paths. In the field of software engineering, graphs are used e.g., to represent the structure of complex software systems, or to represent the internal behavior/states of compilers. In the object-oriented field, graphs are used to depict the relations among different classes, e.g., UML diagrams. In general, any hierarchical structure may be represented as a tree, which is a subtype of a graph. An example for this sort of structure is the file structure of an

Proceedings of IRTG 1131 – Visualization of Large and Unstructured Data Sets Workshop 2011.
Editors: Christoph Garth, Ariane Middel, Hans Hagen; pp. 151–164

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


operation system. The organization structure of an institute may also be represented as a tree. For more information we refer to [53, 22].

Although graph visualization techniques are widely used in many application domains, they have some limitations one has to deal with. For example, the size of the represented graph may become an issue, e.g., providing layouts for very large graphs is possible, but often comes along with the loss of readability, at least for untrained users. This is associated with the limited human cognitive power and the screen space constraints given by the visualization devices. Providing a suitable technique helping the user interacting and navigating through the data is another important issue. The goal of graph visualization techniques is to increase the comprehension level of the data by providing intuitive, intelligible layouts as well as suitable interaction mechanisms.

This survey is organized as follows: In Section 2 we present a general overview concerning layout algorithms and a set of criteria to generate clean layouts. In order to increase the comprehension level of the visualized information, many interaction techniques have been proposed in the literature. In this context we present a brief introduction in Section 3. We conclude the survey by Section 4.

2 Graph Layout Algorithms

As mentioned in Section 1, a graph consists of a set of nodes connected by a set of edges. The trivial way to display this sort of data is to use node-link diagrams. They depict the relations among the data elements in form of lines [53, 22]. In [25], another visualization approach is proposed to display graph structures by exploiting space-filling techniques or space-nested layouts which implicitly represent the relations. This section provides an overview describing both approaches and the used algorithms.

2.1 Node-Link Layouts

The basic requirement of the node-link layout concerns the computation of the coordinates of the nodes and the representation of the lines. To increase the readability a clean layout should comply with the following criteria [29]:

- Nodes and edges should be evenly distributed.
- Edge-crossings should be minimized.
- Depict symmetric sub-graphs in the same way.
- Minimize the edge bending ratio.
- Minimize the edge lengths, which helps readers detecting the relations among different nodes faster.
- In cases where the data is inherently structured distribute the nodes into different layers. This increases the understandability of the underlying graph. For example, in data-flow diagrams it is recommended to separate the graph elements into different layers in a way that the final representation reflects the original semantics.

Many other criteria have been proposed in the literature, for more details please refer to [53, 29]. It is worth mentioning that it is hard to combine most of the criteria. Some of them conflict with others. In contrast, others are hard to realize in an efficient way. Many solutions have been proposed [29, 53] to overcome these issues. Most algorithms in practice represent a trade-off. Specifying the required criteria is an application dependent process. Prioritizing a set of criteria is an important pre-condition for finding suitable layout algorithms. The work of [41] concentrates on the topic of how to prioritize such criteria.

2.1.1 The Spring Layout Algorithm

The spring layout algorithm is widely known as ***force-directed layout***, which was originally proposed by Eades in 1984 [12]. Due to its simplicity and its ability to produce a symmetric layout, the force-directed layout is considered to be one of the popular node-link layouts. The spring layout algorithm represents the graph as a physical system, in which the graph nodes are considered to be a charged particles connected to each other using a set of springs. The first model was proposed by [12]. Each node is associated with two types of forces: *attraction forces* and *repulsive forces*. Given the node coordinates and the spring attributes the method aims at reducing the total energy of the the spring system by repositioning the nodes. The attraction force f_a is applied to the neighbor nodes which are connected by a spring, while the repulsive force f_r is applied to all graph nodes. These forces are defined as follows:

$$f_a(d) = k_a log(d), \qquad f_r(d) = \frac{k_r}{d^2} \tag{1}$$

where k_a and k_r are constants and d is the current distance between two nodes. Figure 5b shows a small example that emerged from applying this algorithm. Although, the force-directed approach produces clean, symmetric layouts with respect to graphs having moderate sizes, it is considered to be one of the expensive algorithms. Its time complexity exceeds $O(n^3)$ (see [53, 29, 22]), where n is the total number of nodes. Moreover, force-directed layouts lack in terms of predictability ([53, 29, 22, 58]), meaning that running the algorithm twice, produces different results. This leads to problems in maintaining the users mental map during the interaction with unstable layouts [58]. Despite the mentioned disadvantages, the force-directed layout algorithm has been widely used in many visualization frameworks [56]. Furthermore, the algorithm itself has been revisited and optimized many times to overcome its characteristic drawbacks (see [27, 14, 16, 13, 24, 53, 29, 24, 7]).

