User-Centric Parameter Specification for Interactive Virtual and Physical Visual Representations

Sergej Stoppel

Dissertation for the degree of Philosophiae Doctor (PhD)

Supervised by Stefan Bruckner
Co-supervised by Helwig Hauser

Department of Informatics
University of Bergen

July 2018

Scientific environment

The work presented in this thesis was conducted as a part of my PhD studies at the Department of Informatics, University of Bergen. In addition, I have been enrolled in the ICT Research School at the Department of Informatics, University of Bergen. During the research on my thesis I was part of the scientific communities of MedViz and MedIm. Parts of my research have been done at the Institute of Visual Computing and Human-Centered Technology at the Technical University Vienna.

MEDVIZ FROM VISION TO DECISION

MedIm Norwegian Research School in Medical Imaging

ICT Research School In Information and Communication Technology

UNIVERSITY OF BERGEN

Acknowledgements

First and foremost, I want to express my deepest gratitude to my main supervisor Stefan Bruckner who was more than a mentor but also a colleague, a partner and most importantly a friend. Stefan always found time for fruitful discussions and managed to inspire and motivate me when I felt stuck. I thank him for the patience while listening to my half-baked ideas. I have learned a great deal from Stefan and would not be writing this thesis without him.

I also want to thank Helwig Hauser, my second supervisor, for the interesting discussions ranging also beyond research. I want to thank for his shared insight and enthusiasm on chess and astronomy, and for never giving up on teaching me about arts and culture.

I want to thank my second family, a.k.a. the present and past members of the Visualization Group at the Department of Informatics at the University of Bergen. To Åsmund Birkeland who taught me how to fly. To Veronika Šoltészová for the great collaboration and fun in the last years. To Noeska Smit for the best coffee breaks I can imagine. I want to thank M. Eduard Gröller for the discussions and the guided tour through the cemetery, Julius Parulek who shared with me his experience about work and life. My gratitude goes to Andrea Brambilla from whom I inherited the organization for the monthly Visual Computing Forum. To Paolo Angelelli who helped me out with programming questions. To Ivan Kolesar who's time I wasted with discussions about crazy ideas. To Jan Byška who reliably came to all my invitations. To Erlend Hodneland for the collaboration and exchange of ideas. I want to thank the current and past PhD students that went through the same struggles and were a great company during the late evenings. Thank you Andreas Lind, Matthias Labschuetz, Chaoran Fan, Juraj Pálenik and Fabian Bolte. I want to thank Pina Kingman who was an inspiration to think outside the box. I also want to thank the Master students of the Visualization Group with whom I had excellent discussions, Magnus Paulson Erga, Thomas Trautner, Deniz Gezgin, Yngve Sekse Kristiansen, Marius Tendeland Horne and Jens Gåsemyr Magnus

I want to thank my wife Sarah, who has been extremely supportive of me throughout this entire process and has made countless sacrifices to help me get to this point. Sarah, you are my greatest adventure.

Lastly, I want to thank my parents who planted in me the love for science and kept me alive in my early years.

Abstract

Parameters are an inseparable part of virtually all computer applications. The increasing complexity of modern applications makes it difficult to fully comprehend the observed parameter spaces. Commonly, three strategies are used to address the rising complexity of parameter spaces. These strategies are: exploration strategies, reduction strategies and automation strategies. Parameter spaces are a crucial part of visualization applications as well. Data visualization encounters parameter spaces in form of high dimensional input data or constellations for parametrized algorithms in a visualization pipeline. Effective exploration and management of these parameter spaces are important aspects for the reusablility, intuitiveness and portability of visualization solutions. With the increasing size and complexity of data, the complexity of visualization systems is increasing as well. The complexity of visualization systems becomes too vast to be approached with human cognition or brute-force methods. Therefore, it is important to utilize effective parameter handling tools for data visualization.

In this thesis, we present several advancements for exploration, reduction, and automation of parameter spaces in interactive visualization. First, a method for the exploration of time dependent volumetric data is presented that introduces a new visual primitive embedded in the spatial as well as the temporal domain of the data. We propose several interaction techniques to explore the spatial and temporal aspects of the data allowing for the selection and linking over both domains. Further, we present two methods for simplifying interaction spaces. We propose one method that maps volume visualization setups to paper based visualizations. Our approach preserves a certain degree of interactivity, including advanced interactions, such as linked views and opacity modulation. Next, we discuss a method for volume manipulation through surrogate objects embedded directly in the visualization space without occluding the visualized data. Our solution is user-driven and constructs a surrogate widget according to what the user sees. We provide a set of intuitive interactions to create expressive cut away visualizations while providing full control of the widget complexity. Lastly, we introduce two approaches for automated parameter tuning. We solve the tedious task of finding suitable parameters for monochrome line art generation by automatically sampling the parameter space of the visual primitives and choosing the parameter constellations providing a result visually most similar to the input image. Our method shows high scalability in terms of canvas size and supports a wide range of artistic styles. Finally, we propose a system for automated scene illumination with animated lights. Our approach generates a dynamic light traveling on a closed path. The light path is optimized based on a flexible energy function and the scene currently visible by the user. As such the light path adapts on the fly to accommodate changes in the scene or camera translations.

All these different approaches are demonstrated with diverse use-cases, performed with domain experts as well as non-expert users. The proposed solutions present novel "intelligent" user centered interaction methods and open further possible research directions for parameter space regulation.

List of papers

This thesis is based on the following papers:

(A) **Sergej Stoppel**, Erlend Hodneland, Helwig Hauser and Stefan Bruckner. Graxels: Information Rich Primitives for the Visualization of Time-Dependent Spatial Data. In *Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine* (2016), pages 183–192.

(B) **Sergej Stoppel** and Stefan Bruckner. Vol^{2}velle: Printable Interactive Volume Visualization. In *IEEE Transactions on Visualization and Computer Graphics* (2016), 23(1), pages 861–870.

(C) **Sergej Stoppel** and Stefan Bruckner. Smart Surrogate Widgets for Direct Volume Manipulation. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis)* (2018), pages 36–45.

(D) **Sergej Stoppel** and Stefan Bruckner. LinesLab: A Flexible Low-Cost Approach for the Generation of Physical Monochrome Art. Conditionally accepted for *Computer Graphics Forum [Eurographics Conference fast track].*

(E) **Sergej Stoppel**, Magnus Paulson Erga and Stefan Bruckner. Firefly: Virtual Illumination Drones for Interactive Visualization. Accepted for *IEEE Visualization Conference 2018.*

The following publications are also related to this thesis:

(1) Veronika Soltészová, Åsmund Birkeland, **Sergej Stoppel**, Ivan Viola and Stefan Bruckner. Output-Sensitive Filtering of Streaming Volume Data. In *Computer Graphics Forum* (2017), 36(1) pages 249–262.

(2) Veronika Soltészová, Noeska Smit, **Sergej Stoppel**, Renate Gruner and Stefan Bruckner. Back to the Future: Localized Time Warping for Spatio-Temporal Selection. In submission for *Pacific Graphics 2018.*

The manuscripts presented in this thesis where written during the PhD studies of the main author. All papers were written in collaboration with Stefan Bruckner who is the main supervisor of the main author. Stefan Bruckner significantly contributed with advice and guidance to the realization and publication of the scientific work. Furthermore, Stefan Bruckner was a great source of inspiration and helped to shape ideas into novel scientific contributions. Paper A was coauthored with Helwig Hauser who aided with inspiration to the results. This paper was also coauthored by Erlend Hodneland who was responsible for the perfusion model used in one of the examples. Paper E was coauthored by Magnus Paulson Erga who helped with the implementation of the described system.

Contents

Part I

Overview

«I think I am, therefore, I am... I think.»
George Carlin

Chapter 1

Introduction

Parameter spaces are an inseparable part of our daily life, ranging from one dimensional space of room temperature adjustment to complex high dimensional parameter spaces for sophisticated simulations. We encounter parameter spaces in mundane everyday tasks. For example, during the process of baking, the careful adjustment of parameters such as the ratio between sugar, butter and flour, the amount of baking soda, the temperature of the oven, baking duration, and even the position of the cake inside the oven determine the taste and texture of a cake. If just one of the parameters is off, the cake might be ruined. And so it is not surprising that we encounter parameters in countless computer applications as well. The complexity of the parameter spaces is as diverse as the application range. As a child in the mid 1990s the author, as many others kids his age, was responsible for the death of dozens digital pets, better known as Tamagotchi. Behind the plain game interface lay a quite elaborate simulation of the creatures well being, which was controlled by four parameters: Hunger, happiness, health and discipline [247]. This four-dimensional parameter space was already complex enough for many players, and often the reason for the tragic deaths of so many digital pet companions.

The complexity of modern computer applications surpassed the simple game mechanics of such hand-held games by several magnitudes a long time ago. As the complexity of applications grows, so does the parameter space, imposing a greater cognitive load on the user. Visualization researchers proposed several strategies to address challenges arising from the growing complexity of parameter spaces. We segregate the proposed strategies into three categories: **exploratory strategies, reduction strategies, and automation strategies**. The use of these strategies is not mutually exclusive and in fact many applications can be seen as combinations of several strategies.

Approaches following the **exploratory strategy** provide the user with tools for an easier exploration of the parameter space. Here, we interpret the parameter space in a strictly mathematical sense, as the space of all possible combination of values for different parameters. As such we include measurement data into our discussion as well, since each measurement instance can be seen as an element of a high-dimensional parameter space. For instance, patient data acquired over time describes a complex parameter space that often requires exploration for the decision of further treatment. Common examples of exploration techniques are linked views and interactive filtering.

Reduction strategies aim to reduce or simplify a high-dimensional parameter space to a more intuitive low-dimensional representative. This can be done through dimension reduction algorithms or by identifying correlated parameter groups that can be reduced to a group subset. Here, the user still interactively explores the reduced parameter space. Algorithms such as t-SNE [216] or PCA are common strategies to visually abstract high-dimensional parameter spaces.

In contrast, **automation strategies** aim to compute suitable parameters without the constant supervision of the user. Automatic approaches do not require direct user input, however, many automated methods 'observe' the user and try to adjust the algorithm to suit the user's need. Common use-cases of automatic strategies are automatic parameter tuning methods for computational models or machine learning, or intelligent interaction techniques that require only minimal user input.

As mentioned earlier, these three strategies do not exclude each other and many approaches utilize more than one strategy to varying degrees. Which strategy to utilize is a question that needs to be answered for every application case separately. In this thesis we will encounter several application scenarios that will require approaches utilizing the different strategies.

1.1 Problem Statement

The research presented in this thesis is motivated by a number of challenges and problems arising when confronted with complex parameter spaces. The work covers all three strategies to approach parameter spaces. We begin with the exploration of parameter spaces and transition over parameter space mappings to automated parameter tuning methods. Each of the strategies presents its own individual challenges.

Interactive parameter space exploration imposes the need for an effective interaction technique. Depending on the task, effectiveness can be measured through the time needed for completion, precision of the technique, reproducibility of the results or the subjective measure of the method intuitiveness, to name a few examples. In addition, some scenarios might require transitions between different domains. For instance, diagnosis formulation might require the transition from the spatial domain of some data acquisition, such as ultrasound, to the value domain such as time intensity curves within localized regions. Designing and implementing suitable interaction techniques for the individual problems is a challenge that needs to be solved by the visualization expert.

In order to simplify complex parameter spaces, the visualization expert needs to find meaningful mappings from a complex space to a simplified one. Furthermore, it is necessary to design the simplified parameter space suitable for further exploration through interaction. Finding a suitable degree of abstraction while maintaining interpretability and preserving an adequate level of detail remains a challenging task.

For many use-cases the user will not be able to interact confidently in a parameter space. This is especially true for application scenarios tailored towards non-experts, for example in form of interactive museum installations. Such scenarios require intelligent interaction systems that understand what the user wants. Identifying the needs of the user and establishing flexible methods that achieve the user's intent are challenges that need to be addressed separately for each application scenario.

In addition to common quality metrics, some application scenarios require the incorporation of less objective metrics that are challenging to define. Aesthetics is one of these elusive measures. Incorporating aesthetics in automated parameter space tuning requires a construction of a new metric. Since it the traditional and the aesthetic metric do not necessarily share a common space, the construction of a new metric requires careful contribution evaluation of all parts.

1.2 Scope and Contributions

The research presented in this thesis is motivated by the challenges arising from high dimensional parameter spaces. In addition we discuss constraints on parameter spaces for physical visualizations and take into account aesthetic factors for automatic parameter tuning. More specifically, the contributions of this thesis can be summarized as follows:

1. We present a new information-rich primitive for the visual analysis of time-varying volume data. Our method enables the detailed feature-aligned visual analysis of time-dependent volume data, such as 4D ultrasound or contrast-enhanced computed tomography scans, and allows for interactive refinement and filtering. The proposed method provides a spatially localized overview of time-intensity patterns in a single view, establishing a link between the spatial and value domains.

2. We propose methods to create simplified and intuitive interaction techniques for volume visualization targeted towards novice users. To achieve intuitive interactions, we simplify the highly complex parameter spaces of typical volume visualizations by analyzing and identifying mapping strategies to simplified subspaces.

3. We analyze the fabrication limitations of physical visualizations and halftone drawings. Based on this analysis, we propose several strategies to map common visualization spaces to paperbased visualizations, as well as approaches for the physical generation of halftone drawings.

4. We demonstrate flexible approaches to incorporate aesthetics together with traditional metrics into methods for automated parameter tuning.

5. We propose automated parameter tuning methods that automatically adjust the visualization based on changes in the scene. We analyze the data that is currently seen by the user and adjust the visualization according to the current view.

1.3 Thesis Structure

This thesis is composed of two main parts. The first part provides an overview of the research carried out within the course of this thesis. The second part contains individual publications, presented verbatim with only adjusted formatting to fit the layout of this thesis. Furthermore, the bibliographies of the individual publications were merged to a single unified bibliography.

The first part, namely the overview, is structured as follows: The current chapter 1 starts with an introduction to the problem statement and the scope of the thesis. A structured overview of the related work is presented in Chapter 2, in which we discuss previous research work concerned with parameter spaces as well as work on aesthetics and physical fabrication. In Chapter 3, the contributions of this thesis are outlined in relation to the structure of the related work. Demonstrations of the proposed ideas and methods are presented in Chapter 4. Finally, we conclude the thesis in Chapter 5.

The second part of the thesis consists of five papers to provide details on the contributions outlined above. Paper A provides details on the first contribution. Papers B, C, and E clarify the second contribution. Papers B and D expand on contribution 3. Papers D and E illustrate contribution 4. Contribution 5 is shared by papers C and E.

«If I have seen further it is by standing on the shoulders of Giants.»
Isaac Newton

1

Chapter 2

State of the Art

This chapter outlines the state of the art in visualization methods for parameter spaces. In Section 2.1, 2.2 and 2.3, we discuss related work concerned with the three strategies to cope with parameter spaces, namely parameter space exploration, parameter space mapping and automation methods. Furthermore, we review approaches that integrate aesthetic considerations in Section 2.4.

2.1 Parameter Space Exploration

Visualization methods are typically designed for specific use-cases. From the the user's point of view, these tailored, single-purpose visualization algorithms tend to be the simplest ones. However, highly specialized algorithms suffer in terms of diversity of the suitable input data and the range of visual representations. The increase of data diversity and the subsequent need for broader visual representations require more general visualization algorithms. Normally as the visualization algorithms become more general, their parameter spaces tend to become more complex. The increased complexity makes it unfeasible to explore the parameter spaces directly and demands for auxiliary methods for parameter space exploration.

Commonly, the exploration of parameter spaces is achieved through a combination of an interaction technique and a suitable visual representation. Therefore, we divide the related research on parameter exploration into work focusing strongly on the interaction aspect and work with a strong focus on the visual representation. As mentioned in the introduction we understand the parameter space in a strictly mathematical sense, and do not exclude spaces derived from measurements.

Interaction Design for Parameter Exploration

Exploration techniques for parameter spaces strongly depend on the dimensionality of the parameter spaces. Typically, visualization approaches for parameter space exploration are application-oriented and propose solutions for specific application scenarios. Pringer et al. [168] presented an interactive approach for parameter spaces called HyperMoVal. HyperMoVal lays out multiple 2D and 3D sub-projections of the n-dimensional function space around a focal point, visually relating one or more n-dimensional scalar functions to known validation data. Mindek et al. [148] proposed a method for a meta analysis of visual parameters in GPU shaders. Bergner et al. [31] introduced ParaGlide, a visualization system designed for interactive exploration of

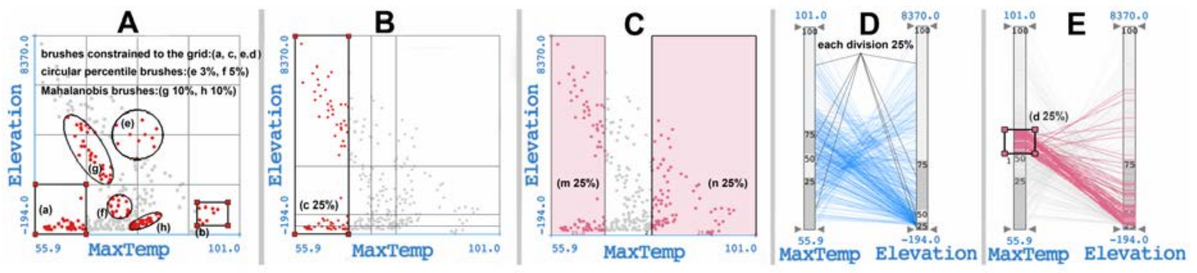


Figure 2.1: Overview of the extensions for structured brushing; courtesy of Rados et al. [182].

parameter spaces of multidimensional simulation models. ParaGlide guides the data generation with a region-based user interface for parameter sampling and divides the model's input parameter space into partitions that represent distinct output behavior. A first conceptual framework for visual parameter space analysis was discussed by Sedlmair et al. [185]. The proposed framework employs a data-flow model and provides navigation strategies independent of the application domain.

A highly effective and common approach to facilitate the understanding of essential information in complex data is the use of interactive visual analysis, or IVA for short. Naturally, IVA approaches utilize both interaction and data representation techniques. In this part, we focus on the interactive aspect first. IVA is an iterative drill down process for information gain. Typically, IVA techniques involve looking at the data through multiple correlated views and iteratively selecting features of interest that are highlighted in all views. In his work on *exploratory data analysis*, Tukey [246] advocated direct interaction with the data. Since the 1970s, the field of IVA experienced many contributions and was underlined with a solid theoretical foundation. The interaction tools of IVA can be divided into four levels of complexity, where the fourth level is specific for each data set and will not be covered here. The first level consists of brushing and linking techniques. This technique can be found in the simulation analysis tool SimVis by Doleisch [55], function graph brushing [125], for weather simulation analysis [56] and many more. The second level of IVA allows the logical combination of multiple brushes, such as AND and OR operations, providing a deeper analysis. The third level of IVA follows the principle that logical combinations of selections might be insufficient to uncover meaningful relationships in the data. One technique to uncover hidden relationships is derivation. Derivation allows for the computation of additional attributes from the data, such as derivatives, statistical properties, or clustering information. Examples of this category are the work by Kehrer et al. brushing using statistical moments [107] and the work by Janicke et al. brushing on attribute clouds [100]. A second technique of the third level IVA are advanced brushes that go beyond point and click selection. Examples of advanced brushes include angular brushes [85], similarity based brushes [159], and quantile brushes [182]. We show an overview of structured brushes in Figure 2.1.

Visual Representations for Parameter Exploration

Effective interaction design requires visual representations that support the interaction. Multiple views are a common approach to visualize complex data. A fast understanding of the data can be provided by presenting different dimensions and abstractions of a complex parameter space in multiple windows and linking them via selections. An extensive state of the art on coordinated multiple views is provided by Roberts [176]. Chang et al. [42] presented WireVis, a coordinated visualization tool based on identifying characteristic keywords of financial transactions. Fang et al. [64] introduced three different similarity measures for time intensity curves and visualized them in three different visualizations, tailored specifically for each measure. Angelelli et al. [26] used multiple views to display measures from small spatio-temporal neighborhoods and explore their temporal development through brushing and linking.

When spatial data is present, the integrated depiction of information in its spatial context can have critical impact on information perception. The possibilities for integrated depictions are highly diverse, ranging from abstract values such as labeling of objects in a three-dimensional space [209], to measurements such as profile curves [150]. Shearer et al. [188] presented Pixelplexing, a technique that encoded time-varying information in a local context through animation. Elmqvist et al. [62] suggested ZAMA, a nested visualization approach for large graph exploration. Later, Javed and Elmqvist [105] constructed a comprehensive overview on composite large scale graph visualization.

Often parameter spaces are too extensive to be visualized exhaustively. In order to display the data in a comprehensible matter the visualization is often designed to reduce clutter. A taxonomy on clutter reduction for information visualization was presented by Ellis and Dix [61]. Common approaches to visually reduce clutter include filtering operations on the data [17, 35, 169, 198], clustering approaches [73, 108, 248], and opacity regulations [65, 84, 109].

2.2 Parameter Space Mapping

During the work on this thesis, we derived our inspiration from research aiming to simplify the interaction with the visualization and the specification of visualization parameters. In this section, we first discuss methods that propose parameter mappings to simplified digital visualizations. The latter part of this section focuses on mapping methods to non-traditional, tangible visualization interfaces. The related work below guided us through our work on simplified interactions for the Vol^{2}velle (Paper B) and the Smart Surrogate Widgets (Paper C).

Mappings to Tradition Interfaces

Interaction is a vital part of visualization. To be used effectively, researchers always thrived to make interactions as intuitive and simple as possible. In 1983, Shneiderman [190] discussed the benefits of direct manipulation via graphical elements as opposed to then-common command line interfaces. The paradigm of direct manipulation has shaped the design of visualizations for over three decades, encouraging designers and visualization experts to transform complex parameter spaces into intuitive widget interfaces. Kwon et al. [127] extended the directness of interaction through the notion of surrogate widgets that embedded the parameter space directly within the visualization space. An approach by Gerl et al. [76] was designed to explicitly specify semantics in volume visualization to visually assign meaning to input as well as to output paramters of the visual mapping. Yu et al. presented FI3D [245], an approach that mapped the complex interaction parameter space to a simple direct touch interface, a

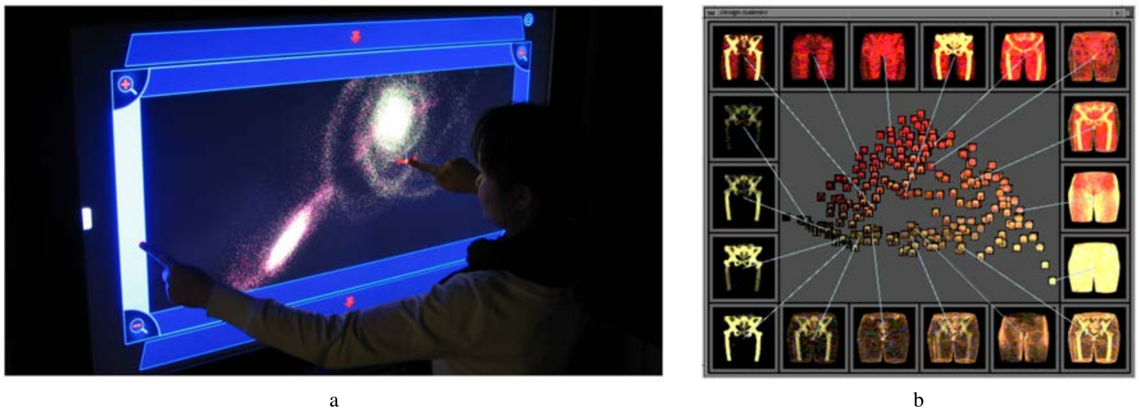


Figure 2.2: (a) An example of a simplified interaction parameter space on a touch screen interface; courtesy of Yu [245]. (b) A visual mapping of the paramter space to an intutive visual representation; courtesy of Marks [144].

photo of this system is shown in Figure 2.2 (a). Mindek et al. [149] suggested data-sensitive navigation model for user-interfaces. The proposed model model normalizes the user input according to the visual change, and also communicates the normalization degree. McGuffin et al. [147] presented an alternative strategy for volumetric cutting and peeling tools by mapping the interaction parameters for clipping and deformation to simple on-screen gestures.

A highly effective way to ease the understanding of paramter spaces are mapping methods that allow the user to observe or interact with the parameter space indirectly. Marks et al. introduced Design Galleries [144], a fundamental concept for exploring parameter spaces by using random sampling. We show a snapshot of this system in Figure 2.2 (b). König et al. [124] proposed a semi-automatic transferfunction manipulation approach that explored the transferfunction parameter space to provide suggestions and previews for different parameter settings. Ma [140] presented a visualization system that mapped the explored parameter space to an image graph and informed the user how parameter changes affect the result image. Coffey et al. [47] introduced a system where the parameter space was mapped directly onto a model, which the user could manipulate intuitively by dragging the model parts. Bruckner et al. [40] simplified the fine tuning of parameters of explosion, smoke, and fire simulations by mapping the parameter settings to the individual results and allowing the user to choose the visually pleasing results through the simulation course. Waser at al. [227] presented World-Lines, a visualization tool that allowed the user to interactively refine a simulation on the fly from within the visualization environment.