2.1.2 Topological Feature-Based Layout

The feature-based graph drawing concept has been proposed by Archambault et al. [2]. The concept is called TopoLayout, which is a multi-level, feature-based approach. The pipeline of this approach consists of four main steps, the first one is called the ***decomposition phase*** in which the graph is decomposed into many sub-graphs based on the topological features of each internal sub-graph. For example, if the nodes in one sub-graph are topologically connected among each other in form of a tree, then the set of nodes are grouped together representing a meta-node. Currently, TopoLayout detects seven topological features, including trees, complete graphs, bi-connected components, clusters, and the undefined structure that is called unknown feature. For more details we refer to [2]. The second step called the ***feature layout phase*** in which the meta-nodes or the grouped sub-graphs are laid out using one of the layout algorithms (tuned with its topological feature). The third phase called the ***crossing reduction phase*** aims to eliminate the crossing ratio in the produced layout. Finally, the ***overlap elimination phase*** aims to change the node sizes in the final layout to ensure that no nodes overlap each other. The final result for TopoLayout is a tree representing the graph hierarchy, in which each node represents a sub-graph in the original graph and each meta-edge represents the relation between tow sub-graphs in the original graph. This layout technique helps in drawing relatively large graphs. Also, it provides the user with details about the internal structure of the graph, which can be useful in extracting more information about the graph itself (see Figure 1). GrouseFlocks [3] was introduced to provide an interactive way to explore large input graphs through cuts of a superimposed hierarchy.

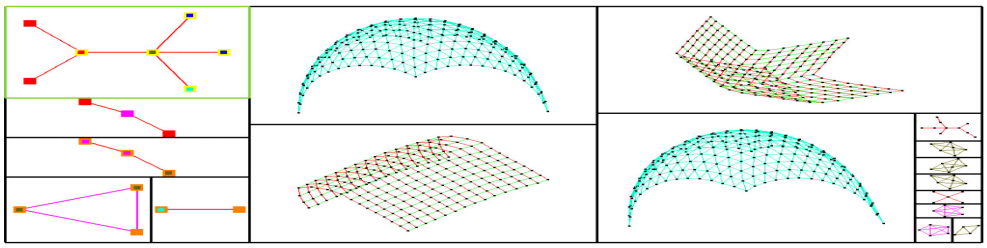


Figure 1 Layout generated by using the TopoLayout algorithm of [20].

The goal was to provide the user with the ability to see several different possible hierarchies of the same graph.

Before we introduce the tree layout concepts, it is worth to mention that both force-directed algorithms and the TopoLayout algorithm work perfectly with undirected graphs. Unfortunately, not many algorithms were designed for visualizing directed graphs. The Sugiyama algorithm was one of the first algorithm for drawing directed graphs [50]. The basic approach is to first layer the graph nodes, which means assigning a layer for each node and placing the nodes into the corresponding layer. Also, the algorithm includes two steps for reducing the edge-crossings and the node-overlappings. In general, directed graph layout algorithms are difficult to implement, this is due to the complexity of directed graphs. Therefore, many of the Sugiyama algorithm steps are considered to be NP-hard (see [17]), and some of them are NP-complete (see [11]).

2.1.3 Planar Graphs

Planar graphs are graphs that can be drawn without edge crossings in linear time. They emerge in various fields: CAD systems, circuit schematics, VLSI schematics, entity relationships diagrams and information system design [53, 29, 22]. To generate a planar layout for a general graph, some pre-requisites have to be fulfilled [22]:

- Testing whether it is possible to draw the given graph without edges crossings or not.
- Finding a planar layout algorithm satisfying the required application constrains.

Drawing a planar graph is supported by two well known algorithms, the first one called Fraysseix [9], Pach [28] and Pollack (FPP) [46] generates a drawing of a graph G on a grid of size $(2n-4)*(n-2)$ in $n\log(n)$ time. Later, the FPP algorithm was improved to run in linear time [28]. The second algorithm has been proposed by Schnyder [46]. It attempts to find a straight line embedding on a grid of n^2 nodes and runs in linear time. An example of a planar graph is shown in Figure 4b.

2.2 Tree Layout

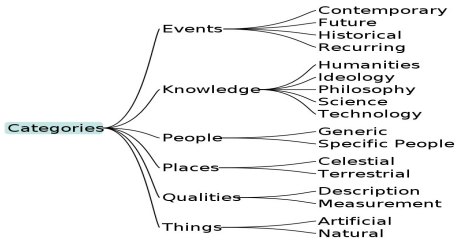
Many layout algorithms have been already proposed. In general, this may be attributed to the tree structure's simplicity and popularity. As a good starting point for tree layout algorithms we refer [53, 29].