Mapping to Tangible Interfaces

Tangible interaction interfaces have been a part of visualization research for years. Nontraditional interfaces are able to present a new form of directness in human-computer interaction. The limited degrees of freedom in the tangible interaction spaces require a careful mapping from traditional interaction to the physical space. Several contributions on tangible interfaces that were presented in the last years guided our research in this area. In 1994, Hinckley et al. [89] mapped the interaction space of cutting planes to intuitive props and advocated the usefulness of real-world interfaces for neu-

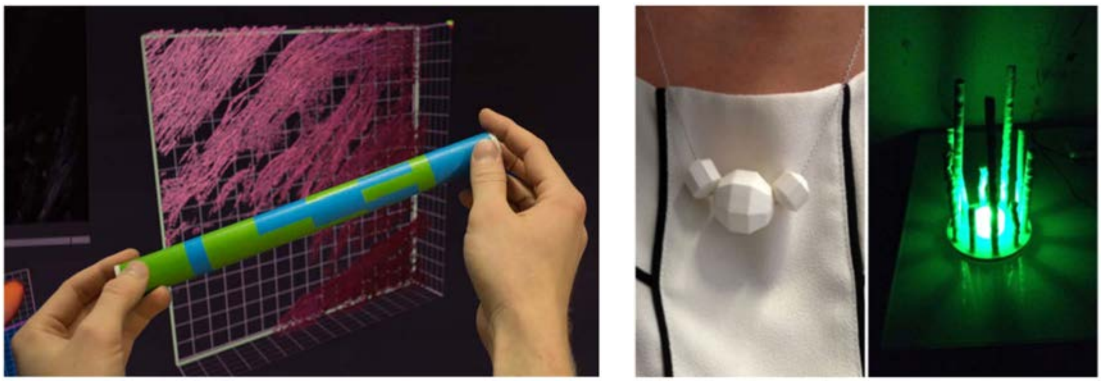


Figure 2.3: Two examples of tangible user interfaces: (a) Fiber Vis, the interaction with a paper tube are mapped to the interaction space on the screen; courtesy of Jackson et al. [98]. (b) Activity Scuplutres as reward for running activities; courtesy of Stusak et al. [204]

rosurgical visualization. Jackson et al. [98] introduced a tangible interface consisting of a paper tube (shown in Figure 2.3 (a)), where the movements of the tube were used to interact with a visualization of thin fiber structures. Holman et al. [93] combined tangible paper interfaces with projections to create interactive information displays called PaperWindows. In a simmilar manner, Spindler et al. presented two tangible paper displays, Paper-Lens [194], a spatially-aware paper display that functions as a magic lens, and Tangible Views [195], interactive paper displays for information visualization. Khalilbeigi et al. [120] presented a prototype for rollable displays and discussed a concept for effective interaction parameter mapping to the new interface. Holman et al. [92] adapted common interaction approaches for mobile devices to one-handed mobile interaction. Furthermore, Holman et al. [91] discussed interaction methods for flexible display technologies and their relation to traditional computer interfaces. Le Goc et al. examined design considerations of physical visualizations [28]. He also introduced SmartTokens [130], tangible tokens that can sense multiple types of motion. Taher et al. [207] evaluated the effectiveness of physical visualizations, but focused exclusively on dynamic bar charts.

Visualizations do not necessarily require electronic components for interactions. A comprehensive list of purely physical visualizations can be found at dataphys.org [57], where many of the presented visualizations still remain interactive without any electronic parts. Stusak et al. [204] used Activity Sculptures to investigate the value of physical visualizations as a reward for running activities. A photo of the activity sculptures is shown in Figure 2.3 (b). Stusak et al. [200] further discussed the design process and other considerations for physical visualizations. A similar evaluation was performed by Jansen et al. [104]. Furthermore, Jansen et al. [103] presented an interaction model that combines the visualization reference model with the instrumental interaction paradigm. A study on how people transform data into physical visualizations using tangible tokens as information primitives was performed by Huron et al. [96]. Stusak et al. evaluated the supportive character of physical visualizations in several studies [201–203]. Their findings suggest that physical visualizations support the analysis process and lead to significantly less information decay. Swaminathan et al. [206] presented MakerVis, a system that supports the user in mapping the digital parameter space to the

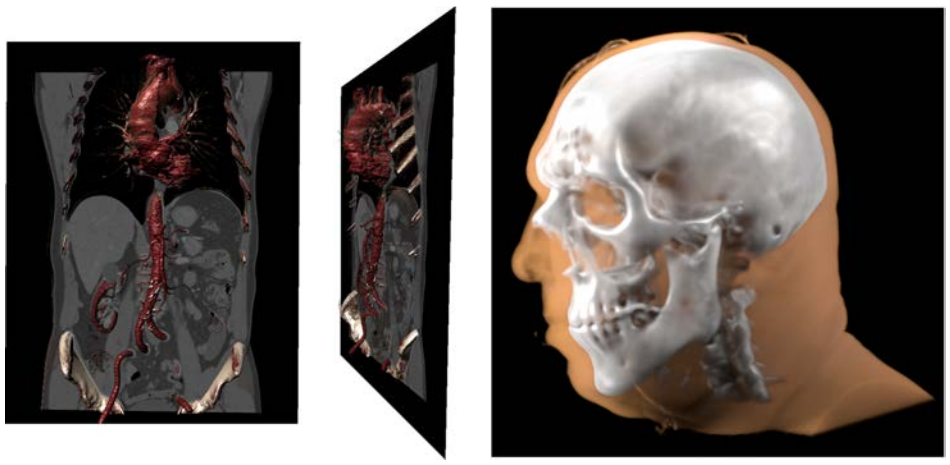


Figure 2.4: Two approaches where automation methods were used: (a) Illustrative Membrane Clipping; courtesy of Birkeland et al. [32]. (b) Automatic opacity control for volume rendering; courtesy of Ament et al. [25].

parameter space of physical visualizations. Perin et al. [167] described the fabrication of an interactive Bertin Matrix with modern fabrication devices such as laser cutters.

2.3 Automation Methods

The term automation was coined in the late 1940s by the later vice-president of the Ford Motor Company, Delmar S. Harder [145], when he summed up the goal for Ford's updated production line with the phrase "What we need is more automation.". In the following years, the term automation was picked up by economists, industrials, and engineers and was redefined to the still used definition of automation as a feedback control system. However, automatic systems are much older than one might think. The earliest dated description of a feedback system is given by the Greek inventor and mathematician Ctesibius [77] in the form of a feedback-equipped water clock. Nowadays automation is an inseparable part of computer applications. With the increasing complexity of applications, the need for automation becomes only more and more prominent. In this section, we present several general automation methods for interaction support first and focus on automated parameter tuning in the second part of this section.

Interaction Support
In visualization, automatic approaches aim to reduce the cognitive load for the user by evaluating the given data or scene and automatically compute results of otherwise tedious tasks. The application area for automated methods is extremely versatile. For example, the selection of three dimensional structures in datasets proposes a challenge in many use-cases. Radiologists often specify three dimensional structures by performing selections in two-dimensional slices and compose the selections to three-dimensional volumetric structures. Such approaches are very time consuming and tend to produce artifacts. To simplify the selection process, researchers have developed various methods to select three-dimensional features through two-dimensional computer interfaces.

Owada et al. [162] approached the challenge of three-dimensional selection by asking the user to draw the perceived outlines of the shown volume and reconstructed a segmentation based on the sketch. Later, this approach was extended by preprocessing the 2D domain before applying a 2D-to-3D stroke elevation algorithm [163]. Wiebel et al. [231] proposed visibility-driven picking in 3D. This work was later extended to map 2D strokes on the most visible feautes in a 3D rendering [230]. Building up on this idea, Stoppel et al. [199] proposed a method for selecting surface patches in direct volume rendering. Yu et al. [244] developed a family of gesture-driven Context-Aware Selection Techniques (CAST) for the interaction with and analysis of large 3D particle clouds.

Clipping is a popular technique to resolve occlusion. However, naive clipping operations cut through all features indiscriminately. Birkeland et al. [32] presented an automated technique for illustrative clipping with feature preservation by using an elastic membrane that adjusts itself to salient structures. One result of the illustrative clipping is shown in Figure 2.4 (a). Le Muzic et al. [152] proposed Visibility Equalizer, a visualization approach for mesoscopic biological models that selectively applies cutting planes based on the data semantics. Another common way to deal with occlusion is by modulating the opacity of the occluding elements. Viola and Gröller [219] tackled the occlusion with smart visibility techniques, including ghosting and cutaways. Ament et al. [25] formulated a minimization problem that automatically computes the opacity contribution of volumetric data along the viewing ray based on an importance function. In Figure 2.4 (b) we show the visualization of a human head, where a high importance was assigned to the skull. Similarly, Günther et al. [79] split the occlusion problem into two minimization problems that were solved with the least squares method.

Automated Parameter Tuning

A common problem in many applications is the choice of suitable parameters. For instance, in volume rendering the specification of a transferfunction depends on a number of parameters, and presents a challenging task that requires a high level of expertise. Many approaches have been undertaken to ease the process of defining those parameters. Ruiz et al. [179], for example, introduced a framework that allowed for the definition of a transferfunction based on a target distribution provided by the user. Borga et al. [34] used an optimization based solution to shift a preset transferfunction to account for general deviation and local variation in the data. Nguyen et al. [154] approached the definition of transferfunctions by clustering the data first, and then generating a specialized transferfunction for each cluster.

The increased complexity of virtual scenes imposes a growing demand on the cognitive abilities of the user and impairs the ability to navigate freely. To address this, methods for automatic or semi-automatic navigation were introduced. Already in 1990, Mackinlay et al. [142] proposed an automated technique for targeted viewpoint movement by allowing the user to select a point of interest and automatically moving the camera to the target. Freitag et al. [71] described a method that automatically adjusts the camera speed based on the view point quality to reduce the cognitive effort for camera control in indoor scenes. To improve the user's sense of direction in virtual reality, Xie et al. [238] proposed an automatic camera path planing method that defines a view point metric by measuring the distortion in the scene.

Illumination has a crucial impact on shape perception in virtual scenes. However, to arrive at desired results a tedious trial and error approach might be necessary. To bypass this tiresome process, Cost et al. [50] presented an automated lighting design method. The proposed approach employs optimization with an objective function based on geometry, material properties, and design goals. To address the needs of non-experts, Shacked et al. [187] developed a fully automatic technique for lighting design based on a perceptual quality metric. Gumhold [78] presented another fully automated method based on entropy, to find a light position for maximizing the information gain trough illumination. Halle and Meng developed LightKit [82], a lighting system for 3D scenes crafted after lighting design methods of artists and photographers. Similarly, Wambecke et al. [222] developed a lighting setup approach based on photographic rules, taking into account the shapes and materials of objects. Lee et al. [131] proposed to optimize the lighting setup locally on surface patches and introduced Light Collages with globally inconsistent but locally optimized lights. Wang et al. [225] divided lighting effects into global and local effects and proposed a lighting system to enhance both with different lights. Zhang and Ma [249] employed global illumination to extend three-point lighting systems to volume rendering. Consistent lighting setups are especially important in augmented reality, since the illumination of the virtual object must correspond to the existing light conditions. To add more realism to the augmented scene, Haller et al. [83] used shadow maps to automatically synchronize digital and physical lights. To match the quality of the rendered models to the captured video, Okumura et al. [160] and Klein et al. [122] incorporated blurring operations on top of the augmented image. Aittala [22] observed a real-world diffuse sphere and automatically adjusted the virtual lighting parameters to match the conditions in the real world.

2.4 Aesthetics

Artists have a long tradition of experimenting with new media forms, and so it is not surprising that they discovered computational art and computational aesthetics as a new discipline shortly after the introduction of computers. The term computer art was introduced by Edmund Berkeley in 1962 and inspired the first annual Computer Art Contest in 1963. This annual event propelled the development of computer-generated art. Due to unavailability of the hardware and the lack of expertise with computers, the first computational artworks were developed by only a handful of artists with scientific background. Pioneers of computer graphics like Georg Nees [7] or Frieder Nake [6], for example, used axial plotters to create the first artworks that were constructed digitally and then drawn by the plotter. As the technology evolved so did the possibilities of integrating aesthetics in computer applications. Nowadays computational aesthetics plays a significant role in computer-generated animations, games, and modern computational art. In this section we outline several illustrative examples in these areas.

Computational Art and Image Stylization
Computational art has experienced an immense evolution since its introduction in 1962. Recent developments in non-photorealistic rendering and computer-based image stylization are able to convincingly mimic human artists. An extensive overview of artistic stylization techniques was provided by Kyprianidis et al. [128]. A common strategy

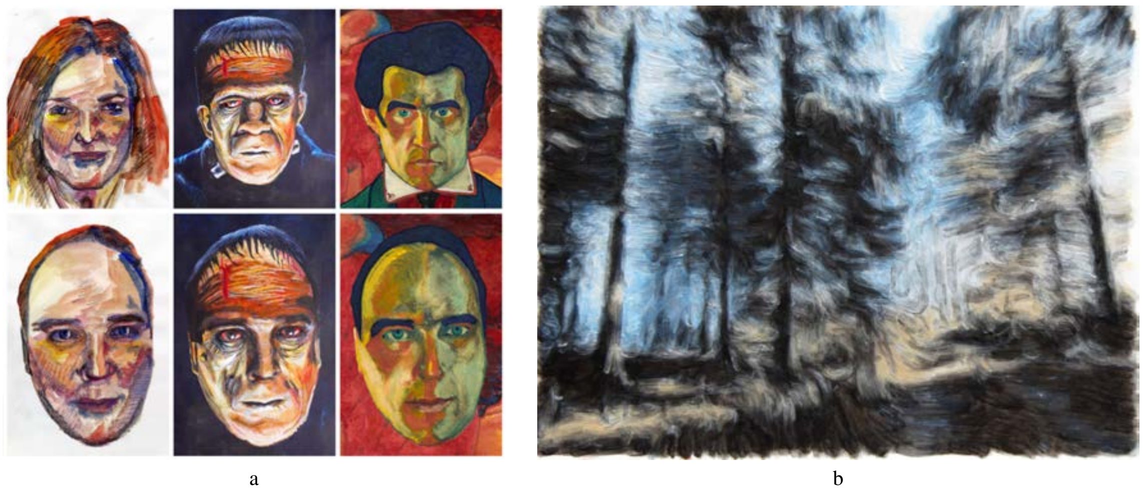


Figure 2.5: Two examples of publications with a strong focus on aesthetics. (a) Example-Based Synthesis of Stylized Facial Animations. Upper row shows exemplar styles that were applied to the target on the lower row; courtesy of Friser et al. [68], (b) Sunbeam In Forest, an artwork by E-David; courtesy of Deussen [53] and Lindemeier [137]

to generate stylized images is to use stroke-based rendering such as brush stroke techniques. Normally, these techniques use low-level abstractions based on local image properties. Wen et al. [229] followed this strategy and presented an approach for color sketch generation through a segmentation algorithm. Some approaches tried to guide the stroke placement through higher-level parameters. Shugrina et al. [191], for example, evaluated the emotional state of the viewer in real time and proposed "empathic painting" that adapts its appearance based on the viewer's emotional state. Colton et al. [48] proposed a similar approach and developed a stroke-based rendering algorithm which heightens the emotions of the depicted person. Some approaches try to mimic existing styles by using examples as input. Friser et al. [67], for instance, used hand drawn spheres as a basis for stylization of 3D renderings. Selim et al. [186] used neural networks for transferring a painting style from one head portrait to another. Friser et al. [68] presented an approach for example-based face stylization that is able to recreate the exemplar style up to pixel precision. Three examples of the style transfer are shown in Figure 2.5 (a). Durand et al. [58] introduced an interactive system that supports the user in the creation of drawings from photographs by taking input images and performing semi-automatic tonal modeling from the exemplar.

Physical Fabrications and Halftone Techniques

Early machine-generated drawings were mostly of abstract nature. Over the last years, artists began to develop means for more realistic machine-generated artwork. Pindar Van Arman [11] is constantly improving AI-controlled painting robots, collectively referred to as *cloudpainter*, with the goal of achieveing original compositions. Tresset and Fol Leymarie [214] described *Paul the Robot*, a robotic installation for face drawings. A more general setup was achieved with *e-David*, a feedback-guided painting robot, whose rendering techniques were discussed by Deussen et al. [53] and Lindemeier et al. [137]. We show a physical drawing created by e-David in Figure 2.5 (b). The project *Caravaggio* [3] by Michele Della Ciana aimed to create halftone drawings consisting of a single line. Many approaches presented in the past went beyond common drawing machines and explored rather unusual media. Galea et al. [75] discussed a stippling method with flying quadrotor drones. Jain et al. [99] presented a force-controlled robot that was not limited to flat surfaces. Jun et al. [112] used a humanoid robot to create drawings on a wall. Calinon et al. [41] used a humanoid robot to draw artistic portraits for entertainment purposes. Prévost et al.[171] guided the user in creation of large-scale paintings with spray paint by pointing out the differences between the painting and the target image.

Many techniques of image abstraction employ monochrome styles such as halftoning, stippling, or hatching. Pnueli and Bruckstein [170] were among the first to discuss gridless halftoning. Ahmed et al. investigated line-based halftoning techniques and suggested several techniques including amplitude modulation [20], recursive division [18], and an approach to address the general brightness/contrast problem of line-based halftoning methods, by using error diffusion as preprocessing step [18]. Pang et al. [164] aimed to preserve the structure and tone similarity between the original and the input image and suggested a structure-aware halftoning technique. Later, Chang et al. [43] used an error-diffusion approach to create visually similar results to the method by Pang et al., but with significantly decreased computation time. Hiller et al. [88] generalized stippling techniques by considering primitives other than points. Pedersen and Singh [166] integrated a maze semantic in the tonal abstraction and presented an approach for the synthesis of organic maze structures. Similarly, Xu and Kaplan [239] discussed a set of algorithms for designing maze structures based on images. Kaplan and Besch [114] described a single line halftone drawing technique that employs the Traveling Salesman Problem (TSP) algorithm. Chiu et al. [45] presented a tone- and feature-aware circular scribble halftone style by extending the TSP algorithm. Bartesaghi et al. [30] introduced a hatching style for three-dimensional shapes based on multiple images with fixed positions and angles to the camera. Son et al. [192] proposed feature-oriented structure grids for stippling styles, where the flow of primitives is parallel to the feature boundaries in the image. Using paper as a two-tone abstraction medium, Xu et al. [240] presented a technique for procedurally-generated paper cut designs with guaranteed connectivity. The work of Li et al. [136] proposed an algorithm for the automatic creation of popup illustrations from 3D models.

Aesthetics in Entertainment Media

As mentioned earlier, light is an important factor in the perception of geometry. In the cinematic context, lighting is often used to convey certain moods or emotions. An overview of different lighting setups and their effects on the viewer can be found in the book Advanced RenderMan by Apodaca and Gritz [27]. De Melo et al. [52] proposed a model for expressions in virtual humans with a composition of lights, shadows, and chromatic filters. Wisessing et al. [232] evaluated how viewers perceive animated characters under different lighting setups. To accommodate variations of dramatic scenes Nasr et al. [60] presented a lighting system that automatically adjusts to support the mood. Dvoroznak et al. [59] eased the manual process of creating 2D animation by introducing an example-based approach for synthesizing hand-colored cartoon animations that translated a given skeleton motion to a new target. Noris et al. [157] presented a technique to easily control temporal noise present in sketchy animations. Using a combination of Monte Carlo integration and neural networks, Kallweit et al. [113]

developed a technique for efficient image synthesis of atmospheric clouds. Taylor et al. [210] employed deep learning to automatically generate natural looking speech animation that synchronizes to input speech.

2

«If we hit that bulls eye, the rest of the dominoes should fall like a house of cards.
Checkmate.»
Zapp Brannigan

Chapter 3

Contributions

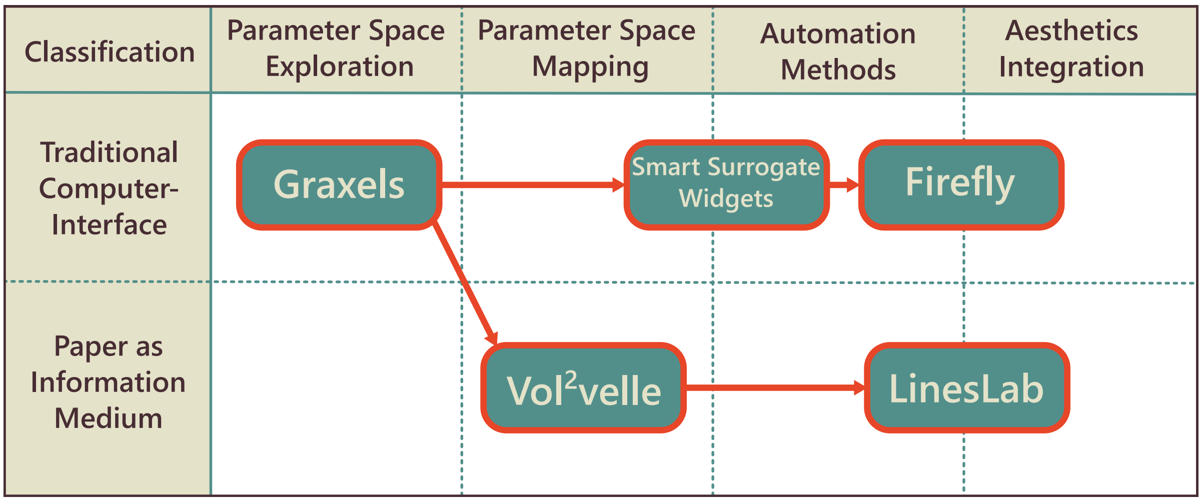


Figure 3.1: Overview of the relationships among the individual contributions within the context of parameter space strategies.

This thesis presents a general foundation for strategies to cope with complex parameter spaces, where the individual contributions of this thesis can be seen as part of a continuous progress of parameter space techniques towards fully automated methods. In Figure 3.1, we illustrate how the individual contributions relate to the three strategies of parameter space exploration, parameter space reduction and methods for automated parameter tuning. The contributions accomplished in this thesis started with interactive exploration of parameter spaces, followed by two approaches for parameter space reduction and simplification for interactive scenarios. Finally, the last two contributions on the right side of Figure 3.1 present methods for automated parameter tuning. In addition, the presented work branches out into two categories. The upper branch covers traditional computer interfaces, while the lower branch accounts for a physical medium, namely paper. In the remainder of this chapter, we outline the theoretical achievements of this thesis in three detailed sections that cover the three strategies for parameter spaces and aesthetic considerations.

3.1 Exploration of Information Rich Parameter Spaces

The term information explosion was first used in a New York Times article by Walter Sullivan in 1964 [205], but since the advent of the internet, it has affected almost all aspects of our life. Technological advances in medicine enable us to obtain patient data cheaper and more detailed than ever before. Today, medical scans can produce thousands of images and measurements for a single patient in seconds. Wearable technology is able to uninterruptedly collect patient data and cohort studies acquire data of thousands of patients, but identifying the relevant parts in this growing data is an ever-increasing challenge.

To extract useful information from the data, doctors need efficient methods and effective analysis tools. While automatic techniques become increasingly precise, practitioners tend to mistrust the automatic results and prefer explorative methods. Since doctors cannot afford to spend extended time periods on one patient, the proposed solutions in medicine are highly specialized and result-driven to be as efficient as possible for the specific tasks. However, the design of a suitable solution is a challenging task. Many pathologies, for example, require a combination of volumetric and temporal measurements to be correctly classified. This time-dependent volumetric data has significant applications in areas reaching beyond medicine, such as climatology and engineering. However, the concurrent evaluation of the spatial and temporal features proves to be very challenging. Common visualization techniques either aggregate the data in the temporal dimension to a single volume, visualize the temporal behavior in an animated view, or allow users to define a point or a region of interest and display the temporal development for the selection. These approaches might not be suitable for some application scenarios. For example, in dynamic contrast-enhanced MRI (DCE-MRI), the shape of a time-intensity curve in a particular region in space can provide important information about the malignancy of a tumor. Evaluating the time-intensity curve from an animation or an aggregated volume rendering is nearly impossible and defining regions of interest might result in overlooking important features. To address this challenge, we propose a visualization approach that provides a comprehensive overview of temporal relations directly embedded in a spatial context. The first contribution of this thesis is the introduction of a new information-rich visualization primitive, which we call graxels (graph pixels), for interactive visual analysis of time-varying volume data.

Graxels are inspired by the concept of small multiples. Tufte [215] described small multiples as "shrunken, high-density graphics", which are "drawn almost entirely with data-ink". Tufte believes that small multiples are extremely well-suited for data comparison. He writes "Our eyes can make a remarkable number of distinctions within a small area. [...] 100 points in one square centimeter". As such, small multiples provide a dense, but highly efficient approach for comparative visualization. However, small multiples are commonly used for abstract data without any spatial context. Our contribution was to extend the concept of small multiples to be directly integrated into the spatial domain. Our proposed graxels provide a link between the spatial and temporal domain, as they integrate both aspects in one primitive.