2.2.1 Node-Link Tree layout Algorithms

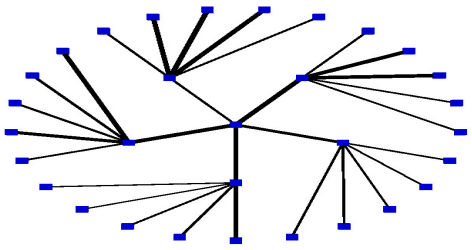
One of the basic approaches to draw a tree is to use node-link diagrams in which the parent-child relations are depicted as edges (see Section 2.1). The ***classical tree layout***

algorithm proposed by [42] is one of the early methods (see Figure 2a), it produces clean trees-representations in 2D and its implementation is straight forward. However, the technique is not declared space efficient because of its preference for one of two dominating growth directions, i.e., horizontal growth or vertical growth. To cope with this problem some compact tree layout algorithms have been implemented to produce a classical tree appearance in more compact fashion [10, 53, 29, 58, 22, 7].

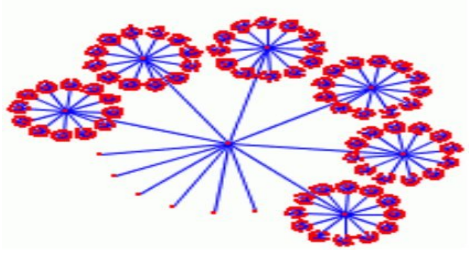
Another example of a node-link tree layout is the ***radial layout algorithm*** which was proposed by Eades [10]. A radial layout including its variations, places the root in the middle of co-centric circles and distributes the children of a sub-tree into circular shape according to their depth in the tree recursively. The radial layout uses space in more efficient way than the classical method. But it lacks the understandability of classical tree layouts, e.g. it is difficult to find the root of the tree (see Figure 2b) [53, 10, 39, 59]. As a sibling of the radial layout, the ***balloon layout*** has been introduced in [6]. Here, sibling sub-trees are drawn in a circle centered at their parents. This layout is effective in showing the tree structure. The balloon layout can be obtained by projecting a cone tree onto a plane [43, 53, 29, 58, 22] (see Figure 2c). ***H-Tree*** produces a classical layout for binary trees and works perfectly for balanced trees. But, again, it is hard to identify the root position [47] (see Figure 2d). All tree layout algorithms produce predictable results in at least linear time (the usual the complexity reaches from $O(n \log(n))$ to $O(n)$) [53, 29]. As a result of the comparison of different tree layout algorithms, we observed that the classical tree layout perfectly depicts the hierarchy structure of the tree, with sacrificing the screen space. While the radial layout, the h-tree layout, and the balloon tree layout use the screen space more efficiently but with difficulties in finding the root [53, 29, 58, 39].



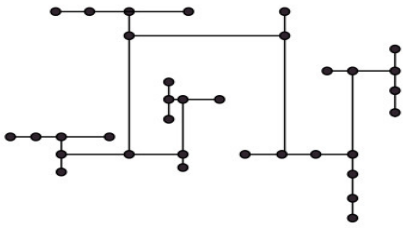
(a) Classical tree layout, produced with [19].



(b) Radial tree layout Example.



(c) Balloon tree layout: produced by [22].

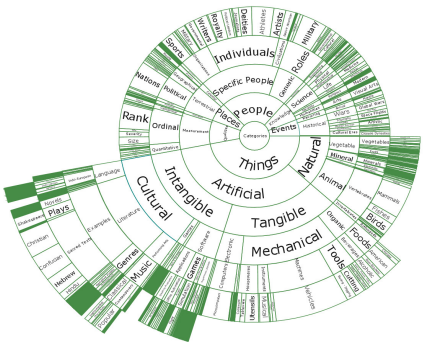


(d) H-Tree layout: produced by [22].

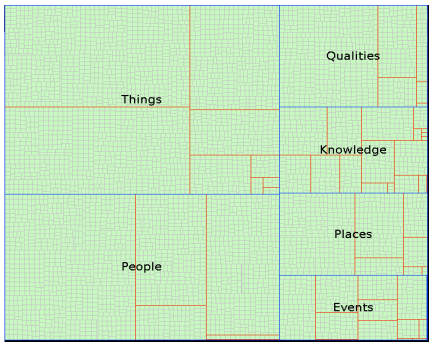
Figure 2 Tree Layout Examples.