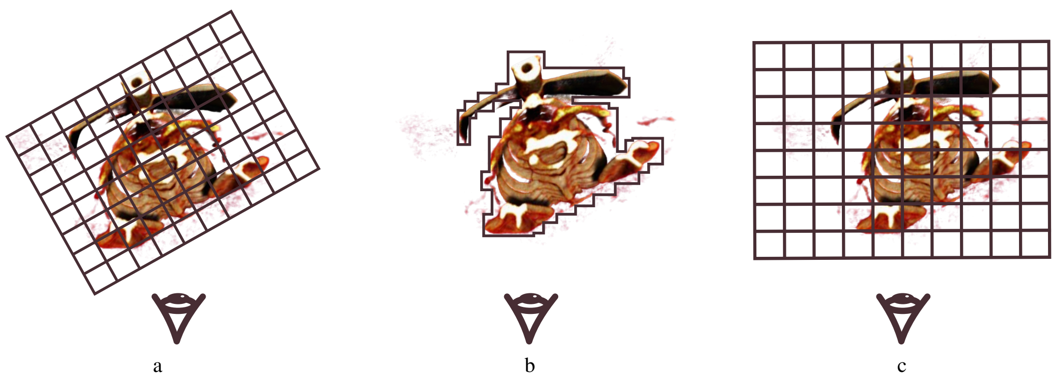


Figure 3.2: Three possible strategies for spatial data aggregation. (a) Uniform aggregation in object space can lead to arbitrary cuts in the data. (b) Clustering-based aggregation depends highly on the clustering algorithm. Furthermore, the resulting cluster might be badly suited for a canvas. (c) Uniform aggregation in image space can lead to arbitrary cuts as well, but the volume position can be easily adjusted to an appropriate subdivision.

3.1.1 Concepts of Data Aggregation

In order to visualize four-dimensional data on a two-dimensional screen, the data needs to be abstracted or aggregated. There exist several possibilities to aggregate data for the graxels. A common straight-forward method is to aggregate data in object space. Here, two conceptually different approaches arise: Aggregation over uniform volume blocks (Figure 3.2 (a)) or clustering the volume based on a similarity measure (Figure 3.2 (b)). The first approach has the disadvantage of being too rigid. Uniform subdivision according to the volume bounding box may lead to arbitrary cuts in the volume, as can be seen in Figure 3.2 (a). The second approach seems appealing at first, as graxels can be integrated into each cluster. However, the results would rely heavily on an appropriate metric and the clustering algorithm, which is a non-trivial task. Furthermore, the resulting clusters may have highly irregular shapes, as can be seen in Figure 3.2 (b). Such irregular shapes would be badly suited as a canvas for graxels. In addition, while clustering algorithms become more and more powerful, they still impose a high computational load on the process.

Our concept of graxels is inspired by uniform volume subdivision, but is conceptually a user centered approach. We avoid the rigid subdivision by performing the data aggregation outside the object space. Instead, we aggregate the data in image space, as can be seen in Figure 3.2 (c). The data aggregation for graxels is conceptually related to conventional volume ray casting. However, as the graxels cover a larger screen area than just one pixel, the aggregation is performed not through ray casting but through tile casting for each Graxel. In addition, the casted tiles are further subdivided into a user-defined number of slabs that control the degree of aggregation in depth. For each subdivision, we store the maximal, minimal, and the average intensity value for each time step. This approach allows for flexible and dynamic adjustment of the aggregation in screen space and volume depth. The concept of a graxel is very similar to a sensor with a fixed resolution. To increase the level of detail, the sensor can be moved closer to the region of interest. This is analogous to zooming into the data with the camera. If the camera orientation is sub-optimal, one can rotate or translate the data to find a better orientation, and the graxel sensor will immediately start recording the data from the new position. This allows for intuitive change of the aggregation domain, through familiar operations such as zooming, translation, and rotation of the volume.

3.1.2 Graxel Computation

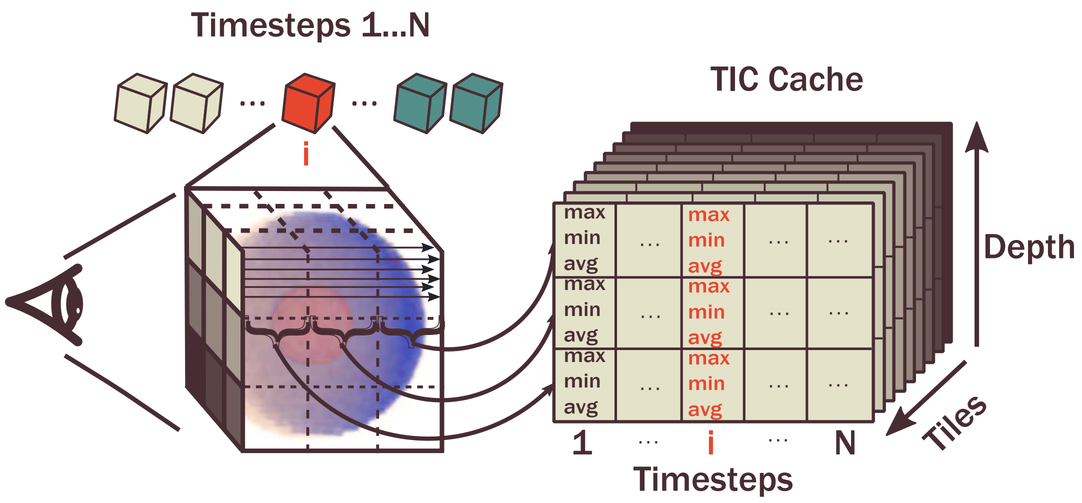


Figure 3.3: Overview of the graxel computation. Tile casting is conceptually similar to conventional ray casting as it evaluates and aggregates the volume covered by the tile. In contrast to ray casting, each tile is uniformly subdivided into slabs. For each tile and each slab we compute the maximal, minimal, and average value within the compartment, creating an envelope function of the data. These values are stored in a time-intensity curve (TIC) cache for each time step.

An important requirement for the exploration of time-dependent volume data was the necessity of being able to process streaming data as well as recorded data. Therefore, the proposed method must have interactive performance. To achieve this, we have chosen an incremental approach that processes the data as it arrives. A key concept for graxel rendering is tile casting, which can be summarized as follows: Each incoming volume undergoes the traditional volume rendering pipeline. In addition to the conventional volume rendering, we perform further aggregation for each tile. In order to reduce the memory overhead and to allow for interactive computation, the tiles are evenly subdivided into a user defined number of slabs. For each slab, we compute the maximal, minimal, and average value for each time step and store these values in a cache. We illustrate this process in Figure 3.3. The number of slabs can be changed at any point. The result of the aggregation is an envelope function consisting of the maximal, minimal, and average value of the time-intensity curve for each compartment. These three values have shown to be well-suited for characterization of time-intensity curves within aggregated regions.

3.1.3 Exploration through Interaction

As can be seen in Figure 3.2 (c), graxels consists of view-aligned small multiples depicting the time-intensity curve of their covered volume. Directly embedded in the

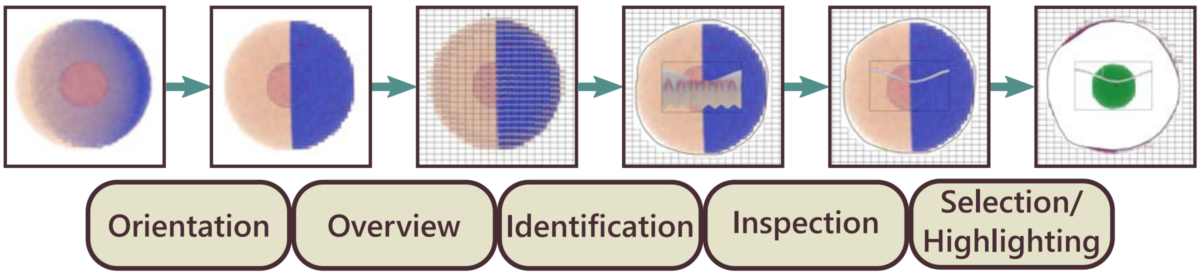


Figure 3.4: The exploration interaction design with graxels starts with an *orientation* phase, followed by an *overview* phase of the temporal development. After the overview, regions of interest are *identified* and the spatial and temporal space are closely *inspected*. Finally, we allow *selection* and *highlighting* within the spatial and temporal domain.

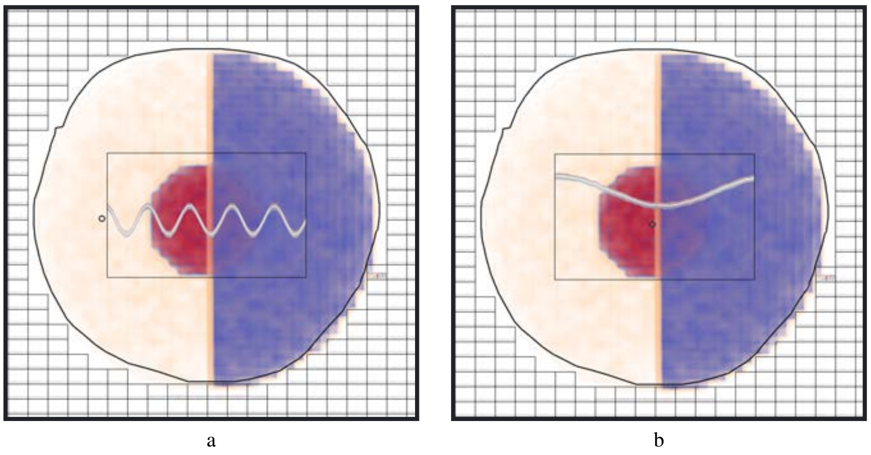


Figure 3.5: A close up of the graxel interface during the inspection phase. In image (a) the time-intensity curves of the left sphere half are inspected, in image (b) the user inspects a smaller sphere inside the volume.

spatial visualization of the data set, graxels provide a direct relation between the spatial and temporal structures. Because the graxels are conceptually embedded in the temporal as well as the spatial domain, we provide exploration and selection methods in both domains. Parameter space exploration with graxels can be conceptualized in a five-step drill-down exploration. We illustrate the five interaction steps in Figure 3.4. The first step of the interaction is *orientation*. During the orientation phase, the graxels are rendered transparently and the user interacts with the visualization exactly as during conventional volume visualization. The orientation phase allows the user to explore the spatial aspect of the data, and to find an interesting viewpoint through common interaction techniques, such as rotation, zoom, or translation. After a suitable view is found the graxels start to fade in, announcing the *overview* phase. During the overview phase, the time-intensity curves can be inspected for the whole volume. To change the level of detail, the user can group the slabs to create a denser data aggregation or show individual slabs and browse through them. This way regions of interest can be *identified* in screen space and depth. The user is able to group the graxels of the identified region of interest (ROI) with a lasso tool. All graxels caught by the lasso will be displayed in a single super graxel within the region of interest. When the region of interest is identified, the *inspection* phase begins. During the *inspection* phase, the user can explore the spatial and temporal characteristics of the ROI in more detail. Lastly, we allow the user to *select* and *highlight* the data in the temporal and spatial domain during the last interaction phase. The described process can be restarted at any point. When the user interacts with the volume, for example by rotating the view, the graxels immediately fade out in order to not distract the volume visualization. As soon as a new suitable view is found and the volume interaction stops, the graxels start to fade in again.

One of the key aspects of our approach is that it allows us to seamlessly explore within the spatial and temporal domain. Based on the feedback we received from domain experts in the medical field, we set several goals for the graxel interaction design. One of the main requirements was the ability to select and extract temporal features. Another requirement was the highlighting of temporal events with direct links to the spatial location. Lastly, the domain experts wished for grouping and separation mechanisms for the interactive management of different classes of temporal and spatial features. To achieve these goals, we focus mainly on two interaction tools. Selection within the spatial domain is performed with the *focus lasso* tool. Further interactions within the temporal domain of the time-intensity curves are performed with the *curve selector*. In the following we briefly outline the interaction with the *focus lasso* and the *curve selector*.

Focus Lasso

The focus lasso is used after the user has identified one or multiple regions of interest. The user can group graxels by simply drawing an outline on the screen. All graxels that are enclosed by the lasso outline are grouped and displayed in one single canvas. Two examples of a grouped graxel cluster can be seen in Figure 3.5. If it is desired to compare two or more regions, multiple selections with the focus lasso are possible. If a selection with a focus lasso is smaller than the original graxel size, it is ignored as it would show less details than the initial representation.

Typically, focus lassos contain more than one time-intensity curve. Displaying the time-intensity curves in the same manner as for the initial graxels might lead to reduced readability. To provide a simple but effective exploration tool we use two strategies for the graxel rendering inside the focus lasso. Firstly, we reduce the opacity of the shown time-intensity curves within the focus lasso. This simple strategy creates visual clusters of similar time-intensity curves, providing an overview of the main trend within the focus lasso. Secondly, we allow for close inspection of the temporal domain within the focus lasso through a point of interest. The point of interest is moved by the user with the mouse within the focus lasso region. Time-intensity curves closer to the point of interest are displayed more opaquely, while time-intensity curves further away are becoming more and more transparent. The slope of the opacity decrease can be modified by the user to change the degree of detail within the lasso.

Curve Selector

As mentioned before, one of the requirements by the medical domain experts was the selection within the spatial and temporal domain. The focus lasso already provides a coarse grouping within the spatial domain and the point of interest allows for fine-scale filtering in the spatial domain. With the curve selector, we provide a mechanism for fine-scale selections in the temporal domain, i.e. the value domain of the time-intensity curves. We allow selections in the value domain through direct selection of time-intensity values with a rectangle or a circle selection tool in the graxel cluster of the focus region. If a time-intensity curve crosses the selected area, it is extracted and shown in a separate window. This allows for a close comparison of several time-intensity curve families.

In some application scenarios, the practitioners are interested in the spatial location of data with certain temporal properties. To account for this need, the user can highlight the volume data via selections in the temporal domain. The rectangle and circle tools for curve extraction can be changed in functionality to be used as a highlighting mechanism. When marking time-intensity curves with the highlighting functionality, all volume points that lie within the range of the selected values are displayed in an alternative transferfunction. This second transferfunction can be applied to the region of the current focus lasso, all focus lassos, or the whole volume. For a detailed inspection of the shape and the exact spatial position of the selected data, we provide the ability of rendering non-selected data as completely transparent. Using these two simple tools, we enable the simultaneous exploration within the spatial and temporal domain in a uniform visual representation.

3.2 Interaction Design in Reduced Parameter Spaces

One of the goals of the research conducted within the scope of this thesis was to support visualization applications aimed towards non-experts. Such application scenarios require either simple intuitive user interfaces or automated methods that solve complex tasks for the user. In this section, we focus on simplified user interfaces. More specifically, we discuss design considerations that arise when rich interaction spaces must be mapped to simplified or "reduced" parameter spaces. As outlined in the beginning of Section 3, the contributions of this thesis branched into two categories considering the medium of information, traditional computer interfaces and physical paper-based interfaces. In the remainder of this section, we will discuss the interaction design for these two media separately, as they require conceptually different design choices. We will start with paper as information medium and then proceed with the concept of surrogate interaction for traditional or touch-screen interfaces.

3.2.1 Paper as Interaction Medium

Before discussing the concepts of interaction with paper as information carrier, it is worthwhile to explain why we have chosen paper as an interactive visualization medium in the first place. At first glance, paper seems to be badly suited for interactive visualization. However, interactive information displays with paper have been widely used long before the introduction of computers in the form of Volvelles. Volvelles typically come in the form of a wheel chart or a slide chart. These paper constructions with movable parts can be seen as primitive analog computers. The first Volvelles can be traced back as far as 1000 BC to the Persian astronomer Abu Rayhan Biruni who contributed to astronomy, mathematics, and related subjects like mathematical geography [180]. Over the centuries, Volvelles have been carefully planned and crafted to accommodate scientific visualizations and computations in various disciplines, such as

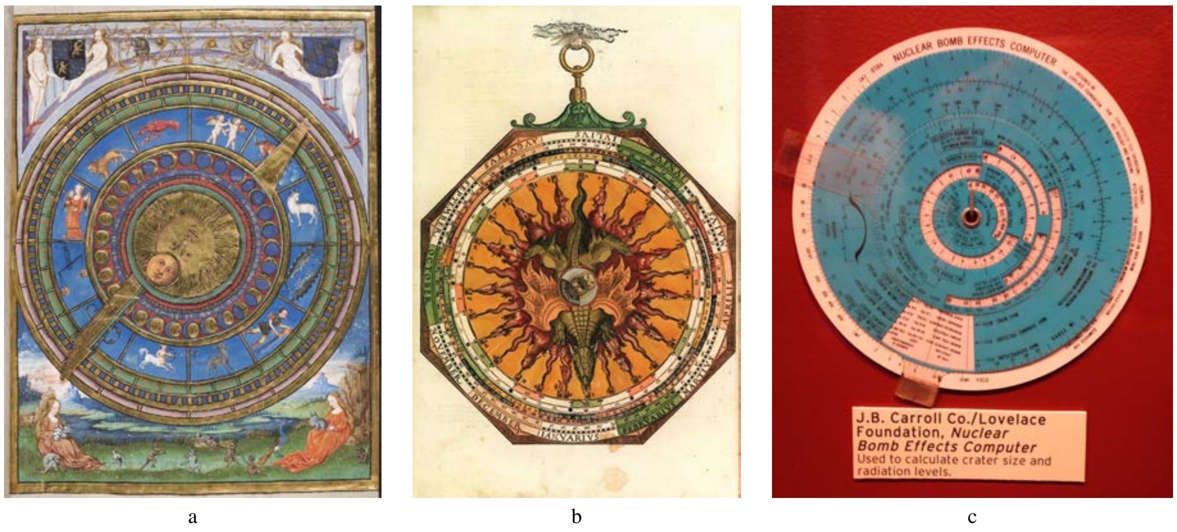

a
b
c
NUCLEAR BOMB EFFECTS COMPUTER
J.B. Carroll Co./Lovelace Foundation, Nuclear Bomb Effects Computer
Used to calculate crater size and radiation levels.

Figure 3.6: Three examples of historic Volvelles. (a) A page from Geomantie' (Geomancy) - Codex Palatinus 833 Germanicus, which was used as religious almanac and prediction calendar. The shown page depicts a solar and lunar calendar. (b) Astronomicum Caesareum, another calendar Volvelle used to predict the eclipse of the sun and the moon. (c) Nuclear Bombs Effects Computer. This "computer" chart was sold during the Cold War, as the public became interested in actual consequences of nuclear weapons; courtesy of MIT museum.

astronomy, geography, or physics. In Figure 3.6, we show three Volvelle examples ranging from the 16th century to a relatively modern Volvelle from the Cold War era.

Clearly, we do not claim that interactive wheel charts can compete with the utility provided by computers or hand held devices such as tablets or smart phones. However, paper as information medium has many advantages. Paper charts can be produced very cheaply, they are recyclable, and easily accessible. Furthermore, the direct interaction with the material has additional benefits. For example, it has been shown that direct interaction with the visualization medium increases the likelihood of remembering the encoded content [87, 203]. Direct interaction leads to increased user engagement and can faciliate a better understanding of the data [202]. If the user is involved in the assembly process of the Volvelle, an additional benefit arises from the IKEA effect [158], creating a more engaging and memorable experience for the user.

Considering these benefits, Volvelles seem to represent a cost-friendly alternative for personalized visualizations. However, the design and creation of a Volvelle is a tedious and time consuming process. Depending on the complexity of the Volvelle, the estimated production time for a Volvelle ranges from several weeks to months. Our contribution is a flexible approach that allows to close the fabrication gap for the generation of Volvelles for volume visualization. We propose an approach that maps the interaction space of interactive volume visualization setups to tangible paper-based visualizations that can be printed on a conventional printer and still preserve a user-specified degree of interactivity. We call our approach volume Volvelle or *Vol*^{2}*velle*. The printable visualization generated through our approach can be used in many different scenarios. One can imagine how these printouts can be used as instantly generated personalized visualizations for public displays, such as museums, medical education, or marketing activities, where a paper-based visualization can be used as an eye-catcher

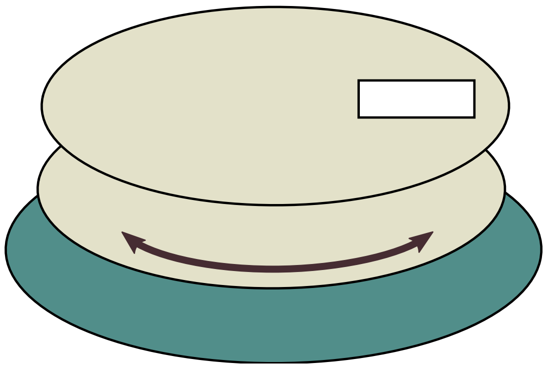


Figure 3.7: An illustration of the simplest Volvelle type consisting of a fixed front cover, a rotating wheel chart, and a fixed base.

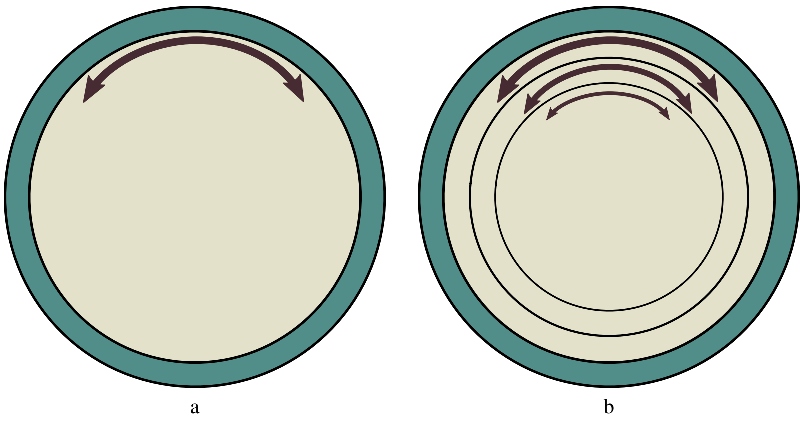


Figure 3.8: The Volvelle can have one rotating wheel (a) or multiple stacked wheels, that can be rotated independently (b).

to distribute the URL to a fully interactive web-based visualization. Before we can explain the interaction mapping of an interactive volume visualization to a Vol^{2}velle, we need to discuss the interaction space of paper-based visualizations first.

3.2.2 Interaction-Space of a Wheel-Chart

In this section, we discuss the general interaction model for interactive paper-based visualizations. We contribute an analysis of traditional Volvelle interactions as well as novel interaction methods that, to the best of our knowledge, have not been used for paper-based visualizations before.

The most common and simple Volvelle consists traditionally of three layers: A fixed cover with a window, a rotating wheel, and a fixed base. This basic layout is shown in Figure 3.7. The third base layer has historic reasons and is not necessary. Volvelles were often integrated into manuscripts and therefore by default attached to a fixed basis of the manuscript. Modern stand-alone Volvelles often consist only of a cover and a rotating wheel. While most Volvelles employ one rotating wheel, as shown in 3.8 (a), the amount of wheels can vary. Theoretically, each additional wheel introduces an additional degree of freedom in the Volvelle interaction space. In practice, however, the number of wheels is limited due to space constraints. Furthermore, stacking up

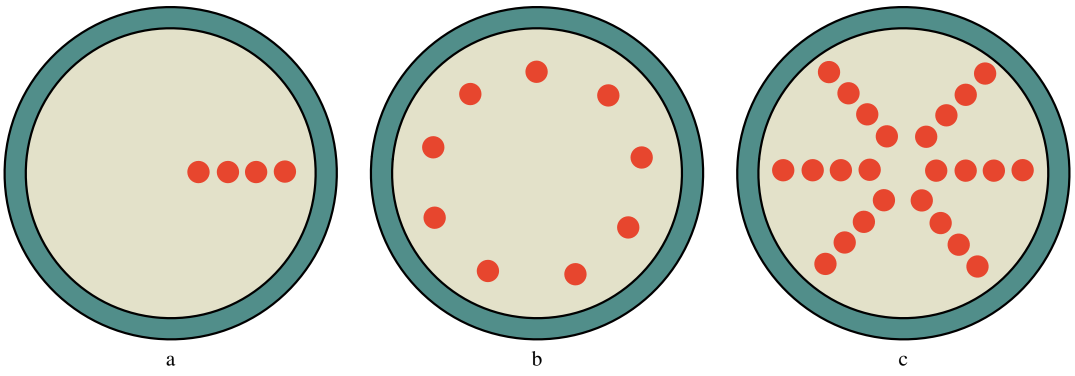


Figure 3.9: The information layout on each wheel can be either concentric (a) or radial (b). The two layout strategies are not mutually exclusive and can be combined to a layout as shown in image (c).

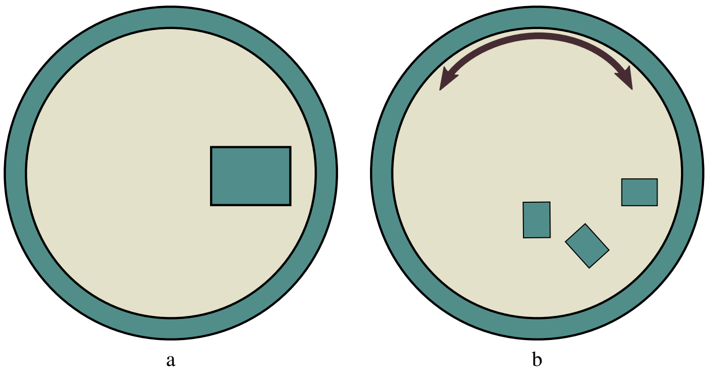


Figure 3.10: Two possible design choices for a Volvelle cover: (a) a single fixed cover or (b) a second rotating cover wheel with multiple windows for the radial layout.

wheels introduces additional complexity, both in terms of assembly and with respect to the practical interaction, as shown in Figure 3.8 (b). Using transparent media instead of paper can partially solve the space requirement, but the increased complexity remains an obstacle. Additionally, the stacked wheels increase the friction of the whole Volvelle assembly, which hinders the interaction process.