2.2.2 Space-Filling Techniques

Space-filling techniques can be subdivided in two types: the ***Space-Division layout*** and the ***Space-Nested layout***. In the ***Space-Division layout***, the parent-child relation is depicted implicitly by attaching the children to their parent. ***Sunburst algorithm*** uses radial or circular space-filling techniques. The general belief of the developer community is that radial layout methodology better convey a hierarchy's structure without sacrificing the efficient screen space usage [49, 26]. One of the problems of this layout is that it is difficult to distinguish between the child-parent relationships and the sibling relationships, because both of them are expressed using adjacency. Moreover, due to the different number of children for each parent, the nodes sizes are difficult to control, the final layout might occupy a large space for node, which has many children. While other nodes are represented using a tiny thin rectangle that is not enough to show the node's label or the node's color (see Figure 3a). Whereas, in ***Space-Nested layouts*** the child-parent relationship is drawn using nested boxes. The idea is to place the children within their parent node. A good common example is the ***Treemap***, (see Figure 3b) [25, 48]. Nodes are represented as rectangles, each rectangle is subdivided into number of sub-rectangles equal to its children number. The subdivision process is performed recursively. This technique is popular because it uses the screen-space efficiently, and it shows the size of the leaves in a tree. However, this technique lacks the ability of showing the hierarchical structure of the tree. Also, due to the subdivision process it is highly possible to produce long and thin rectangles, which leads to problems in with the interaction (especially in selecting or highlighting the rectangles) [25, 48, 58, 22, 55].



(a) SunBurst layout.

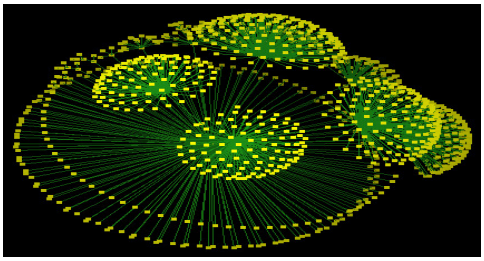


(b) TreeMap layout.

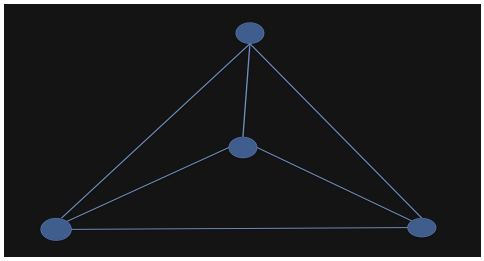
Figure 3 Examples of space-filling techniques [19].

2.3 Matrix Visualization

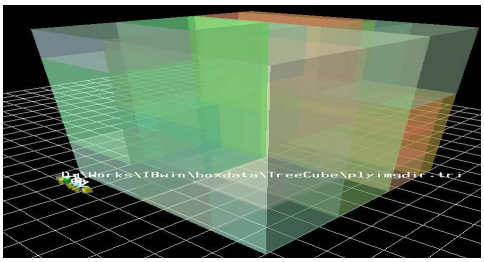
The ***matrix visualization*** is another technique that represents graph nodes relations implicitly (see Figure 5a). Here, each row and each column represent a node. The edge between two nodes is represented by the cell at which the corresponding row and column intersect. Edge attributes can be shown using different visual parameters such as color, size and shape. The scalability is limited, but the layout can produce clean representations of graphs having moderate size. However, the way the data is represented makes it difficult to detect the graph paths. For more details please refer to [1, 21].



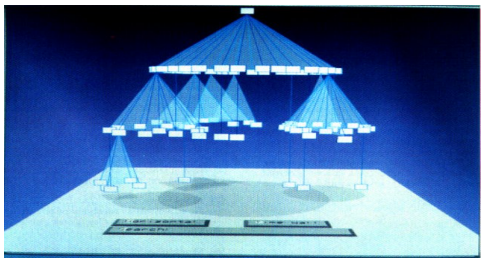
(a) Hyperbolic tree layout, produced with [52].



(b) Planar Graph Example.



(c) TreeCube layout, produced by [51].



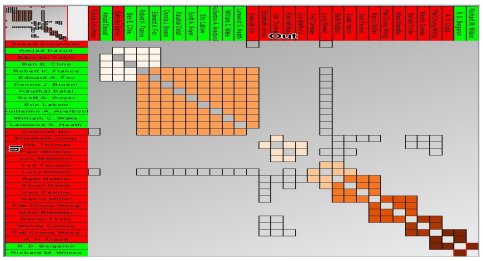
(d) Cone trees, produced by [43].

Figure 4 Graph layout Examples.