The rotating wheels are commonly used as the main information carrier as they are often the only interactive parts of the Volvelle. The information on the rotating wheels can be distributed in two layouts: concentrically (see Figure 3.9 (a)) or radially (see Figure 3.9 (b)). These two layouts are not mutually exclusive and can be combined as shown in Figure 3.9 (c).

Using a concentric layout imposes additional complexity as the Volvelle assembly requires a mechanism to move the cover window radially. This can be achieved by a second rotating cover wheel with multiple windows and a base cover with an elongated window. Therefore, two possible designs arise for the Volvelle cover. The first design is a static cover with one or multiple fixed windows for a radially-aligned, one-dimensional layout. The second possible design is a combination of a fixed cover with an elongated window and a second rotating cover wheel with multiple windows for the radial layout. These two window designs are shown in Figure 3.10 (a) and (b).

While the circular Volvelle is the most common mechanism for paper-based in-

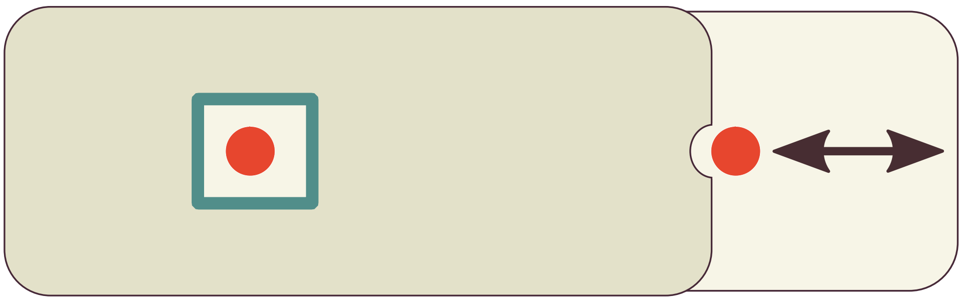


Figure 3.11: Interactive charts do not require circular layouts. A slide chart is able to encode one-dimensional interaction with an envelope-like cover and an information chart that can be pushed through.

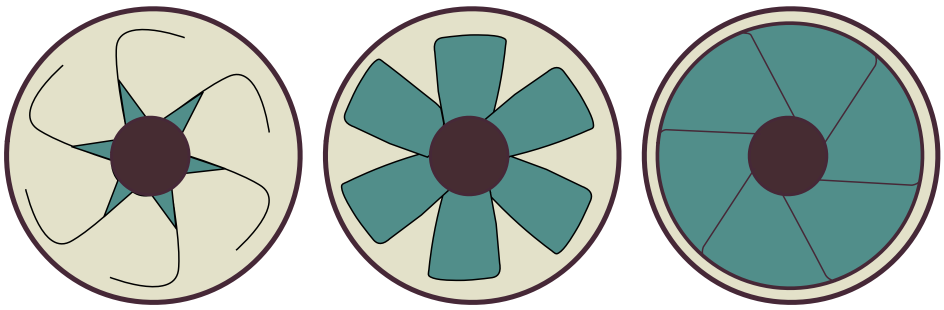


Figure 3.12: A more complex wheel can be constructed for the Volvelle by interleaving two wheels to an iris mechanism.

teractive charts, there are other possible constructions as well. A linear slide chart, consisting of an envelope-like cover and a rectangular paper chart, is one of such constructions. We show a slide chart in Figure 3.11, where the rectangular information sheet can be pushed along the envelope.

While the presented examples have been used for interactive wheel charts before, we further introduce constructions that have not been used for interactive wheel charts. One of such mechanisms presents itself as a rather complex Volvelle wheel with an iris mechanism. We show such a wheel in Figure 3.12. An iris wheel consists of two separate wheels that are interleaved in a way that rotating the two wheels against one another controls which wheel is visible. Such a wheel can be then used as a binary switch to change the encoding between two states.

As indicated earlier, Volvelles are not exclusively restricted to paper. In our approach we allow the use of transparent media to introduce additional degrees of freedom in the interaction space. Our Vol^{2}velle model for transparent media incorporates rotating wheels made from transparent film suitable for printers. Such a transparent film is particularly useful in the context of illustrative volume visualization, as a transparent model can be used to overlay multiple independent layers. The final image is then composed similarly to the multiplane camera used in early animation movies. We show a possible assembly of transparent layers in Figure 3.13. Since the order of the wheels is fixed, the visualized volume parts need to be aligned in the same spatial order if correct occlusion is desired. There are various additional possibilities for mechanical construction of interactive physical information charts, but we consider the described options to be the best fitting for our purposes.

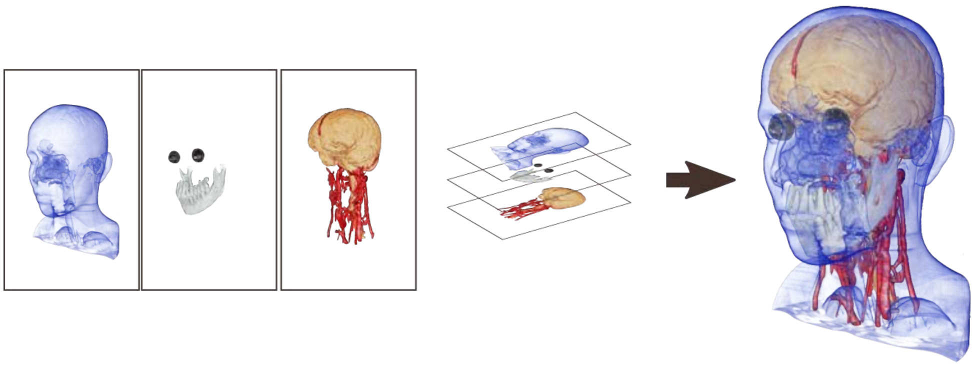


Figure 3.13: An example of a possible assembly with a transparent medium. The final image is composed of three separate layers from an illustrative volume visualization.

Now that the possible mechanisms for a Vol^{2}velle are described, we can proceed with our main contribution for physical visualizations, namely a framework to map interactions from conventional volume visualizations to the limited space of a Vol^{2}velle. In order to formulate the mapping between these two spaces, we need to define the interaction spaces first. So far we we have outlined the basic components of a Vol^{2}velle. Now we evaluate how these concepts can be used to formulate an interaction space for a discretized function of multiple scalar parameters. In our approach we allow the user to choose between a handful of Vol^{2}velle models. In principle, we distinguish between a paper model and a transparent model. The paper model employs, as the name suggest, exclusively paper as the information medium, while the transparent model uses transparent film as material for the individual wheels. For the paper model, we allow to encode one or two-dimensional parameter spaces directly on the wheel through the radial or concentric layout. These two layouts are conceptually different. While the radial layout allows a cyclic parameter space, the concentric layout can only encode sequential parameters. The transparent Vol^{2}velle model allows the use of up to three transparent wheel charts. We found that using more than three wheels made the Vol^{2}velle assembly and the interaction too cumbersome for most users. The three transparent wheels, similar to the paper model, can each encode two parameters in the radial and concentric layouts.

In addition to these two models we allow a slide chart extension for both models. The slide chart can encode one additional independent parameter. We employ the slide chart to display an additional attribute of the data that can be linked to the visualization encoded on the wheel chart. A natural choice for the slide chart is an additional slice view visualization. However, we do not restrict our system to this modality and the slide chart can be used for any visualization that can be encoded with one parameter.

These models allow the Vol^{2}velle to encode all three tabular data types: categorical, ordinal, and quantitative data. Because ordinal and quantitative data can be treated similarly, we refer to both categories as ordered in the remainder of this thesis. Thus, the parameter space of the Vol^{2}velle can be summarized as follows:

- Paper model
  - 1 Ordered Cyclic Parameter (radial layout)
  - 1 Ordered Sequential Parameter (concentric layout)
  - 1 Categorical Parameter (changing picture wheel)
- Transparent model
  - 3 Ordered Cyclic Parameters (radial layout)
  - 3 Ordered Sequential Parameters (concentric layout)
- Slide chart extension
  - 1 Ordered Sequential Parameter (slide chart)

We have integrated the Vol^{2}velle generation into an existing volume visualization framework. In this framework, the whole parameter space is expressed through a set of properties which can be modified interactively, e.g., through widget interfaces, mouse and keyboard commands, or scripting commands. Any of such modification can trigger a change of the displayed visualization and we can consider the resulting visualization images as individual samples of the visualization parameter space. Therefore, the generation of the Vol^{2}velle involves a mapping from the Vol^{2}velle interaction space to a corresponding sampled subset of the visualization image space. For simplicity, we refer to the digital visualization space as V and the Vol^{2}velle space as V'. These two interaction spaces are inherently different. Next to obvious differences, such as the dimensionality of the parameter space, the Vol^{2}velle space V' has a trade-off between the image size and the parameter density. Furthermore, the parameter space of the Vol^{2}velle space V' can only contain discrete parameters. Clearly, some fidelity of V will be lost when mapped to the Vol^{2}velle space V'. Formally, we can consider V and V' as functions to the visualization image domain Im and Vol^{2}velle image domain Im'. Constructing a direct mapping V' to Im' is non-trivial. Therefore, we substitute the direct mapping by a combination of simpler and more intuitive mapping operations as follows:

$$\begin{tikzcd} V \arrow[r] & Im \arrow[d, "S"] \\ V' \arrow[u, "F"] \arrow[r] & Im' \end{tikzcd} \quad (3.1)$$

The advantage of this detour over the visualization space V is that the mapping of V to Im is already given. Therefore, the only functions we need to define are the mapping F from V' to V and the sampling S from Im to Im'.

The user is able to define the mappings freely. However, the nature of the Vol^{2}velle model leads to some natural mappings. We provide the user with a set of meaningful functions that are applicable to many visualization scenarios. These simple functions can be used as a starting point and can be modified by the user via an integrated scripting language. For details on the individual mapping functions, we refer to paper B.

The sampling function S maps the image space of the visualization to the image

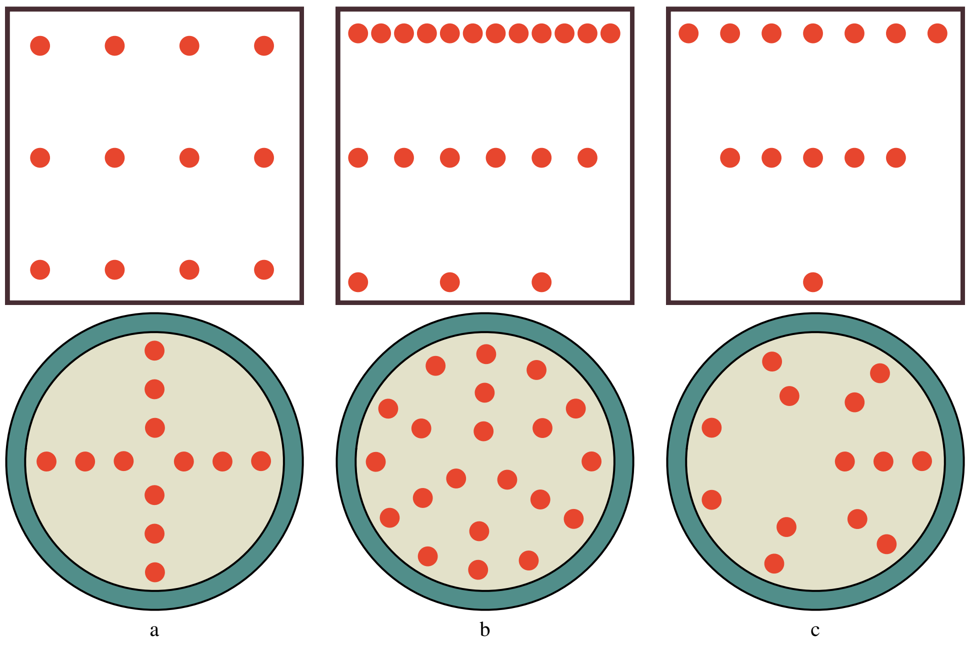


Figure 3.14: Comparison of different sampling strategies in the visualization space vs. the Vol^{2}velle space. (a) A uniform sampling in the visualization space V leads to a sparsely populated sampling on the Vol^{2}velle wheel. (b) On the other hand, a densely packed Vol^{2}velle wheel requires non-uniform sampling in the parameter space. (c) Some parameter constellation, such as trilinear interpolation, are well-suited for a conical layout.

space of the Vol^{2}velle. More precisely, S discretizes the the function F by sampling over the Vol^{2}velle interaction space. The possible layouts on the Vol^{2}velle enforce several constraints onto the sampling process. Intuitively, one would sample the parameters of the visualization space uniformly. However, uniform sampling in all dimensions is badly suited for the Vol^{2}velle geometry. The reason for this is that the Vol^{2}velle wheel provides more space with distance from the center. A uniform sampling would simply waste this space as can be seen in Figure 3.14 (a). To make use of the expanding space, one can place more samples with a growing distance from the center. This leads to a non-uniform sampling strategy as illustrated in Figure 3.14 (b), where two dimensions can be encoded effectively on one wheel. Another possible encoding can be realized with a conical layout as shown in Figure 3.14 (c). In such a layout, each corner of the cone can be interpreted as one of the initial conditions that are to be interpolated, and the samples within the cone can be seen as the interpolated samples. The definitions of the mapping function F and the sampling function S fully describe the Vol^{2}velle.

3.2.3 Interaction Through Surrogate Objects

The main focus of this thesis is the exploration of interaction methods targeted towards non-experts. We have presented an approach for the generation of paper-based visualizations in the previous section. In the following sections, we will focus on simplified interaction design for digital interfaces. There are several reasons why the development of simplified interactions is important. Visualization setups with intuitive interaction are a very important tool for public education as they are accessible to a broader audience. Furthermore, simplified interactions can be necessary for people with certain disabilities. However, the design of intuitive interaction models is not straight forward. While the interaction space of physical visualization tends to be intuitive because of the low-dimensional interaction space, simplified interaction spaces for virtual displays need to be artificially constrained. There are many examples for intuitive and easy-to-understand interaction models. In this thesis, we follow the paradigm of direct manipulation by Shneiderman [190]. Since the introduction of direct manipulation in 1982, it has substantially influenced the development of user interfaces. The notion of direct manipulation was further refined into a more integrated approach using surrogate objects by Kwon et al. [127].

Inspired by these approaches, we propose Smart Surrogate Widgets for direct volume visualization. Our approach employs visibility-driven surrogate widgets that are automatically constructed and placed within the visualization space. The widgets allow for several interactions which are translated directly to local interactions in the visualization space. The interaction design of the surrogate widgets closely follows the four interaction principles outlined by Shneiderman [190]:

- Continuous representation of the object of interest
- Physical actions instead of complex syntax
- Rapid, incremental, reversible operations whose impact on the object is immediately visible
- Layered or spiral approach to learning that permits usage with minimal knowledge

Following this paradigm, our goal is to provide the users with intuitive yet sufficiently powerful and flexible interaction widgets integrated directly into the visualization space. To integrate the widgets in the visualization space, the system needs to "understand" what the user sees and find an appropriate fit for the widgets. Therefore, we call our surrogate widgets "smart".

When novice users are confronted with volumetric data, they often try to find ways to look inside the data by "opening up" the volume locally. Traditional interaction methods often prove to be unsuited for such tasks due to their uniform structure, and novice users struggle to define a desired operation. In particular, the placement and navigation of such tools provides a challenge to novice users. In such scenarios, the users would benefit if the system "understood" their intention and supported the users in the interaction process by providing them with a set of appropriate operations. Of course, it is difficult to capture such subjective notions. However, to provide a familiar frame set for the user, we can look at how cutaways are used in other media. Frequently, illustrations use simple, symmetric shapes to partially remove occluding structures, allowing the viewer to reconstruct the entire object more easily. Technical illustrations, for example, often use a combination of spherical and cylindrical cutaways to explain complex structures, as was done in the illustration of a Space Shuttle Orbiter in Figure 3.15. Inspired by these intuitive illustrations, we propose the construction of Smart Surrogate Widgets out of simple shapes such as spheres and tapered cylinders. Our

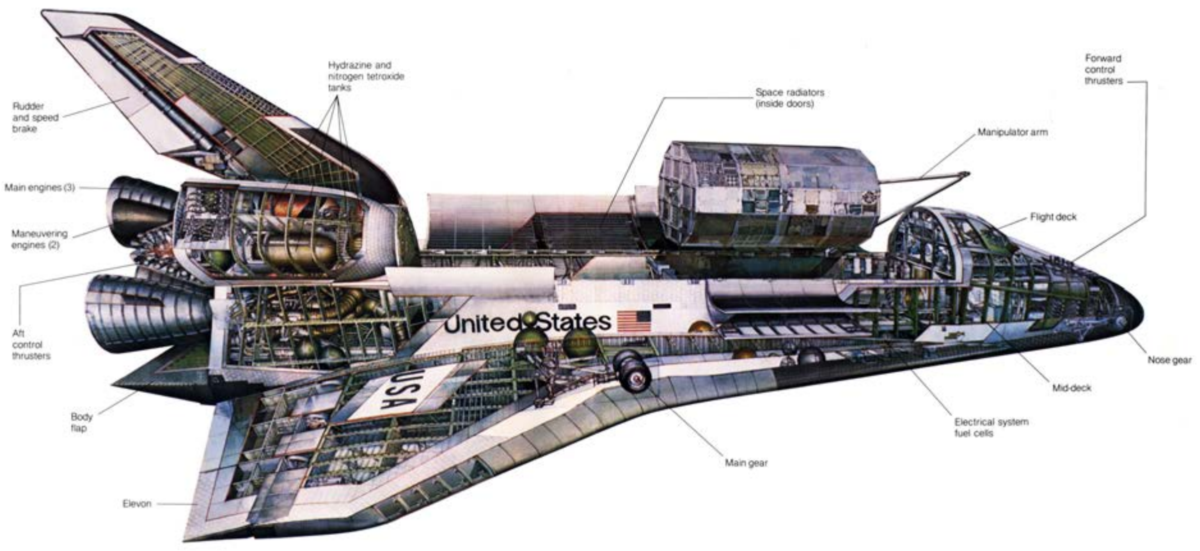


Figure 3.15: A cutaway illustration of a Space Shuttle Orbiter with callouts; courtesy of NASA

approach can be subdivided into two parts: widget construction and interaction design with the widgets.

3.2.4 Smart Surrogate Widgets

The main concept of the smart surrogate objects is the integration of the widget within the visualization. To achieve this, we perform an incremental approach to construct and extend the widgets. As the first step, the visualization presented to the user is analyzed and the visible structures are determined. Using this information, a sphere that approximates the visible structures is fit into the visualization by using the random sample consensus (RANSAC) model [66]. Finally, the sphere is integrated into the existing widget. The user is able to repeat or revert this process to adjust the widget complexity to his desire.

We have designed the Smart Surrogate Widgets to provide means for context-dependent and localized manipulation of the volume data, more specifically we focused on cutting and highlighting operations. We assume that the user adjusts the viewing parameters such as transparency and camera position to focus on features of interest. Our approach automatically evaluates the visible structures and provides interactive surrogate widgets within the visualization. Following the notion of surrogate objects and the direct manipulation paradigm we consciously restrict the interaction space of the smart surrogate objects to a handful of simple and easy-to-interpret operation that still provide a considerable degree of flexibility.

We derive the visual design of the widgets from popular graphical editor software, such as Photoshop. Typically, such software provides two types of interaction models. The first interaction type allows direct manipulation through surrogate objects embedded in the image. These surrogate objects have a minimalist design, consisting only of a frame around the selected object, which can handle drag operations such as deformation or rotation. The second interaction model is realized through control panels that do not have a spatial relationship to the image, such as tonal adjustment. Following this widely used convention, we divide the interactions with the smart surrogate wid-

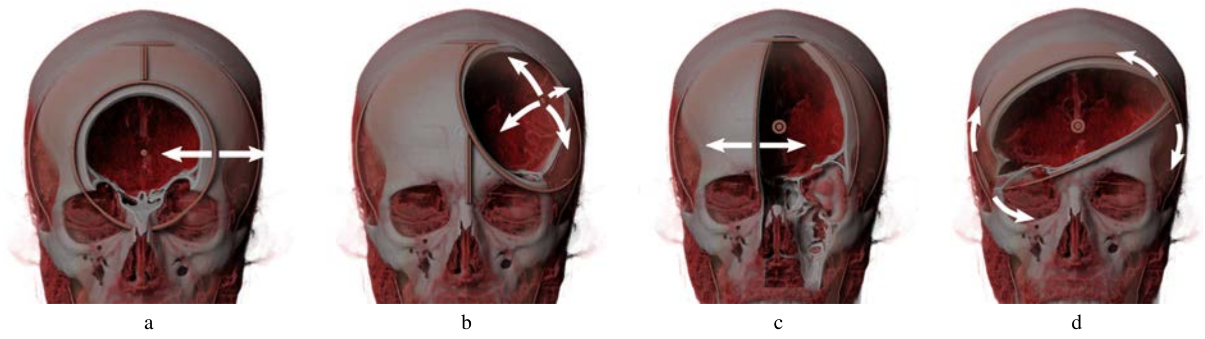


Figure 3.16: Four interaction methods with the smart surrogate widget in the visualization space. Images (a) and (b) show interactions with an iris tool, whereas images (c) and (d) show interactions with a wedge tool.

gets into two categories, direct manipulations within the visualization space through *space widget* and interactions on non-spatial properties through *detached widgets* that are realized as a detached panel located in the upper part of the view port. However, both interaction models follow the paradigm of direct manipulation.

The user is able to directly manipulate the volume rendering by interacting with the space widget. In Figure 3.16 we show a set of possible direct manipulations with the smart surrogate widget. The detached widget has the purpose to control the appearance of the focus region and to allow additional global operation that would be hard to integrate in the visualization space. The detached widget consists of three components. To ensure the third main principle of Shneiderman (rapid, incremental and reversible interaction with immediately visible impact), the middle part contains two buttons, one to add a new sphere to the widget and one to remove the last sphere added to the widget. Additionally, this component has a radio button. If activated it allows the user to move the individual spheres along a plane perpendicular to the viewing direction. However, throughout this thesis we have not used this functionality and show only the automatically computed spheres. The remaining two components control the opacity and the used transferfunction within the smart surrogate widget.

In combination with common interaction methods for volume data, such as rotation, translation and zooming, our smart surrogate widgets provide the user with an intuitive volume manipulation tool that can help creating expressive cutaway visualizations without prior volume segmentation. We do not map the existing parameter space of the visualization to the widgets but instead create a new simplified space for local interactions. The simple widget interface is directly integrated into the volume visualization and allows seamless translation of gestures to volume manipulation. Furthermore, the simple drag functionality of the widgets makes them suitable for touch interfaces as well as traditional computer setups. We present expressive cut-away visualizations created with our method in chapter 4. In the following section we step beyond interactive systems and present fully automated methods for parameter tuning.

3.3 Automated Parameter Tuning and Integration of Aesthetics

The research contributions presented in this thesis have a clear trajectory from explorative towards automated methods. The contributions discussed so far have focused on parameter space exploration and on simplified interaction models with low dimensional parameter spaces enabling the user to interact with the data more easily. We complete the development of simplifying the interaction process by providing fully automated methods for parameter tuning.

Similarly to the contributions on simplified interaction, the automated methods presented in this thesis diverge into two categories of paper-based and digital interfaces. In the remainder of this section, we will discuss the general concept of automated parameter tuning first, and then explain how this general concept was applied to distinct application scenarios.

3.3.1 Parameter Tuning Strategies

Parameter tuning or parameter optimization constitutes its own research field with many diverse methods. The goal of parameter tuning is the selection of parameters, which are optimal in some desired way, for example by minimizing an objective function. In this section, we briefly outline some of the possible optimization strategies and explain the reasoning of why we have chosen a particular strategy for our approach. The most common types of approaches for parameter optimization are:

- Gradient-based approaches
- Evolutionary approaches
- Bayesian approaches
- Grid search approaches
- Random search approaches

Gradient-based approaches are used when it is possible to compute or approximate the gradient of the objective function with respect to the optimized parameters. There are many gradient-based methods, but they are all based on an iterative approach to find a local minimum by "stepping" into the negative direction of the gradient and possibly some correction factor. Gradient-based approaches can be badly suited for problems where the gradient is not well defined or the objective function of the parameter space is highly irregular. Because the object space in our application scenarios was not necessarily smooth, we have decided against gradient-based approaches.

Evolutionary approaches are methods for the global optimization of noisy black box functions. Typically, evolutionary algorithms are designed as an iterative optimization process where an initial population of random solutions is generated and evaluated with respect to its fitness. Then the weakest parameter constellations are deleted and new parameters are generated through crossover or mutation of the best parameter constellations. These steps are repeated until a satisfactory state is reached or the solution is no longer improving. Evolutionary approaches are very powerful and perform generally well on smooth as well as noisy functions. However, evolutionary algorithms tend to be computationally complex and thus not well-suited for scenarios where computation needs to be fast or interactive. As speed was one of the requirements in our applications we have not chosen an evolutionary approach for our solutions.

Bayesian approaches create a probabilistic model of the objective function from an initial parameter set. This probabilistic model is then iteratively validated and updated. This approach is very effective when the object function is not known or is very expensive to evaluate. However, to get an appropriate estimate of the objective function, the Bayesian approach needs a high number of parameter evaluations. This number is further increased if the objective function is noisy. Since we cannot rule out noisy objective functions in our application cases, we have decided against a Bayesian approach.