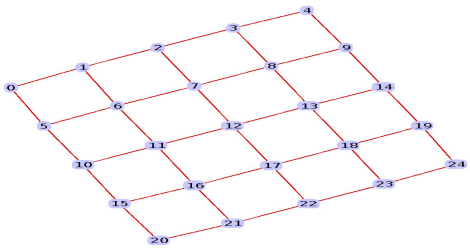
2.4 3D Layout

In addition to 2D representations, many layout algorithms have been extended to 3D. The reason behind it is that we are familiarized with 3D in the real world. So it is often more natural for us to explore data in 3D space. One example for a 3D layout is ***Treecube*** (see Figure 4c), a technique that has been proposed by [51] as an extension for the traditional treemap layout; it uses nested cubes to represent the parent-child relationships. The hyperbolic layout algorithm appeared for the first time in [32, 33], then it has successively been used by many others (e.g., [38, 37, 36]). The idea was to distribute the data entities over the hyperbolic space. Figure 4a shows an hyperbolic tree layout for a walrus-directory graph, which has been generated using the Walrus package [52]. This method can be displayed in 2D and 3D, providing a distorted view of the tree, which makes the interaction with large trees easier [22]. It is worth to mention that most of the force-directed techniques could be generalized easily to three dimensions (see [8]).

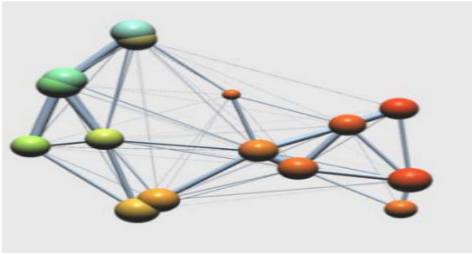
Conetree [43] is a technique that was originally developed to layout trees in 3D space. It places father nodes at the top of a cone with its children arranged evenly in the cone base. The layout has many layers; each one represents a tree level, in which all cones have the same height. The cone-base diameter for each layer is reduced in bottom-up fashion. This helps the lowest layer to fit into the width of the box containing the full cone tree, see Figure 4d . Based on [34], 3D visualization techniques face multiple challenges: First, objects in 3D may occlude each other which causes an issue while exploring the data set. Second, providing a suitable layout algorithm that assures less object-overlapping and reduces the edge-crossing is also considered as a big challenge. Third, the development of interaction techniques that are making the data exploration task easier and more intuitive is another big challenge. Finally, choosing an appropriate metaphor that increases the information understanding process is often hard to find. Many real-world metaphors were used to present data in 3D; examples can be found in [31].



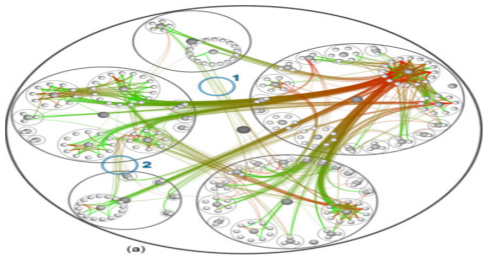
(a) Matrix visualization layout [21].



(b) Example of a force-directed layout.



(c) Clustering example by [54].



(d) Edge bundling example by[23].

Figure 5 Graph layout Techniques examples.

2.5 Nodes and Edges Clustering

Clustering techniques were introduced in graph visualization as a method to reduce visual cluttering in the final layout. This is achieved by producing an abstract view for the input graph. Reducing the number of visual elements does not only increases the clarity but also increases the rendering performance [30]. Clustering algorithms can be classified in two main types based on the criteria in the clustering process. The first type is called ***natural clustering***, here the structural information among the graph nodes is used to find a pattern of nodes having the same common criterion [44]. The second type is called the ***content-based clustering***, here the semantic meaning of the relations among the graph nodes is taken into account [57]. This type of clustering is rarely used, since it heavily depends on the application domain (reusing the same content-based clustering technique in another application usually is not possible). Therefore most graph visualization applications are using structure-based clustering algorithms. Many structural characteristics have been used as clustering criteria, such as the distances between graph nodes and node degrees. Natural clustering is widely applied to preserve the structure of the original graph [44]. This kind of clustering enables improved interaction facilities, because it eases applying of filtering techniques for the layout result and leads to an increased searching speed for specific data patterns. This could be accomplished by partitioning the nodes into a set of groups, then filter them based on a specified criteria, and finally narrow the search domain to the remaining clusters. Figure 5c shows an example of clustering techniques applied in the graph visualization field.

In [4], a clustering techniques for a special type of graph called small-world networks is presented. [53, 29] give a good overview on clustering and its applications, as well as a set of heuristics for each clustering method.

Another method for reducing the cluttering ratio is the ***edge clustering*** approach. Its goal is to free more space by grouping sets of edges that share the same end points, which reduces the visual cluttering in the final picture. Edge-bundling techniques are also proposed

to increase the readability of the graph. This is achieved by reducing the visual cluttering from the adjacency edges (see Figure 5d). For more details we refer the reader to [23].