Grid search approaches are among the oldest and most naive optimization methods. The principle of grid search optimization is simply an exhaustive search through the parameter space or a manually-defined parameter subspace. Grid search methods are often "embarrassingly parallel", meaning there is little or no effort needed to separate the problem into a number of parallel tasks. However, grid search suffers from the so-called "curse of dimensionality", the explosion of the parameter constellations that need to be evaluated. The "curse of dimensionality" and the generally slower performance made us decide against grid search approaches.

Random search approaches are conceptually similar to grid search approaches, but they avoid the exhaustive evaluation of all possible parameter combinations by selecting them randomly. Random search approaches often outperform grid-based approaches, especially when only a a few parameters affect the objective function outcome. In such cases the optimization problem has a low intrinsic dimensionality. Similar to grid search, random search algorithms are often "embarrassingly parallel". The sampling of random search optimization can be steered by including prior knowledge of the problem and specifying the distribution function. If no prior information about the objective function is at hand, common strategies employ random sampling such as the Monte Carlo method. However, is has been shown in several publications [117, 189, 241] that structural sampling methods such as Latin Hypercube sampling can provide an improved convergence rate. Considering the parallelization and the flexibility of random search methods, we employ this strategy with a Latin Hypercube sampling in our solutions.

In the next section, we will outline our contributions on automated parameter tuning for the physical generation of monochrome art, and for an automated approach for scene illumination with dynamic light sources. We discuss LinesLab, a system for the generation of physical monochrome art first, and then focus on Fireflies, virtual illumination drones for interactive visualization.

3.3.2 Parameter Tuning for the Generation of Monochrome Art

The field of computational art emerged shortly after the introduction of computers. The earliest artworks, for example by Georg Nees [7] and Frieder Nake [6], consisted of abstract compositions of geometrical elements. Over the years artists progressed towards more realistic computer-generated artworks. The digital or physical painting of human portraits is among the most popular tasks for computational art. However, the presented methods often employ only a single style or require an exemplar image as a

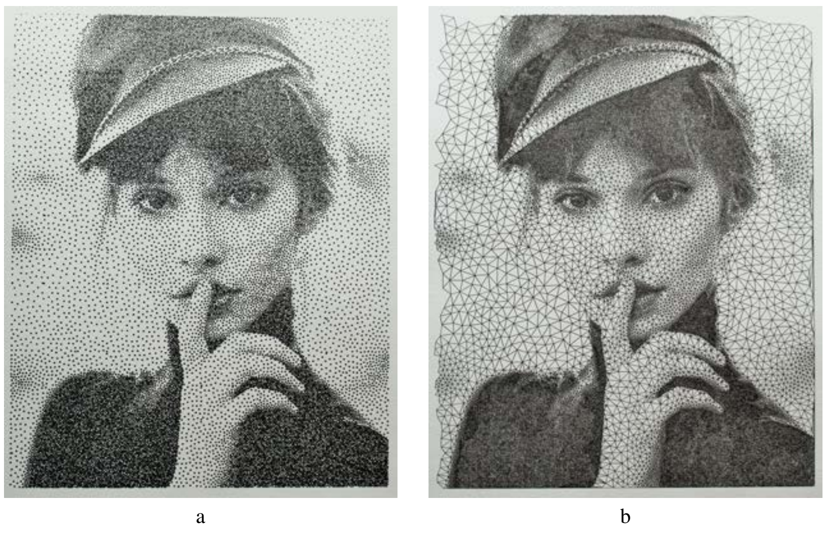


Figure 3.17: Two examples of positional encodings, a stippled image in (a) and an image created by triangulating the stippling points in (b). Both styles depict the image through the position of the primitives and not their shape.

style reference. For more flexible approaches, computational artists often use dedicated software such as Processing [12] to create their artworks. While such frameworks provide a multitude of tools for art generation, they usually require the artist to write code from scratch and do not support the user in the process of finding suitable parameters for the visual primitives, potentially leading to a trial and error process. The goal of our research was therefore to develop a system that provides the user with a comparable degree of flexibility for the generation of monochrome are, while automatically supporting the user in finding suitable parameter settings at the same time. Similar to other approaches presented in this thesis, we also aimed for a system that is suitable for novice users with little programming experience.

One of the key concepts behind our LinesLab system is the construction of visual primitives as a combination of a positional and shape encoding. Our system automatically samples over the parameter set to find a parameter constellation that produces an artwork representing the input image most faithfully according to a visual metric. In the following section, we discuss the concept behind the visual primitive construction through a combination of positional and shape primitives.

Many monochrome artworks experience a significant degree of overlap in terms of their visual encoding. Halftone styles, for example, are usually generated by placing differently-sized circles on a fixed grid. Often, to achieve a variation of the style, the grid density is changed, while the shape of the primitive stays fixed. On the other hand, one can compose a new style by using a different shape. To avoid unnecessary redefinition of the visual primitives, we seek inspiration from art. Most stroke-based drawing techniques consist of two abstractions, the position of a pen stroke and the shape of the stroke. In fact, art students often train their skills and the understanding of the styles by limiting themselves to a single abstraction. Following this concept, we divide visual primitive encoding into *positional* and *shape* encoding.

Positional encoding describes all styles that depict the image content using position and density of the visual primitives, while the essential shape of the primitives stays

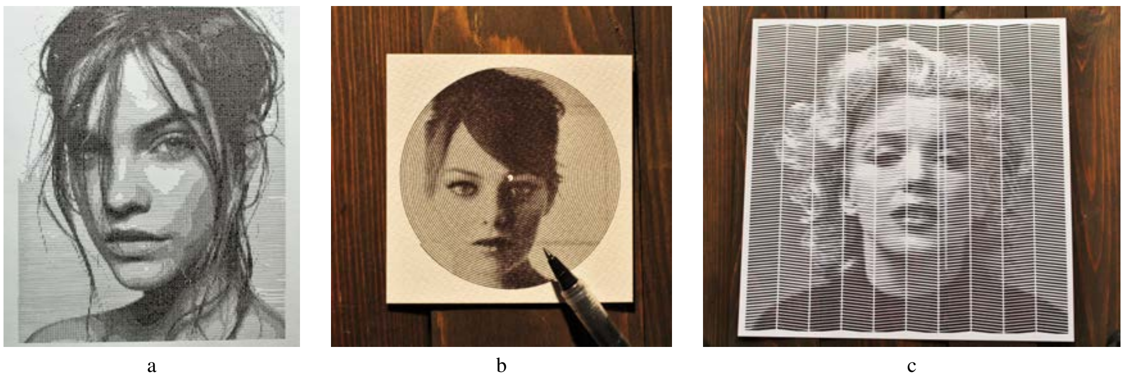


Figure 3.18: Three examples of styles where only the image intensity is encoded in the shape of the visual primitive, (a) shows a version of a halftone technique, (b) shows an amplitude modulated and (c) a width modulated cut-out illustration.

unchanged. The most prominent example of a style using a purely positional encoding is stippling. We show an example of a stipple illustration in Figure 3.17 (a). Notice that only the density of the dots conveys the tone of the image while the shape of the primitives stays constant. However, positional encodings can be represented using more complex forms. In Figure 3.17 (b) we create a new illustration by triangulating the stippled points. While the line length changes, the line thickness stays constant.

With shape encoding, we describe all techniques that map the image intensity to a geometric property of the primitive. Common choices for a mapping target are primitive size, texture, or orientation. The position of the primitive is fixed and not affected by the mapping. Typical examples of shape encodings are halftone images, such as shown in Figure 3.18 (a). However, the primitives do not need to be aligned to a regular grid. We show two examples of such illustrations in Figure 3.18 (b) and (c). Figure 3.18 (b) shows an amplitude-modualted illustration where the visual primitive, a cosine function, is aligned to an Archimedean spiral. Figure 3.18 (c) shows a width-modulated portrait created by cutting out strips of paper with varying width.

These two encoding types are not mutually exclusive and can be combined to generate new drawing styles. Most monochrome line art can be expressed as a combination of positional and shape encodings with varying degrees of contribution. In Figure 3.19, we provide some examples of design combinations with different degrees of freedom of the positional and shape encoding, respectively. Notice that the styles with fixed positional and shape encoding cannot reproduce a tonal image and are not included in the diagram. Using this notion, we can interpret the styles in Figure 3.17 as combinations with a high contribution through the positional encoding and no contribution trough the shape encoding, i.e. a fixed shape. In contrast, illustrations shown in Figure 3.18 are constructed through positional encoding only with a fixed positional encoding.

A crucial part of our contribution on monochrome art generation is an efficient system for the simulation of a large number of drawings with different parameter sets. As mentioned earlier, we employ Latin Hypercube sampling of the parameter space to find a parameter set that generates a drawing visually closest to the input image. To compute the image similarity, we rasterize the input image and the drawing simulation and compute the normalized mean square difference between them. This simple metric is chosen because it is less sensitive to small structural differences between the input

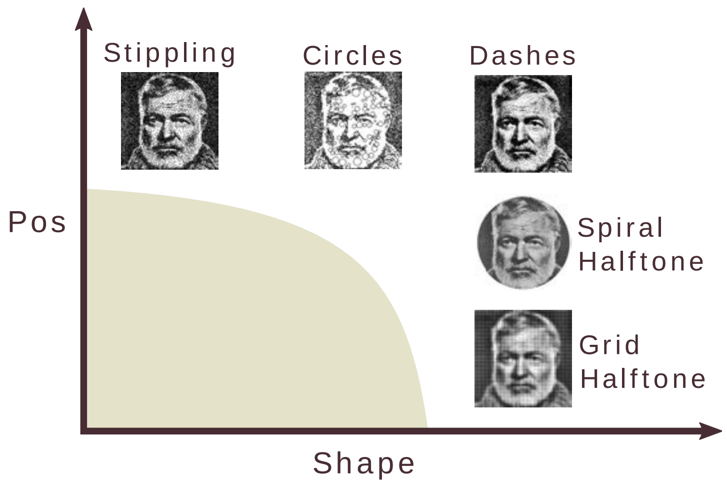


Figure 3.19: The design space of monochrome line art can be seen as a space defined by the contributions of positional and shape encoding. Note that restricting the shape and positional encoding results in a fixed illustration and is therefore not suitable to reproduce tonal images.

image and the simulated drawing. Moreover, our system does not create typical failure cases of the mean square error metric, such as image transformations or color shifts.

However, drawings can only be abstractions of the input image. Therefore, it is unfeasible to compare the two images directly. To account for this we apply a Gaussian filter to both the input image and the rasterized drawing simulation. Using the Gaussian filter, we furthermore can account for the distance from which the drawings will be viewed. By incorporating information about the average retinal resolution, we can optimize the similarity of the drawings dependent on the viewing distance by using an appropriate kernel size.

3.3.3 Parameter Tuning of Dynamic Lights

Our last contribution in this thesis is an automated system for scene illumination with dynamic light sources. The position of light has a crucial impact on the perception of three-dimensional scenes. However, finding appropriate light positions is a challenging task. Traditional approaches for lighting design employ an iterative process of trial and error, where the user continuously adjusts the light parameters and evaluates the rendered image. To address this problem, several methods for automatic light placement have been investigated over the last years [82, 131, 222, 225]. While automated light placement is a highly complex problem, this is even more true for dynamic lights. Instead of evaluating a light position, dynamic light setups require the evaluation of the whole light path, drastically increasing the complexity.

Nevertheless, users might significantly benefit from dynamic lights participating in interactive scenarios. Typically, to assess the shape of objects, users rotate them back an forth. If the object cannot be moved easily or to evaluate local features of the object, users tend to move the light source and evaluate how the light interacts with the object. Moreover, dynamic light sources are often used in modern animation approaches to convey temporal changes in the scene, provide atmospheric changes, or adapt the light conditions to dynamic scenes. The simultaneous control of light sources

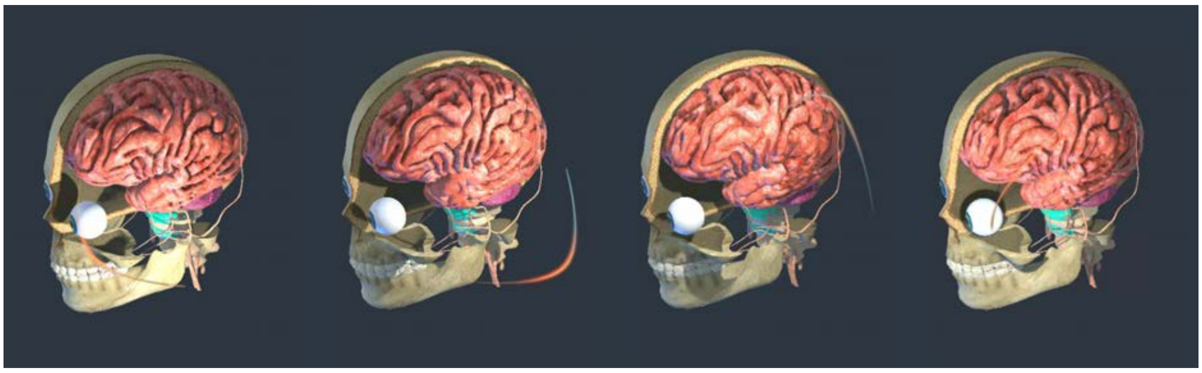


Figure 3.20: The dynamic nature of the Firefly allows to emphasize multiple aspects of the scene.

and interactions in the scene, however, can be cumbersome. To address this challenge, we propose an automatic approach for scene illumination with dynamic light sources that offers additional perceptual cues. In addition to the static light in the scene, we generate a Firefly – a virtual light drone that illuminates the scene while flying on an automatically generated path. Our system continuously optimizes the Firefly path based on a flexible energy function, designed for a number of visualization tasks. The dynamic nature of the Firefly allows it to illuminate the scene from various positions, for example to emphasize local variation of the object as shown in Figure 3.20.

While previous approaches for the placement of static light sources could rely on precompuation, our goal was to provide a fully automatic solution that dynamically adapts to changes in the scene or camera movements, while not imposing any constraints on the scene. As in our previous contributions, a key component of Firefly is the user-oriented computation, i.e. the Firefly acts in a view-dependent manner, adapting the light path to what the user actually sees. Our approach can therefore be seen as an online optimization process. This imposes an additional challenge on performance, as the Firefly path needs to be optimized in real time without affecting the user's interaction. To still achieve interactive performance, we employ a parallel rendering pipeline for the light path evaluation.

We obtain a high degree of flexibility for numerous visualization tasks and address both non-expert as well as experienced users by designing the Firefly system as a detached optimization process. The optimization process is controlled through a plugin-based energy function, where the energy function can either be selected from a catalog or be defined through a code snipped by an experienced user. Our system aims to find a Firefly path that minimizes the chosen energy function by performing an adapted version of Simulated Annealing with Latin Hyper Cube sampling

3

«No quote before Chapter 4 ... hhhmmmmmm?»
Sarah

Chapter 4

Demonstration Cases

The contributions of this thesis cover exploration, mapping and simplification, and optimization of parameter spaces. In this chapter, we illustrate the utility of our contributions in different application scenarios. Following the classification presented in Chapter 3, we will demonstrate results for parameter space explorations first, followed by simplified interaction spaces, and finish with results from automated parameter tuning.

4.1 Exploration of Perfusion Data

Medical practitioners can obtain valuable information from the shape of time-intensity curves from medical acquisitions with a temporal dimension. For example, the shape of time-intensity curves in contrast-enhanced MRI scans can inform the practitioner about the pathological nature of the tissue. Perfusion data, for instance, can help diagnose malfunctioning kidneys. The kidney cortex has a high absorption rate with a well-defined shape. Perfusion curves that differ substantially from the expected shape indicate organ malfunction, such as kidney fibrosis. With our method, we enable the easy inspection of the time-intensity curves as well as the identification of spatial regions with particular characteristics. In the following demonstration case, we present a 4D contrast-enhanced MRI scan of the abdominal region. Typically, to investigate the perfusion rate of the kidneys, the practitioner selects a point of interest in each kidney and compares the time-intensity curves of the two. However, this process is very sensitive to outliers and the practitioner needs to know a priori where to place the points of interest. Our approach helps to locate the region of interest and provides additional robustness compared to the standard procedure. In Figure 4.1, the user created a region of interest around both kidneys to gain an overview of the perfusion rates within each kidney. Next, the user extracted time-intensity curves to compare them in a separate window. In this particular case, the data set was obtained from a healthy volunteer and thus both kidneys show regular perfusion rates.

Often, the practitioner is interested in the shape of healthy kidney tissue. Diseases such as fibrosis typically do not impair the the overall perfusion rate of the kidney but the tissue with healthy perfusion rates would appear deformed or reduced. To gain information on the spatial location of tissue with a certain perfusion characteristics, the user can perform selections in the value domain. In Figure 4.2 (a), the user has selected data points with high perfusion rates and displays them with a second distinct

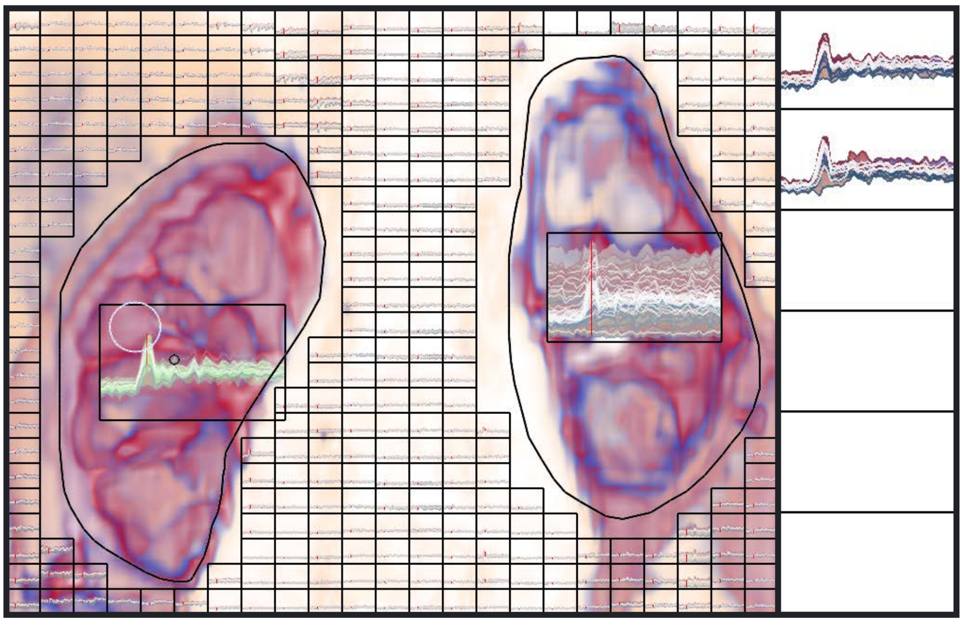


Figure 4.1: Our method allows users to investigate the perfusion rates of the whole kidneys. Furthermore, we provide mechanisms to extract and compare the perfusion rates in detail in a side panel.

transferfunction in Figure 4.2 (b). Again, since this data was obtained from a healthy volunteer, the kidney shape shows no signs of deformation.

4.2 Interactive Paper-Based Volume Visualization

In this section we present three examples of interactive visualizations with a Vol^{2}velle. The first paper-based visualization preserves interaction with multimodal volume rendering of a PET-CT scan of a human head with a tumor on the right side of the chin. Multimodal data depicts morphological as well as functional information and is commonly used in medical applications. Often, practitioners utilize linked views during doctor-patient communication to explain morphological structures in detail. Our approach could be used to create a physical visualization for patient education prior to an upcoming surgery, that the patient could inspect in detail after an meeting with the doctor. We fix the transferfunction of the PET volume and map the cyclic parameter of the Vol^{2}velle to the opacity of the CT scan. Changing the opacity of the CT scan allows the patient to understand the nature of the depicted pathology better. Furthermore, we use the slide chart extension to preserve the slice view. The position of the slices is annotated in the volume visualization. The slide chart can increase the understanding of the complex data. A photo of the assembled Vol^{2}velle is shown in Figure 4.3. The left side of the image shows the whole Vol^{2}velle assembly and the images on the right side show two linked view constellations and a close up of the transferfunction of the CT scan with the data histogram.

A very natural choice for the cyclic parameter of the Vol^{2}velle is the position of a light source. In the following Vol^{2}velle example, we use a microCT scan of a millipede species [153]. Similar data sets are often presented in museum exhibitions as part of a virtual installation. Typically, the visitors have several interaction methods with

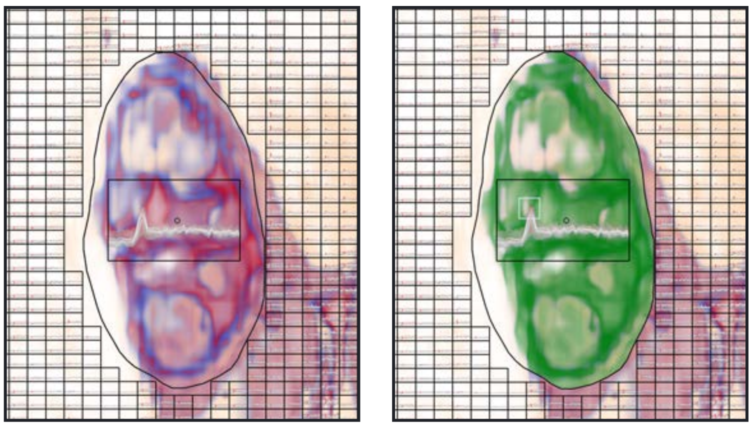


Figure 4.2: To investigate the shape of data with specific value characteristics the user can highlight data in the spatial domain through selections in the value domain.

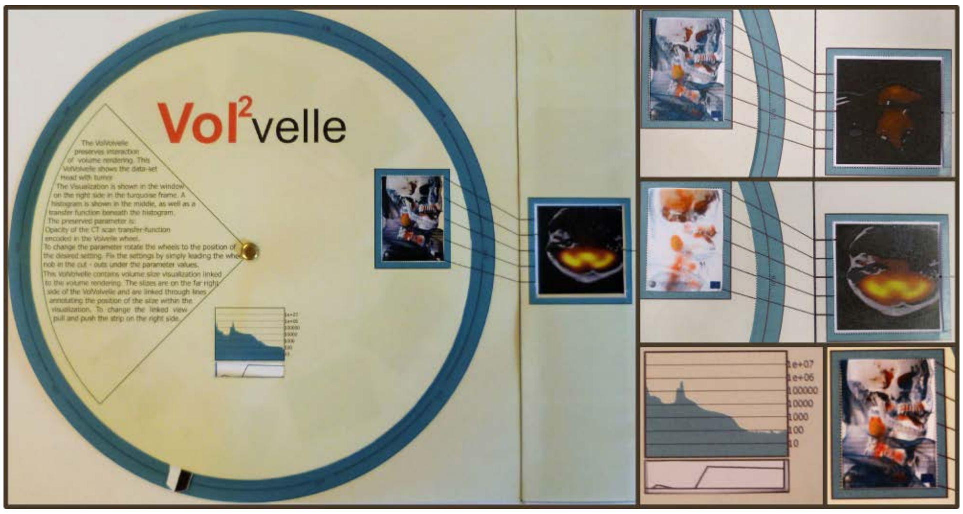
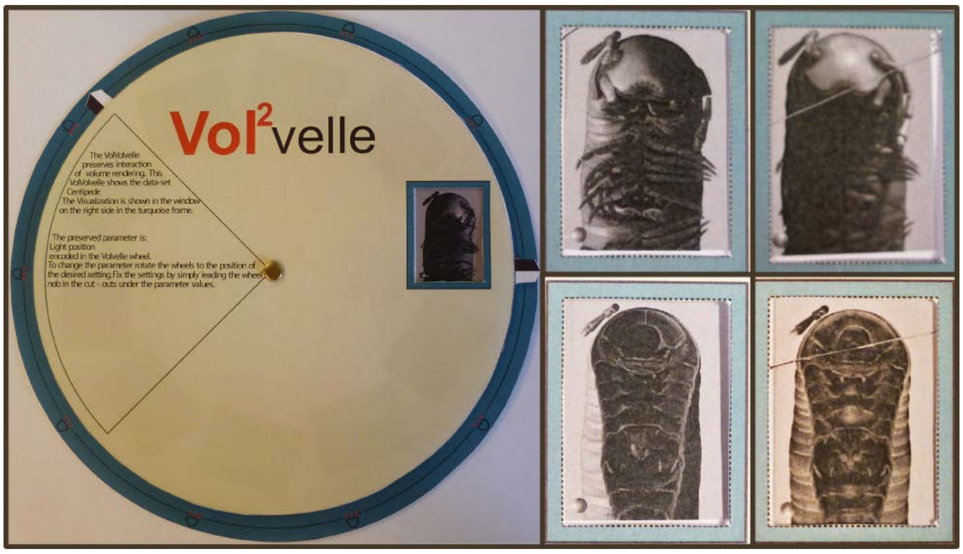


Figure 4.3: A Vol^{2}velle depicting multimodal data of a PET-CT scan with a linked slice view.

the visualization, such as modification of the light position, rotation of the specimen, or the placement of cutting planes. Such a setup provides an interesting application for the Vol^{2}velle. One can imagine that the visitors would have the option to create a small paper-based version of such interactive visualizations as a souvenir. In this particular example, we generate the Vol^{2}velle by preserving the interaction with the light position. Furthermore, we map the binary Vol^{2}velle parameter to the cutting plane position, essentially creating an on/off switch for the cutting plane. This defines a changing picture Vol^{2}velle with an iris mechanism. We show a photograph of this Vol^{2}velle in Figure 4.4. In order to enable the iris mechanism, some visualization images must be cut through. However, these cuts are barely noticeable under normal light conditions. Such an iris Vol^{2}velle is the most complex in terms of assembly and this particular example took 17 minutes to build. On the other hand, the iris mechanism makes this Vol^{2}velle particularly enjoyable as most users experience the mechanics as an element of surprise.