Another example is the flow-map method: all edges that have the same source are grouped into one thick edge, generating a pattern of the edge-flow [40]. This technique intuitively shows the flow of the data from a single source to different destinations. However, it is only applicable for specific graph visualization applications, e.g., the migration path from a single source. Furthermore, in case of multiple sources, this approach causes a visual cluttering among the different flow maps, which leads to difficulties in reading the underlying graph.

3 Interaction Techniques

The goal of the visualization techniques mentioned above is to increase the comprehension level of the given data. Often, this goal cannot be achieved by only producing a static image representing the data. The ability of interacting with data has to be provided. Therefore, interaction and navigation techniques to facilitate the data exploration mission have been researched (e.g., [22, 48, 58]). In this section, we list a selection of interaction concepts that have been applied together with the visual layout algorithms. In [60] a summary of popular interaction techniques is presented:

- *Selecting:* giving the user the ability to highlight and process specific objects.
- *Abstracting/Elaborating:* changing the level of detail of the representation scheme. This allows users to get different insights into the data.
- *Reconfiguring:* giving the user the ability to change the layouts for the same representation scheme, such as sorting the graph nodes based on a specific criteria.
- *Encoding:* switching between different layout methods, such as converting the node-link diagram into a sunburst layout.
- *Exploring:* this is related to giving the user the ability to change the view point of the graph layout. Zooming and panning are examples of this category.
- *Filtering:* removing unnecessary detail and displaying the remaining items in a more visible fashion. The main concept is to filter the data nodes based on their attributes in order to make the querying process easy and fast. For more examples see [5].
- *Connecting:* giving users the ability to highlight the paths between relevant objects and the focus object.

3.1 Zooming and Panning

Zooming and Panning are basic tools for exploring large amounts of information. Panning means moving the camera across the scene, while zooming allows users to switch between the abstract and the detailed views. Geometric zooming adjusts the screen transformation and thereby allows increasing or decreasing the magnification of the displayed graph. Semantic zooming means that not only the size of objects but also the displayed information may change when approaching a particular area of the graph.

Both, zooming and panning, are complement to each other. An example are geographical maps, like the ones used by Google Earth: suppose the user zoomed into an area next to Frankfurt in Germany. If he or she wants to change the view to another area, lets say Amman in Jordan, he or she usually has to zoom out first to get a better overview, then pan to the Amman region, and finally again zoom into Amman. Doing this procedure without panning in the middle will need a long time to find the destination [22, 58]. In [15], an elegant model was introduced to explain how zooming and panning work together. The proposed concept is called space-scale diagrams. It defines an abstract space by first creating multiple copies

of the original 2D image. In a second step, they are stacked up to construct an inverted pyramid, on which each copy is placed in a certain magnification level. The space-scale diagram can be used with both zooming types, not only for the geometric zooming but also for the semantic zooming [15, 22].

3.2 Focus+Context Techniques

Focus+context techniques are addressing the problem of losing context when zooming into given data. Suppose you have zoomed into a picture, the result would be that you can only see the zoomed-in area without having an idea about the surrounding areas in the picture. Here, focus+context comes into play: it gives users the ability to see the primary object in a detailed view (focus) together with an overview of all the surrounding information (context). In general, losing context is considered to be an issue in information visualization applications. In order to alleviate this problem, focus+context techniques appeared to give the user the ability to focus on some details without losing the global context [45]. This concept does not replace the zooming and panning methods, but rather complements them. The majority of visualization application systems implement both techniques together as an interaction tool.

Many approaches provide focus+context views. Overview+detail is one of the earliest focus+context approaches, in which separate display regions for different resolutions are used. It enables users to switch between different displays frequently [35]. Fish-eye is one of the most popular focus+context techniques [45]: the area of interest becomes larger while at the same time the other regions of the layout are successively shown with less detail. In the fish-eye approach, computing the hyperbolic coordinates is faster than the layout algorithm, which is considered as an issue for the interaction with the visualization [22, 58]. The distortion appears as a negative consequence of this technique, which leads to destroying many aesthetics criteria controlling the layout algorithm, e.g., unwanted edge crossings might appear [22, 58].

4 Conclusion

The purpose of this survey was to give a brief and general overview on fundamental graph visualization techniques, a sub-field of information visualization. Graph visualization focuses on representing abstract data elements and the relationships between them visually, thus reflecting the structure of the data. The goal is to increase the cognitive level of the local and global structure.

Node-link diagrams were the first introduced approaches to depict graphs. In this regards, a graph is drawn using a set of points representing the graph vertices which are connected by lines or curves representing the graph edges. These approaches perform well for graphs of moderate size. However, data sets reflecting real world data often become very large. Consequently, this sort of algorithms appear to be insufficient and do not scale well. To adapt to larger graph sizes, new layout schemes have been developed. Space-filling techniques such as Treemaps are one approach attempting to display relatively large graphs, specifically trees, by representing the relations between the nodes implicitly. Therefore, it is difficult to answer the question: which approach performs better; This highly depends on the application and the particular user requirements. On one hand, node-link approaches lack the scalability but are able to display the relations between graph elements. On the other hand, space-filling techniques are space-efficient but lack in terms of understandability.

Along with the visual aspects, suitable and intuitive interaction techniques are key elements to gain better insights into the visualized data. Many interaction methods were introduced in the literature. In this context, zooming and panning are fundamental interaction techniques, but using them separated can cause the loss of context. Therefore, focus+context techniques were proposed to alleviate these drawbacks. Overview+detail, for example, constitutes an approach using separate display regions to resolve those issues. Fish-eye methods can provide different level of details at the same time by integrating them in a single display region. This allows the users to zoom without losing their focus.

References

1 J. Abello and F. van Ham. Matrix zoom: A visual interface to semi-external graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, pages 183–190, Washington, DC, USA, 2004. IEEE Computer Society.

2 D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13:305–317, 2007.

3 D. Archambault, T. Munzner, and D. Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14:900–913, 2008.

4 D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon. Multiscale visualization of small world networks. In *Proceedings of the Ninth annual IEEE conference on Information visualization*, INFOVIS'03, pages 75–81, Washington, DC, USA, 2003. IEEE Computer Society.

5 E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 73–80, New York, NY, USA, 1993. ACM.

6 J. Carriere and R. Kazman. Research report: Interacting with huge hierarchies: beyond cone trees. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, INFOVIS '95, pages 74–, Washington, DC, USA, 1995. IEEE Computer Society.

7 Graph Drawing Community, November 2011. `http://www.graphdrawing.org/`.

8 I. F. Cruz and J. P. Twarog. 3d graph drawing with simulated annealing. In *Proceedings of the Symposium on Graph Drawing*, GD '95, pages 162–165, London, UK, UK, 1996. Springer-Verlag.

9 H. de Fraysseix, J.s Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

10 P. Eades. Drawing free trees. *Bulletin of the Institute for Combinatorics and its Applications, pp. 10 -36*, 1992.

11 P. Eades and S. Whitesides. Drawing graphs in two layers. *Theor. Comput. Sci.*, 131(2):361–374, 1994.

12 P.A. Eades. A heuristic for graph drawing. In *Congressus Numerantium*, volume 42, pages 149–160, 1984.

13 B. Finkel and R. Tamassia. Curvilinar graph drawing using the force-directed method. In *Proc. 12th Int. Symposium on Graph Drawing, 2004, Springer LNCS 3383*, pages 448–453, 2004.

14 T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw: Pract. Exper.*, 21(11):1129–1164, November 1991.

15 G. W. Furnas and B. B. Bederson. Space-scale diagrams: understanding multiscale interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*,

CHI '95, pages 234–241, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

16 P. Gajer and S. G. Kobourov. Grip: Graph drawing with intelligent placement. In *Proceedings of the 8th International Symposium on Graph Drawing*, GD '00, pages 222–228, London, UK, UK, 2001. Springer-Verlag.

17 M. R. Garey and D. S. Johnson. Crossing Number is NP-Complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.

18 N. Gershon, S.T Card, and S. G. Eick. Information visualization tutorial. In *CHI '99 extended abstracts on Human factors in computing systems*, CHI EA '99, pages 149–150, New York, NY, USA, 1999. ACM.

19 J. Heer, Ch. Collins, and M. Dudek, November 2011. `http://prefuse.org/`.

20 Ch. Heine, November 2011. `http://www.informatik.uni-leipzig.de/~hg/libgraph/`.

21 N. Henry and J. D. Fekete. Matrixexplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12:677–684, 2006.

22 I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, January 2000.

23 D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):741–748, 2006.

24 D. Hume. *Graph Drawing Algorithms*, pages 400–401. Springer London, 2 edition, 2006.

25 B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd conference on Visualization '91*, VIS '91, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

26 H. H. Kagdi and J. I. Maletic. Onion graphs for focus+context views of uml class diagrams. In Jonathan I. Maletic, Alexandru Telea, and Andrian Marcus, editors, *VISSOFT*, pages 80–87. IEEE Computer Society, 2007.

27 T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, April 1989.

28 G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996.

29 M. Kaufmann and D. Wagner. *Drawing Graphs: Methods and Models (Lecture Notes in Computer Science)*. Springer, 1 edition, January 2001.