The goal of illustrative visualization is not to provide accurate representation of the data but rather to provide effective visual abstractions to make the visualized data



The VolVolvelle
preserves interaction
of volume rendering. This
VolVolvelle shows the data-set
Centipede
The Visualization is shown in the window
on the right side in the turquoise frame.

The preserved parameter is:
Light position
encoded in the Volvelle wheel.
To change the parameter rotate the wheels to the position of
the desired setting.Fix the settings by simply leading the wheel
nob in the cut - outs under the parameter values.

Figure 4.4: An iris Vol^{2}velle preserving the interaction with the light position and an on/off switch for a cutting plane.

clearly understandable. For this example, we create an expressive Vol^{2}velle with transparent wheels, depicting an illustrative visualization of a segmented CT scan of a human head using style transferfunctions [39]. Each of the segmented parts is assigned a fixed style transferfunction with opacity as the interaction parameter. Based on the distance to the viewer, we group the mandible and the eyes as well as the brain and the blood vessels and link the Vol^{2}velle parameters to their respective opacity settings. Arranging the wheels in the same order as the depicted volume structures allows us to approximate the composite volume rendering. We show a comparison of the rendered volume with a simulated layered image and the physical composition in the Vol^{2}velle in Figure 4.5. As can be seen, the blending with transparent media can only approximate the illustrative visualization. Still it provides an expressive visualization conveying the relationship between the individual structures. In Figure 4.6, we show the finished Vol^{2}velle with four different parameter settings on the right side. The user is able to adjust the transparency of each layer individually. In total, this Vol^{2}velle allows for 125 different parameter settings.

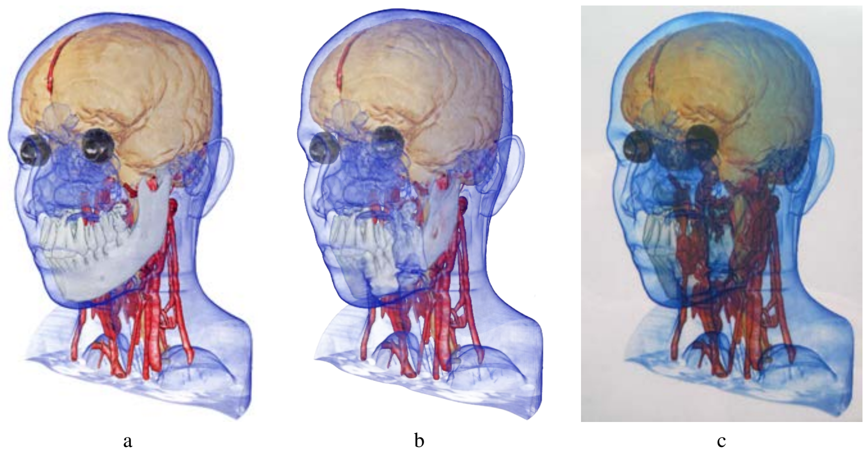


Figure 4.5: Comparison between (a) illustrative volume rendering, (b) simulated overlay image, and (c) physical overlay in a Vol^{2}velle assembly with transparent films. Because most printers are not capable of printing white color, the white jaw is more transparent in the physical blending.

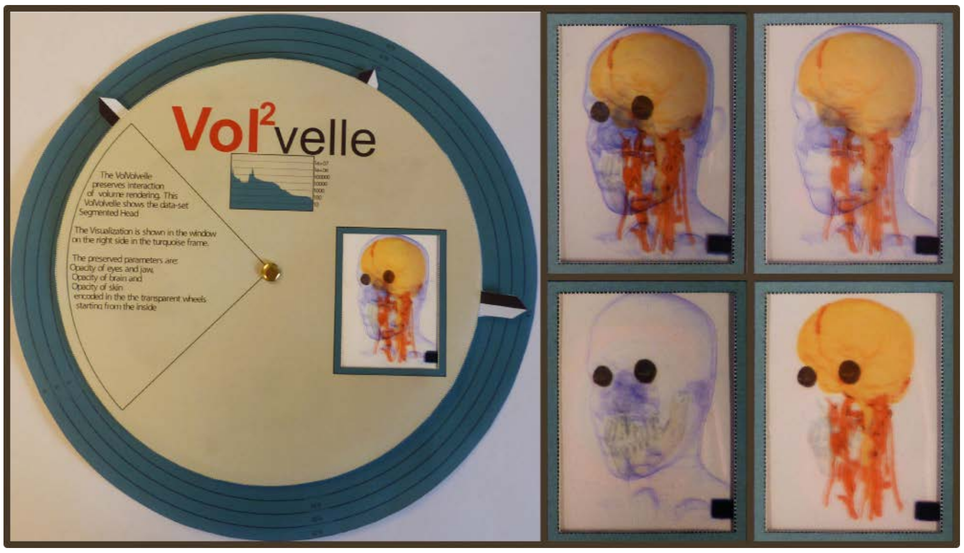

Vol²velle
The VolVolvelle preserves interaction of volume rendering. This VolVolvelle shows the data-set Segmented Head
The Visualization is shown in the window on the right side in the turquoise frame.
The preserved parameters are: Opacity of eyes and jaw, Opacity of brain and Opacity of skin encoded in the the transparent wheels starting from the inside

Figure 4.6: A Vol^{2}velle with transparent wheels can be used to approximate illustrative volume rendering.

4.3 Illustrative Cutaways with Surrogate Objects

In this section, we demonstrate the effectiveness of interaction through surrogate widgets on two expressive visualization examples. Medical illustrations often use cutaways to reveal hidden inner structures. In this example, we present an expressive cutaway visualization of the cochlea which was created by a study participant within just over 7 minutes after a short training period. The human cochlea is the auditory portion of the inner ear that is divided into three spiral-shaped parts filled with lymph fluids. Medical illustrations typically depict the cochlea using a number of non-planar cutaways as shown in Figure 4.7 (a). During a user study, we asked the participants to recreate this illustration in the volume rendering. One result of such an expressive cutaway visualization by a non-expert is shown in Figure 4.7 (b). To create this visualization, the study participant used a branched widget with four spheres in total. The cochlea was opened with the wedge tool, and a volume that is unaffected by the tool inside the widget was inflated. To achieve comparable results, other study participants needed between 3:37 and 13:42 minutes in total. Although the cochlea geometry is rather complex, seven out of nine participants reported that they perceived the creation of this visualization as an easy task.

Technical illustrations often use section views that exploit symmetries in order to reveal inner structures. Typically, in order to achieve this in volume visualization, existing approaches require additional information, such as segmentation. Our smart surrogate widget approach allows us to quickly achieve similar results without any prior prepossessing. In Figure 4.8, we illustrate our technique in a CT scan of a high voltage outlet. Standard volume rendering with opacity modulation does not depict the relationships of the outlet casing and the inner structures clearly, as can be seen in Figure 4.8 (a). Using our technique, we can automatically place a surrogate widget consisting of two spheres (Figure 4.8 (b)). The outlet can be opened with a combination of an iris and a wedge tool, as depicted in Figure 4.8 (c). Inflating an inner volume allows us to display the inner parts with a second transferfunction, creating an expressive visualization, as shown in Figure 4.8 (d). Using our smart surrogate widget, we are able to depict the shape of the inner structures while conveying the overall shape of the object.

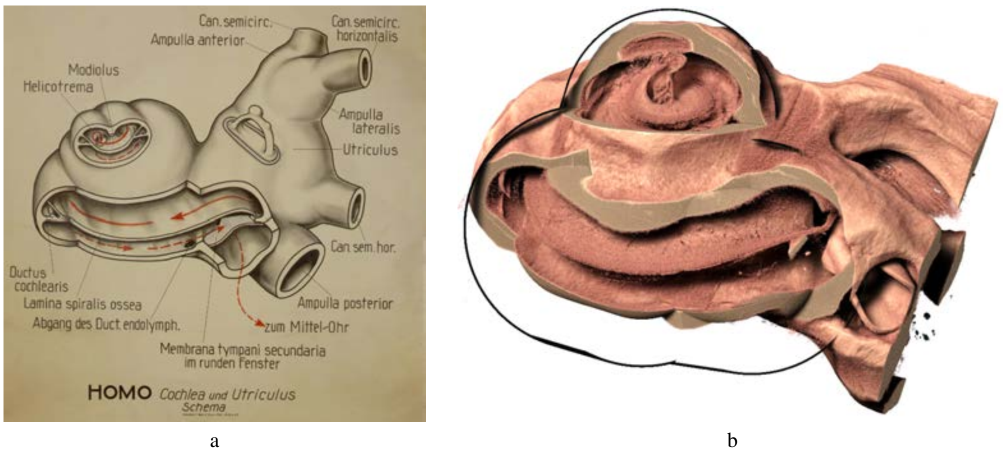


Figure 4.7: (a) Medical illustration of the cochlea from Humboldt-University Berlin. (b) An illustrative visualization created by a non-expert user during a user study.

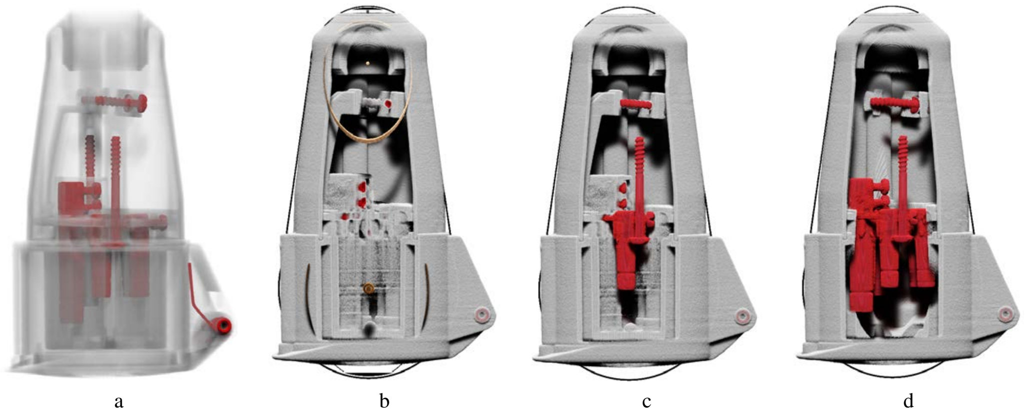


Figure 4.8: (a) Transparency adjustment of the occluding medium does not sufficiently reveal the relationship between the outlet parts. With our method we can place a smart surrogate widget directly on the power outlet (b), and open the volume up with a combination of an edge and an iris tool (c). Furthermore the volume inside the widget can be rendered in a second transferfunction (d), revealing the structural relationship between the individual parts clearly.

4.4 Automated Generation of Monochrome Art

Automated parameter tuning methods require a dense sampling of the parameter space. This imposes a challenge on the algorithm design to be as efficient as possible. To reduce the computational overhead of the computation, we parallelize the iterative generation of the primitives in the LinesLab system. To allow parallelization of the primitive generation, we subdivide the image into overlapping tiles (Figure 4.9), which are then processed in parallel. The primitive computation is then performed as follows: First, the image is subdivided into tiles that have at least a pairwise distance of twice the size of the largest possible primitive. This ensures independence between the tiles when they are processed. Next, the tiles are shifted as shown in the subsequent images in Figure 4.9 and the computation of the visual primitives is repeated in each step.

This subdivision strategy increases the computation speed not only by allowing us to process the tiles individually but also by reducing the image size that needs to be

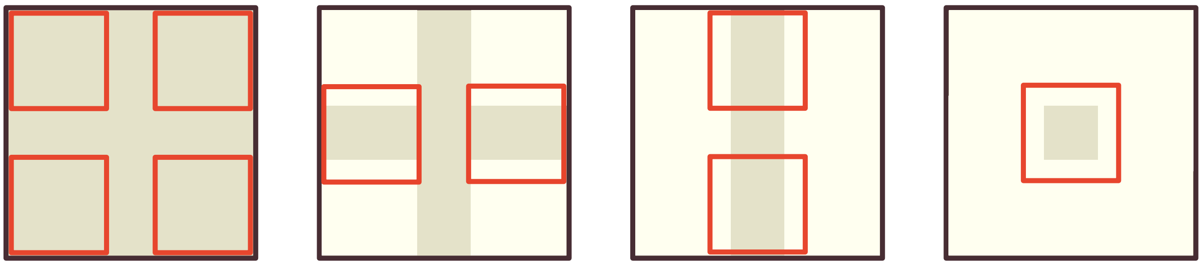


Figure 4.9: To increase the computation speed for the drawing simulation, we subdivide the image into a set of non-overlapping tiles. Computing the primitives in four consecutive steps with a tile arrangement as shown above, allows to compute the visual primitives in parallel for each tile.

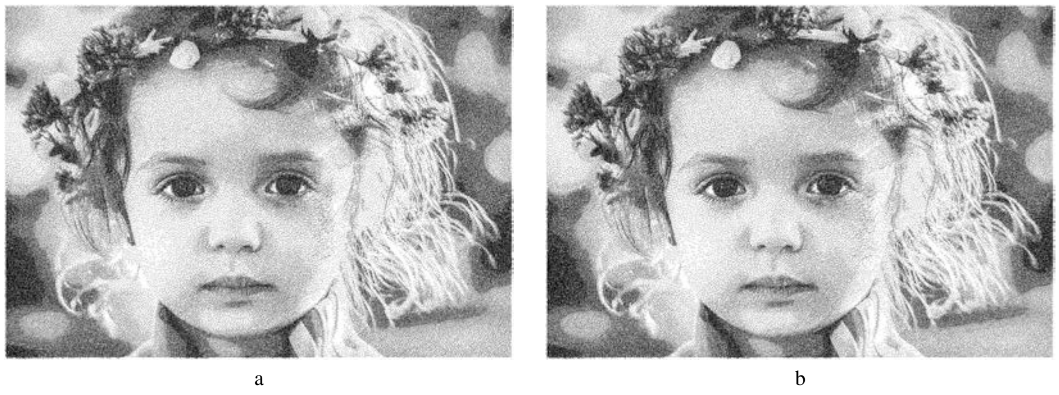


Figure 4.10: Comparison between serial and parallel method. To compute illustration (a) our system required 741 seconds, whereas the illustration in (b) was computed with a parallelizable tiling strategy, requiring only 1.41 seconds.

evaluated for each primitive. This rather simple strategy leads to a performance increase by a factor of approximately 380 compared with the naive approach. However, because the visual primitive positions are computed by scanning the image row by row, the primitives tend to accumulate at the right and lower border of each tile. Potentially, this can produce undesired artifacts. To minimize these artifacts, the primitive computation is performed in small batches, i.e. we subdivide the computation for all tiles into n steps and construct the primitives up to $\frac{k}{n}$ image intensity in the k-th step. This results in more evenly distributed primitives without any noticeable artifacts. In Figure 4.10, we compare two images computed using a completely serial method (a) and using the parallel tiling method (b). The serial illustration in Figure 4.10 (a) took 741 seconds to compute while the illustration in Figure 4.10 (b) only required 1.41 seconds. The resulting images appear identical to the naked eye.

Using the same style for the visual encoding as above, we demonstrate the scalability of our system in terms of medium size. We show three physical drawings generated with our methods in Figure 4.11. The leftmost drawing has a size of 3.5×5 cm and is optimized for a viewing distance of 20 cm, the middle image is 21×29.7 cm big and is optimized for a viewing distance of 120 cm, and the rightmost image measures 29.7×42 cm and is optimized for a viewing distance of 170 cm. Clearly, the smallest canvas does not allow the same level of detail as the larger illustrations. However, our system is able to create illustrations faithfully depicting the input image within the size

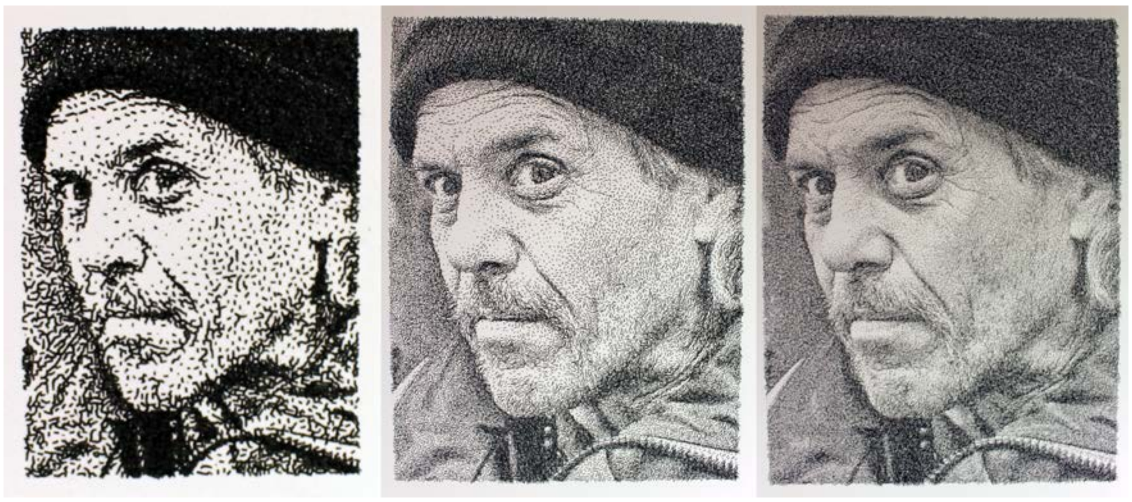


Figure 4.11: A demonstration of the scalability of our system in terms of medium size. Our system is able to create illustrations on canvas of size 3.5×5 cm (left image), 21×29.7 cm (middle image) and 29.7×42 cm (right image).

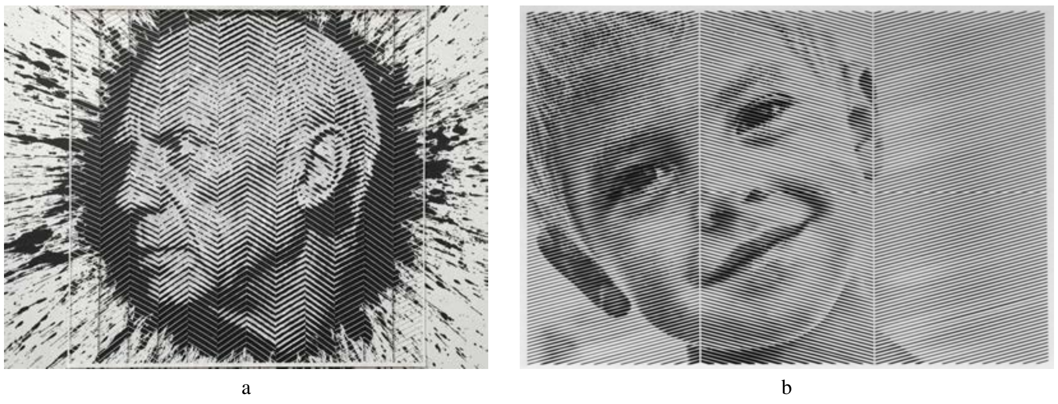


Figure 4.12: (a) Cutout portrait by Yoo Hyun. (b) Cutout generated with our system.

restrictions of the given medium.

The last example in this section demonstrates how our approach can be used for other monochrome art, such as paper cut outs. The Korean artist Yoo Hyun [16] creates detailed portraits by carefully slicing away thin slivers of paper. Hyun depicts the tonal variations in the portraits by adjusting the thickness of the cutouts as shown in Figure 4.12 (a). We are able to recreate this style with our LinesLab system by choosing a similar encoding. Essentially, the positional encoding creates a herringbone pattern for the cutouts and the shape encoding is represented by the local thickness along the cutout. Our LinesLab system automatically optimizes the parameters to create a pattern that resembles the input image closely while ensuring paper stability. One cutout illustration created with our system is shown in Figure 4.12 (b). In contrast to Yoo Hyun, our system is required to leave space between the segments to ensure stability during the cutting process. Hyun spends tens of hours creating such portraits, while with our system the production of the final cutout took just over 45 minutes, including computation and fabrication time.

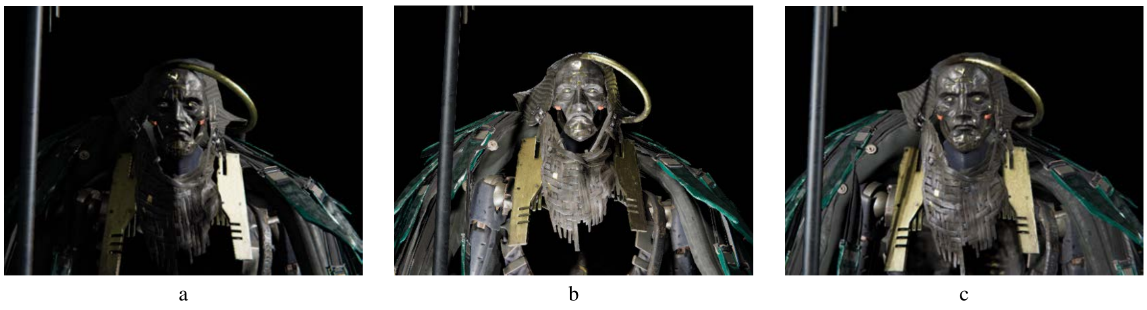


Figure 4.13: Comparison of different moods encoded through the energy function: (a) shows a dramatic scene, (b) a threatening scene, and (e) a calm scene.

4.5 Automated Scene Illumination with Dynamic Light Sources

We have designed our Firefly system with the intent to be easily usable for a multitude of visualization tasks. To achieve a high degree of flexibility, we optimize the Firefly parameters using a plugin-based energy function. In this section, we demonstrate our system on two distinct application scenarios for light design.

Lighting plays a significant role in cinematic applications. Many cinematic animations employ dynamic lights in addition to static setups. Depending on the intent of the designer, light can be used to convey emotions, emphasize dramatic events, or guide the viewer's attention to details in the scene. When illuminated objects move within the scene, the light positions need to be adjusted as well. This imposes a challenge on the animator to carefully construct a light path that captures the scene changes without losing the desired artistic effect. We can address this challenge with our system through automatic tuning of the Firefly path. In this example, we use a character from the animated short film Adam [2]. Because we define the Firefly path relative to the object of interest, it moves automatically with the character and adapts to the changing conditions on the fly. We define the illumination setup with three different energy functions describing different moods for the scene. The first energy definition creates a dramatic atmosphere, the second energy function establishes a threatening appearance of the character, and the third produces a calm atmosphere. We show representative snapshots of the animation in Figure 4.13. Figure 4.13 (a) shows a dramatic scene, (b) a threatening scene, and (c) shows a rather calm atmosphere.

When exposed to realistic scenes, the user typically expects certain light positions. If the user's expectations on light positions are not met in such setups, they might misinterpret shapes and the scene's structure. In such cases, the lighting setup must be designed with special care in order to avoid confusion. Therefore, when designing an animated light for realistic scenes, for example a still life, we need to closely follow photographic lighting setup rules. In this example, we define a dynamic lighting setup for a still life. We formulate the energy function guided by the approach of Wambecke et al. [222]. The method by Wambecke et al. creates a natural lighting setup for one light, by restricting the light position to the upper hemisphere, as expected by users, and choosing a light position that emphasizes the surface variation in the scene. In Figure 4.14 (a), we show the results of this illumination setup with a still live.