30 D. Kimelman, B. Leban, T. Roth, and D. Zernik. Reduction of visual complexity in dynamic graphs. In *Proceedings of the DIMACS International Workshop on Graph Drawing*, GD '94, pages 218–225, London, UK, UK, 1995. Springer-Verlag.

31 E. Kleiberg, H. van de Wetering, and J. J. Van Wijk. Botanical visualization of huge hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, INFOVIS '01, pages 87–, Washington, DC, USA, 2001. IEEE Computer Society.

32 J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

33 Jonh Lamping and Ramana Rao. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages & Computing*, 7(1):33 – 55, 1996.

34 M. Larrea, D. Urribarri, S. Martig, and S. Castro. Spherical layout implementation using centroidal voronoi tessellations. *CoRR*, abs/0912.3974, 2009.

35 H. Lieberman. Powers of ten thousand: navigating in large information spaces. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 15–16+, New York, NY, USA, 1994. ACM.

36 T. Munzner. H3: laying out large directed graphs in 3d hyperbolic space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, pages 2–, Washington, DC, USA, 1997. IEEE Computer Society.

37 T. Munzner. Drawing large graphs with h3viewer and site manager (system demonstration). In *In Proceedings of Graph Drawing'98, number 1547 in Lecture Notes in Computer Science*, pages 384–393. Springer-Verlag, 1998.

38 T. Munzner and P. Burchard. Visualizing the structure of the world wide web in 3d hyperbolic space. *Proceedings of the first symposium on Virtual reality modeling language VRML 95*, pages 33–38, 1995.

39 Q. V. Nguyen and M. L. Huang. Space-optimized tree: a connection+enclosure approach for the visualization of large hierarchies. *Information Visualization*, 2:3–15, March 2003.

40 D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 219–224, 2005.

41 H. C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th International Symposium on Graph Drawing*, GD '97, pages 248–261, London, UK, UK, 1997. Springer-Verlag.

42 E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Trans. Softw. Eng.*, 7(2):223–228, March 1981.

43 G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone Trees: animated 3D visualizations of hierarchical information. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, CHI '91, pages 189–194, New York, NY, USA, 1991. ACM.

44 T. Roxborough and A. Sen. Graph clustering using multiway ratio cut. In *Proceedings of the 5th International Symposium on Graph Drawing*, GD '97, pages 291–296, London, UK, UK, 1997. Springer-Verlag.

45 M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '92, pages 83–91, New York, NY, USA, 1992. ACM.

46 W. Schnyder. Embedding planar graphs on the grid. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, SODA '90, pages 138–148, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

47 Y. Shiloach. *Arrangements of Planar Graphs on the Planar Lattices*. PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1976.

48 G. Sindre, B. Gulla, and H. G. Jokstad. Onion graphs: aesthetics and layout. In *VL*, pages 287–291, 1993.

49 J. Stasko and E. Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proceedings of the IEEE Symposium on Information Vizualization 2000*, INFOVIS '00, pages 57–, Washington, DC, USA, 2000. IEEE Computer Society.

50 K. Sugiyama, Sh. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *Ieee Transactions On Systems Man And Cybernetics*, 11(2):109–125, 1981.

51 Y. Tanaka, Y. Okada, and K. Niijima. Treecube: Visualization tool for browsing 3d multimedia data. *International Conference on Information Visualisation*, 0:427, 2003.

52 CAIDA The Cooperative Association for Internet Data Analysis, November 2011. `http://www.caida.org/tools/visualization/walrus/`.

53 I. G. Tollis, G. Di Battista, P. Eades, and R. Tamassia. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, July 1998.

54 F. v. van Ham and J. J. v. van Wijk. Interactive Visualization of Small World Graphs. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*, pages 199–206, Washington, DC, USA, 2004. IEEE Computer Society.

55 Jarke J. Van Wijk and Huub van de Wetering. Cushion treemaps: Visualization of hierarchical information. In *Proceedings of the 1999 IEEE Symposium on Information Visualization*, INFOVIS '99, pages 73–78, Washington, DC, USA, 1999. IEEE Computer Society.

56 C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *Proceedings of the 8th International Symposium on Graph Drawing*, GD '00, pages 171–182, London, UK, UK, 2001. Springer-Verlag.

57 M. Wattenberg. Visual exploration of multivariate graphs. In *In Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 811–819. ACM Press, 2006.

58 C. Weiwei and Q. Huamin. A Survey on Graph Visualization. *Hong Kong University of Science and Technology Clear Water Bay, Kowloon, Hong Kong*, 2008.

59 K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst. Animated exploration of dynamic graphs with radial layout. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, INFOVIS '01, pages 43–, Washington, DC, USA, 2001. IEEE Computer Society.

60 J. S. Yi, Youn Ah Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.