The nature of static lights does not allow us to emphasize the surface variations of every object simultaneously, instead the approach selects a light position that maxi-

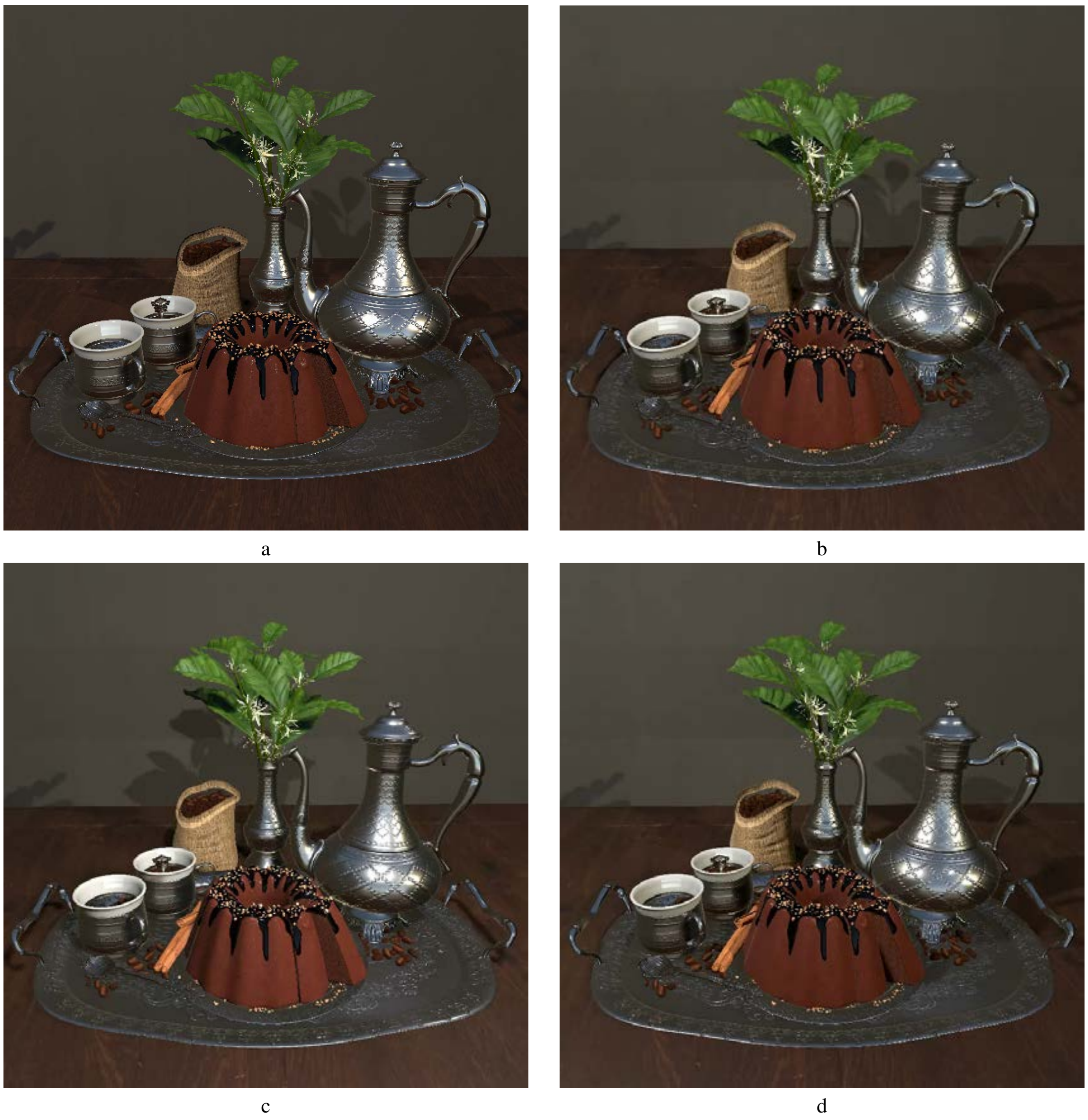


Figure 4.14: Still life rendered with the method of Wambecke et al. (a). The same model illuminated with the Firefly method (b,c,d). The lighting in (a) and (b) is almost identical, but (c) and (d) reveal the surface variation of the remaining model parts.

mizes the surface variation for the overall scene. Potentially, this can lead to misinterpretation of individual elements that cannot be effectively illuminated with a static light. Guided by the approach of Wambecke et al., we construct an energy function which is able to emphasize the surface variation of the whole scene over the course of the light trajectory. Using such an energy function results in a scene illumination as shown in Figure 4.14 (b,c,d). Comparing the images in Figure 4.14 (a) and (b), one can see that our method produces very similar results to the static setup. However, the dynamic nature of the Firefly emphasizes further details in the scene. In Figure 4.14 (c), the Firefly creates shadows that enhance the shape of the cake and in Figure 4.14 (d) the shape of the coffee bag is much more prominent.

4

«A conclusion is the place where you got tired of thinking.»
Martin H. Fischer

Chapter 5

Conclusion and Future Work

The goal of this thesis was to explore solutions for various challenges that arise in parameter spaces. During the work on this thesis, we have steadily progressed towards automated methods. We presented solutions for parameter exploration, parameter space mapping, and automated parameter tuning. We have demonstrated the applicability of our solutions for several application scenarios. We proposed a novel visualization primitive for the exploration of time-dependent volumetric data. The proposed primitive creates a link between the spatial and the temporal domain of the data, and we introduced several interaction techniques for both domains. To address non-expert users, we presented two approaches to simplify the interaction space for volume visualizations. The first approach allowed us to create hardcopies of interactive volume visualizations, while preserving a certain degree of interactivity. We presented several interactive assemblies generated with our system, including assemblies with advanced functionality such as linked views or simulation of volume rendering with transparent media. The second approach for simplified interaction spaces provided a set of intuitive interactions through surrogate widgets within the visualization domain. The proposed surrogate widgets adjust to what the user actually sees and act as data proxy, allowing for goal-oriented modifications without occluding the visual space. An evaluation of our approach through a user study suggest that out method is well suited for non-experts and requires minimal training time.

Lastly, we presented two methods for automated parameter tuning for complex parameter spaces. We have introduced a novel method for the fabrication of monochrome line art that allowed for the generation of a multitude of different styles with reduced effort. Our proposed method automatically adjusts a large set of parameters for the visual primitive encoding to produce art, visually most similar to an input image, while maintaining flexibility in terms of canvas size. The second automated method solves the challenge of automatic scene illumination with dynamic light sources. In addition to the existing static light sources in a scene, we generate a moving light drone. This light drone illuminates the scene while flying on a closed path. The flight path automatically adapts to changes in the scene and camera movement by the user to optimally illuminate the changed scene based on a flexible energy function. We illustrated our technique on several visualization tasks including cinematic scenarios for the communication of scene emotion.

The research presented in this thesis opens up several directions for future research. As data is becoming increasingly more diverse and heterogeneous and algorithms are getting more and more complex, parameter spaces are likely to become even more challenging to investigate in the future. This increasing complexity will surpass the human cognition to fully understand the data and the underlying processes. Inevitably, the human will become the weakest link in many processes. It is therefore even more important to utilize automated methods and to employ the user only when necessary. With the rapid advances in machine learning and artificial intelligence, the scientific community will be challenged to integrate the human into such systems. Currently research directions, such as interpretable machine learning, aim to include the user in the information discovery. However, there is a lack of effective interfaces and interaction techniques, that are able to scale well with rapid methodological advances. To effectively integrate the user in the process, future solutions will need to exploit all three of the presented strategies for parameter spaces.

We see several potential research directions as direct follow-ups from the presented solutions. The presented automated methods, such as automated scene illumination with dynamic light sources, depend on objective functions for the quality measure of the parameter settings. The definition of such energy functions still requires a certain level of expertise. In the future, we plan to address this with an intuitive visual editor for the composition of energy functions that provides the user with a preview of the potential outcomes on the fly.

We have presented an approach for preservation of interaction in paper-based visualization. A promising research direction is to explore interaction preservation for further physical media, such as 3D printed objects or mechanisms created with laser cutters. An interesting question that arises is whether there exists a model for responsive visualization that could automatically propose an appropriate visual representation and interaction space for different information media.

We believe the need for smart interaction will become more pressing in the future. Therefore, the exploration of further intelligent interaction techniques that automatically support the user, provide an interesting research direction. For example, methods that measure the activity profiles of features in time-dependent volumetric data, could automatically navigate the user to active regions or provide additional views of the events.

«Dude, suckin' at something is the first step to being sorta good at something.»
Jake the Dog

Part II

Scientific Results

Paper A

Graxels: Information Rich Primitives for the Visualization of Time-Dependent Spatial Data

Sergej Stoppel[1], Erlend Hodneland[2], Helwig Hauser[1] and Stefan Bruckner[1]

[1] University of Bergen, Norway
[2] Christian Michelsen Research, MedViz

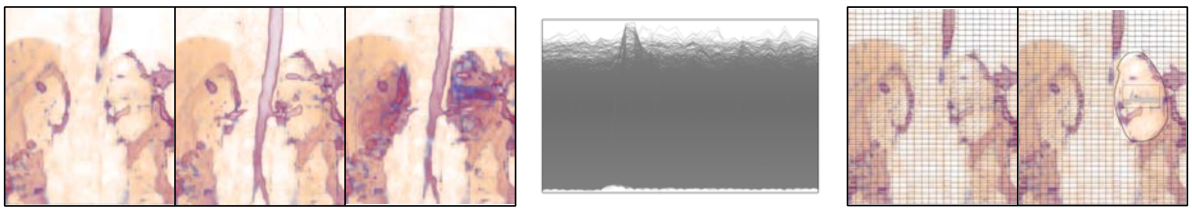


Figure A.1: Conventional volume rendering shows time-dependent data as animation, which can convey qualitative temporal development well but does not provide an overview of the whole time span at once or the quantitative development of the data (left). Showing all time-intensity curves for a volume results in a cluttered appearance (middle). Our method (right) provides information about temporal developments in the form of small multiples in their spatial context. We allow multiple interactions in the spatial and value domain for further data exploration.

Abstract

Time-dependent volumetric data has important applications in areas as diverse as medicine, climatology, and engineering. However, the simultaneous quantitative assessment of spatial and temporal features is very challenging. Common visualization techniques show either the whole volume in one time step (for example using direct volume rendering) or let the user select a region of interest (ROI) for which a collection of time-intensity curves is shown. In this paper, we propose a novel approach that

This article was published in Proceedings of *the 6th Eurographics Workshop on Visual Computing for Biology and Medicine* (2016). pages 183–192

dynamically embeds quantitative detail views in a spatial layout. Inspired by the concept of small multiples, we introduce a new primitive *graxel* (graph pixel). Graxels are view dependent primitives of time-intensity graphs, generated on-the-fly by aggregating per-ray information over time and image regions. Our method enables the detailed feature-aligned visual analysis of time-dependent volume data and allows interactive refinement and filtering. Temporal behaviors like frequency relations, aperiodic or periodic oscillations and their spatial context are easily perceived with our method. We demonstrate the power of our approach using examples from medicine and the natural sciences.

A.1 Introduction

One of the most common and straight-forward strategies for visualizing time-dependent volume data is animation. While simple and conceptually easy to understand, this approach is not well-suited for the detailed analysis of spatio-temporal patterns such as spatially-localized trends in the value domain. For example, in dynamic contrast enhanced MRI (DCE-MRI), the shape of a time-intensity curve in a particular region in space can provide important information about the malignancy of a tumor [119]. For this reason, visualization methods typically combine an animated single time-step visualization with additional views that depict data values over time in the form of curve plots. While these additional views facilitate the detailed analysis of time-dependent data, they do not provide a spatially-localized overview of temporal patterns.

For this reason, we propose a novel approach which provides a comprehensive overview of temporal relations directly embedded in a spatial context. Inspired by the concept of small multiples, we embed view-dependent interactive time-intensity graphs in a 3D visualization. Regions of interest can be easily generated by merging and splitting of these embedded graphs, and selections in time and space are possible. In contrast to previous approaches, our technique allows for the identification of frequency patterns, frequency relations and quantitative relations within time-dependent volumetric data. Furthermore, our method supports streaming data and is therefore well-suited for real-time applications such as the in-situ visualization of 4D ultrasound data.

The main contribution of this paper is a new information rich visualization primitive for the visual analysis of time-varying volume data. We demonstrate its applicability to the identification of temporally and spatially salient features. Furthermore, we show how our approach can be used for fast selection in the spatial and/or temporal domain. The strength of our approach is that it provides a spatially localized overview of time-intensity patterns in a single view. As such, our technique provides a useful addition to multi-view visual analysis systems for time-dependent data by establishing a link between the spatial and value domains which are usually treated separately.

A.2 Related Work

The visualization of temporal behavior has been extensively studied and the book by Aigner et al. [21] presents a comprehensive overview of the topic. However, the majority of the existing techniques focus on data without a spatial reference or the visualization is decoupled from the spatial domain. We divide our related work section into three main categories: visualization in multiple views, aggregation of time for a visualization in a single view, and integrated depiction.

Multiple Views:
Multiple views are a common approach to visualize time-dependent data. By presenting different dimensions or abstractions in multiple windows and linking them via selections, a fast understanding of relationships can be provided. Roberts [176] provided an extensive state of the art report on coordinated multiple views in exploratory visualization. Akiba et al. [23] introduced an approach for the simultaneous classification of the entire time series and explored options for transfer function specification based on a time histogram. Chang et al. [42] developed WireVis, a set of coordinated visualizations based on identifying specific keywords within financial transactions. Fang et al. [64] presented three different similarity measures for time-intensity curves and encoded them in three different visualizations. Akiba et al. [24] introduced a three component interface, which abstracts the complexity of exploration.Wang et al. [223] discussed an importance-driven approach to time-varying volume data visualization for enhancing the most essential aspects of time-varying data.Woodring et al. [235] presented an approach which transforms data points into multiscale time series using the wavelet transformation. Angelelli et al. [26] shown a technique for the exploration and semi-automatic segmentation of 4D ultrasound data by defining regions of interest and exploring their temporal development.

One drawback of such methods is that attributes like time-intensity curves are shown without a spatial context or with only a limited indication of spatial relationships. When displaying the data in separate windows one has to keep track of the links between the windows. With a growing number of windows this task becomes increasingly difficult. While our method provides multiple views as well, we mainly focus on displaying temporal information directly in its spatial context, thus improving the linking of the value domain and the spatial domain.

Data Aggregation to a Single View:
Data aggregation techniques aim to provide an overview of the data in a single intuitive view. Aggregation can be applied to the temporal as well as to spatial domains and can vary from stacking of the data, for instance in the form of a space-time cube, to finding elaborate transfer functions in the spatio-temporal domain.

The concept of the space-time cube, introduced by Hägerstrand [81], provides a seamless integration of the spatial and the temporal domain. In this approach the three-dimensional space inside the cube is used to represent spatio-temporal data. Kraak [126] used space-time cubes to place multiple layers along the time axis, each of them encoding the data at a specific time point, to visualize the development of the data over time. Jankun-Kelly et al. [101] discuss how the dynamic behavior of time-varying data may be captured by a single or a small set of transfer functions.

Woodring et al. [233] presented an algorithm which aggregates volumes over time producing a single view volume to capture the essence of multiple time steps in a sequence. Woodring et al. [237] also introduced an alternative approach to treat 4D data as one hypervolume instead of a collection of multiple time steps. Furthermore, Woodring et. al [234] discussed a method for comparative visualization by using logical operations on the data from multiple time steps. Lee et al. [133] presented an algorithm which estimates the appearance of new trends and extracts important trend relationships. Another approach by Lee et al. [134] computes the similarity between a voxel's time series and a feature. The similarity measure is then used for the visualization. Woodring et al. [236] provided a method for semi-automatically generating transfer functions for temporal data. Lee et al. [132] introduced CycleStack, an ultrasound video visualization technique, which reduced the cognitive overhead by blending video and signal together in a stack-like layout. Wang et al. [224] presented an application-driven approach to compressing and rendering large-scale time-varying scientific simulation data. Hsu et al. [94] discussed a set of visualization techniques for presenting the evolution of 3D flow.

Most of these techniques focus on 2D data. While our approach is suitable for 2D data as well, we focus mainly on time-varying 3D data. Furthermore, we do not aggregate the data into a single volume representation but rather focus on a comprehensive encoding, which captures the temporal behavior but is still easy to interpret.

Integrated Depiction:
Showing information in its spatial context has a critical impact on the understanding of local properties. The variation of integrated depictions is immense. It varies from abstract values, such as street names on a map, to complex representations such as graphs or intensity curves for locally selected features.

Moere [151] demonstrated how the principle of self-organization and behavior simulation can be used to represent dynamic data evolution by extending the concept of information flocking. Mlejnek et al. [150] presented ProfileFlags. In this approach a flag is placed on top of a T2 map of the anatomic scan of the patella. The tissue profile directly under the flagpole is then displayed on the flag. Lu et al. [139] discussed an interactive storyboard approach by composing sample volume renderings and descriptive geometric primitives that are generated through data analysis processes. Shearer et al. [188] introduced pixelplexing, a technique for the effective display of time-varying data on small screens. In a sense, their approach is inverse to ours, as they aim to increase the spatial resolution by animating static visualizations. Tikhonova et al. [212] presented an exploratory technique that enables a coherent classification of time-varying volume data. Forlines et al. [69] described an interactive system for the visualization of multivariate spatio-temporal data. Elmqvist et al. [62] presented ZAME, a nested visualization for the exploration of large graphs. For a comprehensive overview on composite visualization we refer to Javed and Elmqvist [105].

Our approach differs from the discussed ones in the sense that we use a novel information rich primitive in a screen filling way and not only on selected locations. Hence, we provide an overview of spatial and temporal aspects of the data that can guide subsequent selection or filtering operations.

A.3 Graxels

Tufte [215] emphasized the strength of small multiples for data comparison as "shrunken, high-density graphics" which are "drawn almost entirely with data-ink". As Tufte writes: "Our eyes can make a remarkable number of distinctions within a small area. ... 100 points in one square centimeter". Small multiples provide a highly efficient, dense, comparative visualization. However, small multiples are mostly used for abstract data without a spatial relation. We propose information rich primitives, which we call graxels (graph pixels) as a dynamic extension of traditional small multiples for the visualization of time-dependent spatial data. We embed time-intensity curves directly in the spatially-oriented visualization of the data set, thus providing a direct relation between spatial and temporal patterns. We further allow several interactions with the graxels, such as grouping and change of size. The size of the graxels can vary in different image regions and is therefore *non-uniform*. Furthermore, we introduce interaction and selection methods in both the spatial and the temporal domain of the data set.

There are several ways to aggregate data for the graxels. A straight forward strategy is to aggregate the data in object space. Here two conceptually different approaches can be considered: aggregation over a fixed volume subdivision or clustering of data points according to a similarity measure. These two approaches are comparable to the Eulerian and the Lagrangian views of fluid flow in the sense that the Eulerian approach focuses on fixed locations in space through which the flow passes, while the Lagrangian perspective follows particles as they move. Following the Eulerian perspective, we could regularly subdivide the data and perform aggregation into supervoxels. The time-intensity curves could then be displayed in each supervoxel. This approach has the disadvantage of being too rigid and may lead to arbitrary cuts in the volume.

Following the Lagrangian approach, the data points could be clustered according to some similarity measure. A graxel could be created for each cluster. This approach, while seemingly appealing, carries some disadvantages. For one, the results are highly dependent on whether an appropriate metric can be found, which in many cases is non-trivial. Moreover, the resulting clusters may have highly irregular shapes, which makes them badly suited as a canvas for graxels. Furthermore, while clustering methods for streaming data are advancing rapidly, the clustering process still imposes a high computational load on the process.

Our idea is inspired by the Eulerian perspective, but we avoid the rigid subdivision by performing the data aggregation in image space. This allows for an intuitive change of the aggregation through zooming, translation and rotation of the volume. The data aggregation is conceptually similar to conventional volume ray casting. Since our *graxels* cover an area larger than just a pixel, they can employ a more effective visual encoding than simple color mapping. We provide an aggregation technique which conceptually offers the same interaction as interactive volume rendering. Instead of casting rays for each pixel, we perform *tile casting* for *graxels*. This approach avoids the original rigidness of the Eulerian approach and allows for a flexible and dynamic adjustment of the aggregation through well-known interactions.

A.3.1 Overview

The goal of our approach is to capture the temporal behavior of the data without loosing too much detail information or missing important temporal or spatial features. Most of the data sets discussed in this paper are 4D scalar data sets with time as the fourth dimension and we are interested in the temporal development of these scalar values. This development is captured by time-intensity curves (TICs) at each voxel position.

The common way to visualize 3D scenes is to apply a viewing transformation to the data which is then projected into image space. A similar approach can be applied to

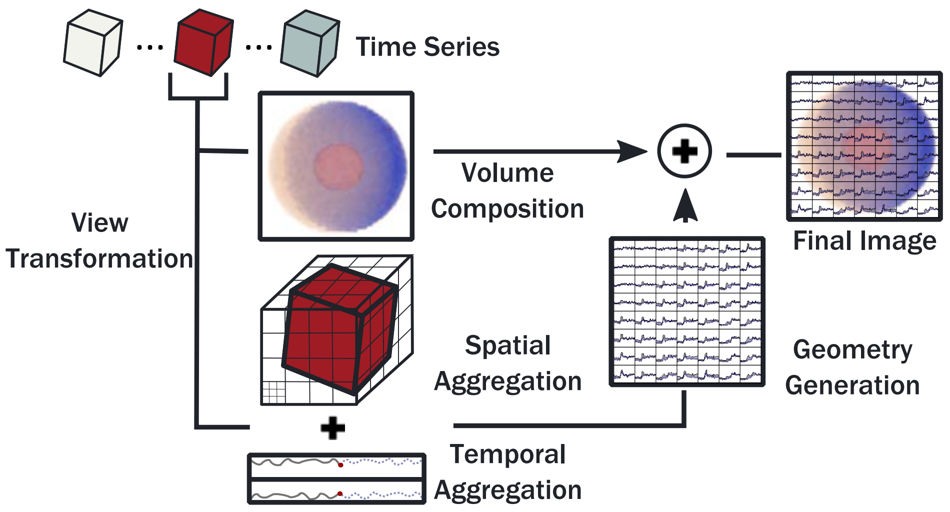


Figure A.2: Each volume of the time series undergoes a view transformation. Afterwards the pipeline splits into the conventional volume composition and the tile casting. Each tile is subdivided into a user defined number of slabs. For each subdivision a representative set of TIC is computed and stored in the TICs cache. In the next step the curves for the graxels are generated. Finally, the volume rendering image and the graxels are optionally blended.

4D data as well by defining a transfer function which captures the temporal aspect as well as the intensity values of the data. Such a technique was presented by Balabanian et al. [29], where a temporal style transfer function was defined and used in volume rendering in order to show the temporal development of the data.

While such an approach provides a useful overview, it lacks detail information of the temporal development and can suffer from occlusions. In our approach we do not combine the time steps and map them to local properties. Instead, we aim to use a more familiar and direct representation of the TIC as a function graph. Our approach is conceptually similar to traditional ray casting. However, instead of treating each ray separately, we subdivide the image space into small tiles. We aggregate the TIC behind the tiles in a parallel process called tile casting. Each tile is further subdivided into a predefined number of slabs, dividing the volume in depth. For each of these slabs we compute the maximal, minimal and mean value of the subdivision. These values are used for the computation of the TIC of the corresponding areas in the volume. The tile casting is performed for each time step. An overview of our approach is shown in Figure A.2. The tiles are grouped to graxels resulting in an aggregated set of minimal, maximal, and mean TICs. In the next step we generate the geometry of the TICs for each graxel. At this point the two parallel processes (volume composition and tile casting) join again and the TICs are displayed over a standard animated volume rendering. To investigate the data further, we allow interactions and selections in the spatial as well as in the temporal domain.

In the following we will discuss the main components of our approach. We subdivide the description of our method into three main parts: *tile casting*, *graxel rendering*, and *interaction*.

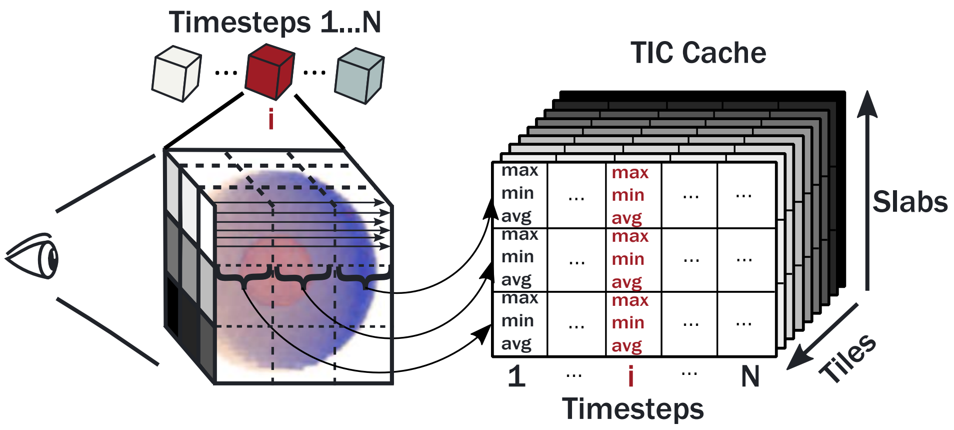


Figure A.3: For each time step we shoot multiple rays for each tile. The rays are subdivided into slabs, the intensity values in those slabs are aggregated into an envelope of the maximal, minimal and the mean value. This process is done for each time step.

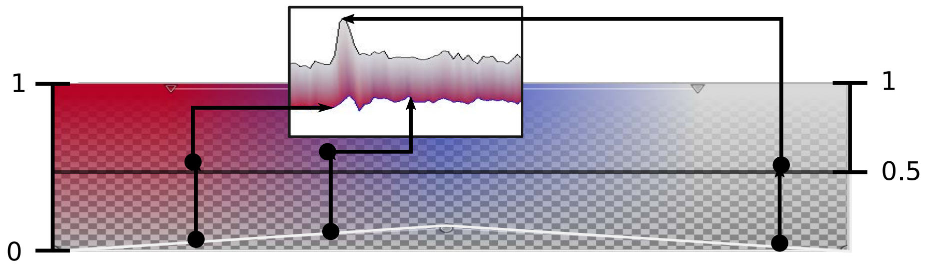


Figure A.4: The color and opacity for the line strip are looked up in the transfer function. The opacity is mapped from the interval [0,1] to an interval $[\alpha_{min},1]$. In our examples we used $\alpha_{min} = 0.7$. The mean curve is not displayed in this figure.

A.3.2 Tile Casting

As outlined in the previous section, our aim is to represent the temporal development in the value domain as familiar function graphs embedded in their respective spatial context. Our approach is to subdivide the image into a set of graxels. Each graxel contains a set of tiles. Each tile consists of fixed number of pixels arranged in a screen filling way, without overlapping each other. One can think of a graxel as a sensor which scans the volume as a projection of the covered image area. To increase the level of detail one can either use a finer sensor, i.e., a smaller number of pixels, or zoom into the data. If the camera position is inadequate, one can rotate the data to find the best viewing transformation, and the sensor will immediately adjust and record from the new viewing position.

We aim at being able to process streaming data as well as recorded data, and hence use an incremental approach which processes incoming volumes as they arrive. We summarize the tile casting as follows: For each time step the corresponding volume undergoes the volume rendering pipeline. In addition to conventional volume rendering, we aggregate the rays for each tile. In order to reduce memory consumption as well as to allow for interactive computation times, we evenly subdivide the volume behind the tile into a predefined number of slabs. For each slab we compute the maximal, minimal and mean value of the data covered by the slab and save those values in a TIC cache, see Figure A.3. The user can change the number of slabs at any point, in this case the TIC cache is emptied and the computation is restarted.

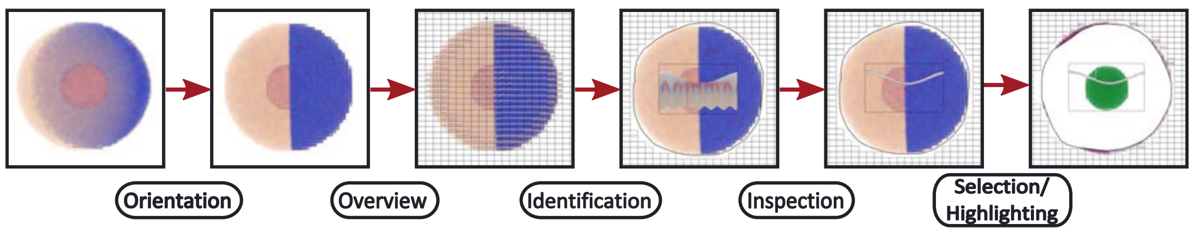


Figure A.5: The first interaction step is orientation, where an interesting view is found. After the orientation the graxels fade in to provide an overview of the TICs. Using the overview the user can identify regions of interest and enclose them with the focus lasso to inspect the TICs. If desired the TICs can be selected, alternatively the user can highlight regions with interesting temporal development.

After each tile and each slab are processed, we continue with the next volume. We repeat this procedure for each new time step until an already processed volume appears again and we completed a loop (in case of recorded data). The result is an envelope of the maximal and minimal value and the mean TIC for each slab of each tile, i.e. the curves represent aggregations in screen space and depth. While one could compute more statistical properties, the maximum, minimum, and mean are easy to interpret and have shown to be well-suited for characterizing the variation of the TICs within the aggregation regions.

In practice, for performance and memory reasons, our algorithm operates in two passes. First, we perform tile casting for a fixed tile size of $x \times y$ (where $x = y = 9$ throughout the paper). We then aggregate these tiles into graxels with $u \times v$ tiles, which are also stored in the TIC cache. The values of u and v can be freely chosen by the user and can be changed at any point. For the examples in the paper we use $u = 4$ and $v = 3$ to generate graxels of 36×27 pixels which results in a good trade-off between spatial resolution and readability for standard screen resolutions.

A.3.3 Graxel Rendering

Having generated a view-dependent representation of the spatio-temporal behavior of the underlying data over the area of each graxel, we now proceed to map this representation to visualization primitives. While there are many potential choices, we focus on a familiar and effective depiction of the TICs as line graphs where the horizontal axis represents time and the vertical axis represents the data value.

The graxel rendering can be subdivided into three stages: *curve filtering*, *curve positioning* and *curve rendering*. The *curve filtering* phase determines which curves will be shown. We provide several methods for selecting the curves in the image plane as well as in depth. After filtering of the curves, their position and size is determined in the *curve positioning* phase. All curves are displayed scaled to the size of their respective graxel. In the *curve rendering* phase the appearance of the TIC is determined. For each graxel we render the envelope of the maximal and minimal TIC as a line strip.

To establish a visual link between the graxels and the volume rendering, we color the line strip between the maximal and minimal value according to the corresponding colors of the maximal and minimal value on the graph for each time step. This means that for each time step the line strip consist of a color transition between two colors corresponding to the maximal and minimal value of the underlying graxel. This process is illustrated in Figure A.4. The curves for the maximal, minimal and mean TIC are rendered over the line strip in constant colors. We use blue for the minimal values and red for the maximal values as default colors as red and blue carry the association of being on the opposite end of the color spectrum. The mean TIC is displayed in light gray. Other colors can be chosen as well. For opacity, we map the alpha values of the transfer function from $[0,1]$ to $[\alpha_{min},1]$, where α_{min} is a positive real number between 0 and 1 with 0.7 as default value (see Figure A.4). Changing α_{min} allows the user to control to which degree parts of the curve that are classified as transparent in the transfer function will be visible in the rendered graphs.

A.3.4 Interaction

The typical interaction with our technique can be subdivided into five steps, illustrated in Figure A.5. The first step is *orientation*. During the orientation phase the graxels are transparent, which leads to exactly the same interactions as with conventional volume rendering. After a suitable view is found and no further rotation, zooming or translation is performed, the graxels start to fade in and the *overview* phase begins. During the overview phase the TICs of the whole volume can be inspected. In order to change between the different slabs the arrow buttons are used. Alternatively we allow the display of a user defined selection of slabs, in this case the user can select the slabs in a GUI and the browsing with arrow buttons is disables. In this way regions of interest can be *identified*. After the identification of interesting regions, they can be *inspected* in more detail. As last step we allow *selections* in the temporal domain as well as *highlighting* in spatial domain. If a volume interaction occurs, the graxels will immediately fade out and fade in again as soon as the interaction stops.

One advantage of our approach is that it allows us to seamlessly integrate spatial and value domain selections. Our implementation features a wide variety of different interaction mechanisms, but here we focus on two primary tools. We allow selections in the spatial domain on the screen with the *focus lasso* and further interactions in the value domain with the *curve selector*. Our main aim is to enable the selection and extraction of temporal features, allow the highlighting of events with direct links to their spatial representation, and to provide mechanisms for grouping of and separation between different classes of temporal or spatial features. With the focus lasso we select and group the graxels by simply drawing an outline on the screen. All graxels that are enclosed by this lasso are then displayed in one single canvas (see Figure A.6). To enable comparisons between different regions, multiple simultaneous focus lasso selections are possible.

Focus Lasso: The focus lasso is used after the *identification* phase and enables spatial grouping of several tiles by drawing a simple sketch. We create a binary mask from the convex hull of the selection where all selected pixels are set to one. From this mask, we create a global lasso region texture which keeps track of all the defined focus lasso regions and their intersections. This lasso region texture is used in the curve rendering phase. All graxels that are covered by a focus lasso region by at least 50% are displayed in a large common canvas inside the focus lasso. A focus lasso region is ignored if it is smaller than one initial graxel, as it would show less detail compared to the initial representation.

Focus lasso regions covering large areas contain more curves, which can lead to

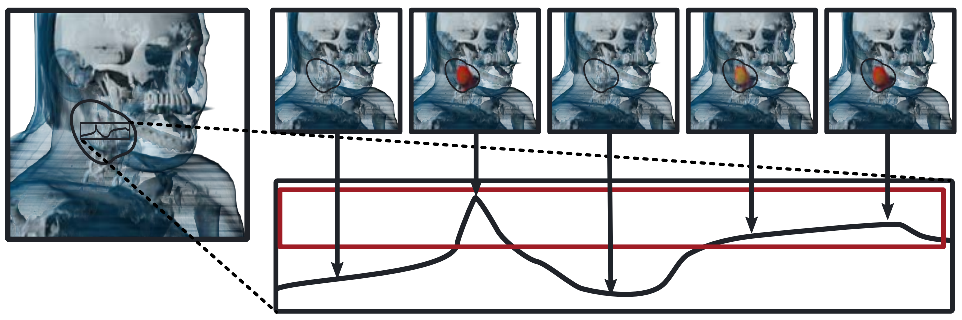


Figure A.6: Selections in the spatial domain (upper left) cover specific spatial areas and all the time steps. Selections in the value domain (lower right) cover specific spatial areas and only specific time steps.

increased overdraw resulting in reduced readability. We use the following two strategies to provide a simple yet powerful *inspection* tool. We *map the opacity* of the time-intensity curves from the space $[\alpha_{min}, 1]$ to the space $[\hat{\alpha}_{min}, \hat{\alpha}_{max}]$, where both $\hat{\alpha}_{min}$ and $\hat{\alpha}_{max}$ can be chosen between zero and one. This mapping is done in order to allow similar time-intensity curves to form visual clusters. If the time-intensity curves have similar intensity values, they will mostly overlap, thus forming opaque visual clusters. We use a *point of interest* inside the focus lasso region. This point is located at the mouse position and can be fixed on the screen. Time-intensity curves closer to the point of interest are displayed more opaque while the ones further away become linearly more transparent until they reach zero opacity. The slope of the opacity transition can be freely modified by the user.

Curve selector: We aim at interactions that allow for the *selection* of curves in the spatial as well as in the value domain. The focus lasso already groups the graxels spatially. In order to perform selection on a finer scale, we provide a mechanism that only selects graphs above a certain opacity threshold. By hovering over a focus lasso region with the point of interest until the desired TIC is prominently shown and fixing the point of interest, we allow fine-scale filtering in the spatial domain. The selection in the value domain is performed by drawing a rectangle or a circle directly over the TICs. If a TIC crosses the drawn area, it is extracted and shown in a separate window.

Some application scenarios require knowledge about the position of specific temporal features. In such a case we employ a highlighting mechanism which allows to highlight regions of the volume based on their values over time. Similar to the extraction of the graphs, a rectangle or a circle can be drawn directly over the graphs in a focus lasso region. All intensity values that are covered by this area at the corresponding timesteps will use a second transfer function, which can be defined freely, see Figure A.6. This new transfer function can be applied to the current focus lasso, to all focus lassos, or to the whole data set. Furthermore, it is possible to render the non-selected values completely transparent for detailed inspection of the exact spatial position of the values of interest. An example is shown in Figures A.10 and A.12 and described in Section A.5.2.

Through the combination of spatial and temporal selection in a single view, the curve selector provides a useful tool for selection and feature extraction. We allow grouping of features in the temporal and spatial domain with the focus lasso. Temporal events can be emphasized by selecting regions in focus with the focus lasso and then

Data set	Conv. Ray Casting	Graxels 10 slabs	Graxels 5 slabs
Synthetic data	0.049 s	0.114 s	0.079 s
Kidney simulation	0.045 s	0.143 s	0.086 s
Perfusion data	0.057 s	0.219 s	0.127 s
Supernova	0.127 s	0.354 s	0.148 s

Table A.1: Performance of our method compared to conventional direct volume rendering as measured on an Intel CPU equipped with an NVidia GeForce GTX 780 GPU, with a viewport size of 987 × 967 pixels and graxels of size 4×3 (36 × 27 pixels) with 10 and 5 slabs and one slab rendered.

highlighting the volume with the curve selector.

A.4 Implementation

Our method was implemented in C++ and OpenGL/GLSL. The tile casting approach is performed in two steps. First, the aggregation of data values into the TIC cache occurs in a GLSL shader using atomic operations of the GL_ARB_shader_image_load_store extension. Further aggregation over graxel areas is performed in a compute shader. The TIC cache itself is represented as a 2D texture array where each texture corresponds to one tile. Each row in this texture corresponds to one TIC, and the number of rows is the number of slabs multiplied by three (for the minimum, maximum, and mean TICs). We use multiple texture arrays if the number of tiles exceeds the maximum texture array size. Curve generation uses a geometry shader which performs lookups into the TIC cache.

Naturally, our method requires more computational time compared to conventional ray casting. The performance is mostly dependent on the number of total slabs and number of displayed slabs, as can be seen in Table C.1. The computation time increases with the volume size as well but this is mostly due to the increased computation time for the volume ray casting. In this table we summarize the average computation time per frame for 30 consecutive frames while displaying the TICs for one slab. Displaying TICs for additional slabs increases the computation time for each frame by 0.087 s on average. The size of graxels does not have significant impact on the overall performance.

A.5 Application Examples

We show the advantages of our technique in four application examples. We demonstrate how our methods support the understanding of frequency patterns and we document the benefit of our technique's interactivity in the context of two medical examples. Further, we show how our approach captures relevant events in the temporal domain of a case from natural sciences.

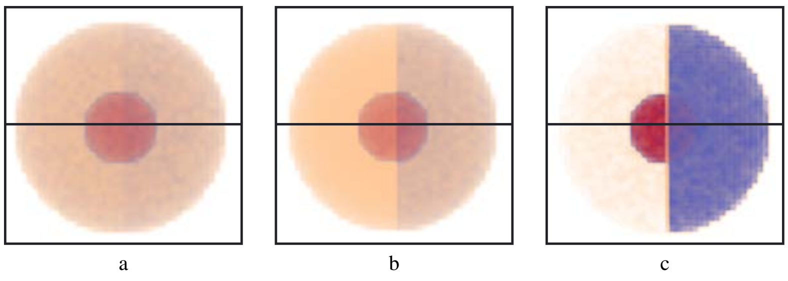


Figure A.7: Volume rendering of consecutive time-steps of two synthetic data sets. The two data sets (lower and upper half) appear almost identical in the volume rendering.

A.5.1 Synthetic Data Set

A very natural application scenario of our method is the comparison of two volumes over time. In this example we study two synthetic data sets. The two data sets are very similar in terms of their geometry. However, they show different temporal behavior. In Figure A.7 we show both data sets rendered at three different points in time, with one shown in the upper half and the other one below. Both data sets consist of two spheres, divided into a left and a right part. The two halves have time-varying intensity values described by the function $i = a + b \cdot sin(\theta + c \cdot \phi) + rand()$. The parameters a, b, θ and c define different temporal behaviors for the two halves. Inside each sphere is a spherical nucleus, with either a constant or a fluctuating value, described by the same formula. The rest of the data set consists of a constant value with random noise added to it. As shown in Figure A.7, the two data sets look almost identical in a volume rendering. In fact a thorough tuning of the transfer function is needed to see any difference at all, but the animation still fails to convey *how* the two data sets differ. The same is true for comparing the left and right sides of the spheres. Volume rendering shows that the two halves differ, but it is not clear how, especially in terms of value change frequency.

The upper data set has a stable nucleus while the lower data set has a nucleus with varying intensities. Identifying and describing the temporal behavior of each region using direct volume rendering alone is demanding and difficult. Using our technique, however, one brings the sphere in the position where the halves are clearly distinct. As the interaction stops the graxels start to fade in providing an overview. Capturing the whole sphere with the focus lasso and investigating the three interesting regions reveals the characteristic temporal relations immediately. Figure A.8 shows all time-intensity curves in one focus region (a) as well as the investigation by region, moving the point of interest, and showing the corresponding time-intensity curves (b–d). These examples demonstrate that integrating an explicit visualization of the temporal data behavior indeed helps to gain a fast overview of the temporal development in the data and different characteristic temporal developments become evident, in particular when compared to DVR alone.

A.5.2 Kidney Simulation Data

In the second example we present how our method was used by domain experts for the location of spatial features. In principle, highlighting parts of the data is possible through the tuning of the transfer function. However, if the exact value range for struc-

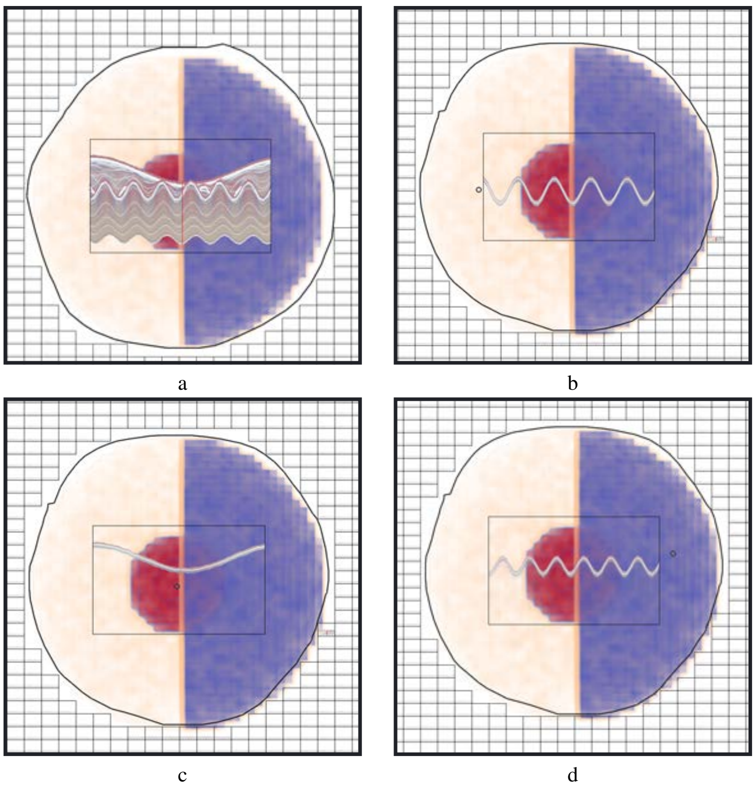


Figure A.8: In alphabetical order: All time-intensity curves in one focus region, intensity in the left half-sphere, intensity in the nucleus and intensity in the right half-sphere.

tures of interest is not known in advance, this process is difficult. Furthermore, in the case of time-dependent data, the relevant value range may change dynamically. This is the case in the following scenario.

We consider data from a numerical simulation of kidney compression during breathing. The goal of this setup was to be able to identify if a kidney is healthy or fibrotic. The latter is stiffer and thus compresses less than a healthy kidney. The simulation was done to study whether the degree of fibrosis and the location of the fibrotic tissue can be diagnosed on the basis of local volume deformation information. The data set is a 64×64 2D set with 34 time steps. A visualization of the data set, based on a sequential mutli-hue transfer-function, is shown in Figure A.9, depicting the time-step with the most prominent differences between the healthy and the fibrotic tissue. A typical task for a domain expert is the identification of the fibrotic tissue inside the kidney. However, the compression rate is highly dependent on the inhaled air volume, thus the compression values of the fibrotic tissue vary for each patient. Using a transfer function with a gradual color change helps to see the difference between healthy and fibrotic tissue but this smears out the exact location of the fibrotic tissue as depicted in Figure A.9.

In this example, the upper part of the left kidney was simulated to be fibrotic. While the visualization of the most prominent differences between the healthy and the fibrotic kidney does not require much time, these images give only little insight into the exact location of the fibrotic tissue. As only the absence of red color is noticeable in the upper part of the left kidney, one perceives the whole upper part of this kidney to be fibrotic and interprets the brighter colors as fibrotic tissue.

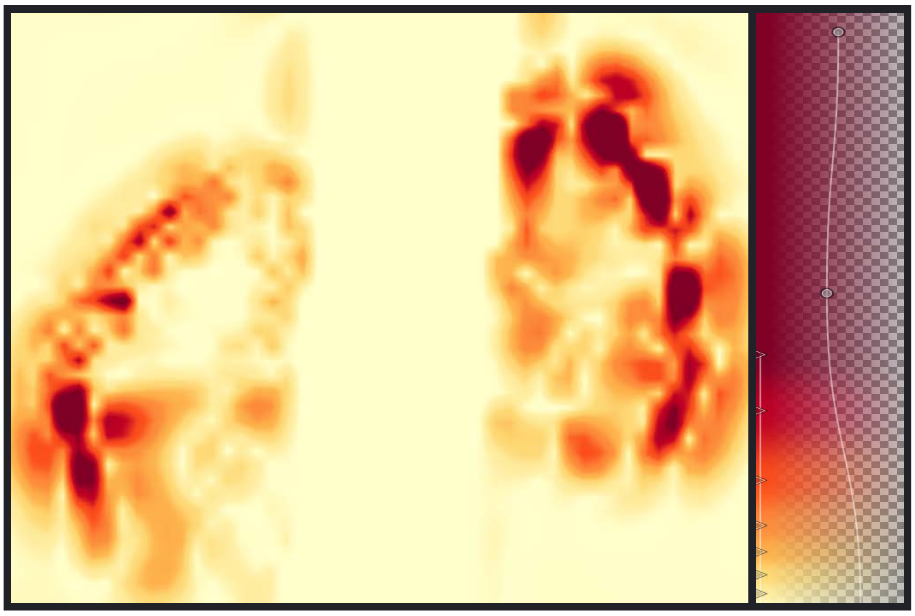


Figure A.9: The upper half of the kidney on the left side was simulated to be fibrotic, while the kidney on the right was simulated to be healthy. The transfer function is shown on the right side with low values in yellow and high values in red.

The relative volume change occurs mostly in the kidney cortex, as the rest is not compressed but moves along. Furthermore, the fibrotic cortex is not completely stiff, just stiffer than a healthy cortex, while still being more elastic than the inner kidney. Domain experts are interested in isolated regions with values in the middle range of the whole relative volume range. Selecting the kidneys with the focus lasso allows us to investigate the TICs of each kidney as shown in Figure A.10 (a). Selecting the characteristic values of fibrotic tissue allows us to render the data with a different transfer function. We immediately see isolated structures in the upper part of the left kidney, as shown in Figure A.10 (b). This way we can determine that the fibrotic tissue does not correspond to the tissue colored in yellow. Our domain experts used our tool to confirm their simulations. In the future they plan to use our approach as a promising means for detection of fibrotic tissue in real patient data, which naturally is more complex and noisy compared to the simulated example.

A.5.3 Perfusion Data

As mentioned earlier, the shape of the TICs in contrast-enhanced MRI can provide valuable information on the pathological nature of tissue. TICs can reveal relevant information on the possible malignancy of a tumor or on the functionality of an organ. Domain experts are also interested in the shape of the tissue with certain perfusion characteristics. The kidney cortex, for example, has a very high absorption rate and a very distinct shape. If the anatomical shape with particular perfusion characteristics substantially differs from the expected shape, this may also indicate a relevant organ malfunction. With our method, we can assess both criteria together by extracting and comparing TICs in multiple windows.

In this example we look at a 4D contrast-enhanced MR scan of the abdominal region. The data set consists of 48 time steps with a dimension of $61 \times 56 \times 48$. Some-

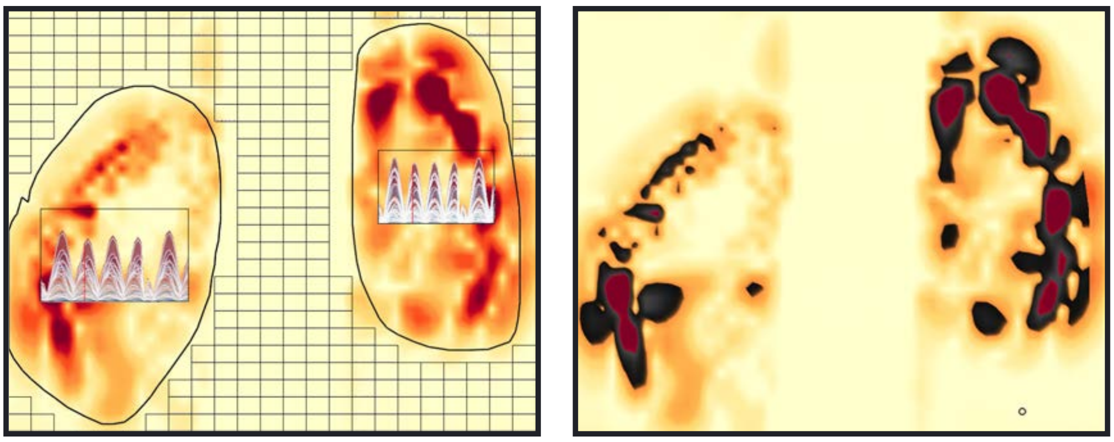


Figure A.10: a: Selecting the kidneys with the focus lasso allows to explore the TIC of each kidney and to select and highlight value range characteristic for fibrotic tissue. b: Coloring the characteristic values of fibrotic tissue in dark gray, we identify real fibrotic tissue as isolated patches of dark grey, this reveals that only the kidney cortex is affected by fibrosis.

times practitioners are interested in the comparison the perfusion rate of the two kidneys, to find out if one kidney may have a malfunctioning filtering process. The standard procedure for this is to select a point of interest in each kidney and to compare the two TICs. However for this one needs to know a priori where to set the point of interest. Furthermore, such a process is sensitive to outliers. Our approach helps to identify the region of interest and provides additional robustness compared to the standard procedures. In Figure A.11 we show the extracted TICs from the left and right kidney. The kidneys are roughly outlined with the focus lasso. Then the point of interest is fixed and a circle is used to make a selection in the value domain. Only curves that cross the circle are selected. Now one can compare the perfusion rates of both kidneys in the windows on the right side. We can observe that this data set was obtained from a healthy volunteer where both kidneys show regular perfusion rates.

As mentioned before a practitioner might be interested in the shape of the kidney cortex with certain TIC characteristics. Seeing the time-intensity curves of the kidney cortex, we can highlight the voxels with the corresponding values. By selecting their outline with the focus lasso, we can hover over a characteristic time-intensity and then select the high values with the rectangle tool. Now we can pause the animation and fix the time step, which corresponds to the peak of the perfusion TIC. This selection can be seen in Figure A.12 (a) . After this selection we can hide the graxels and observe the shape of the kidney cortex more clearly, see Figure A.12 (b).

A.5.4 Supernova

The three previous examples showed data which was to a certain degree fixed in space, with little movement of the structures themselves. In this example we look at a highly dynamic data set of a 4D simulation of a supernova, depicted in Figure A.13. Each time step has the dimensions $432 \times 432 \times 432$ with four bytes per voxel. We used the first 20 time steps of the data set with intensity values representing the entropy field.

This example demonstrates our approach as an Eulerian view of flow very clearly.

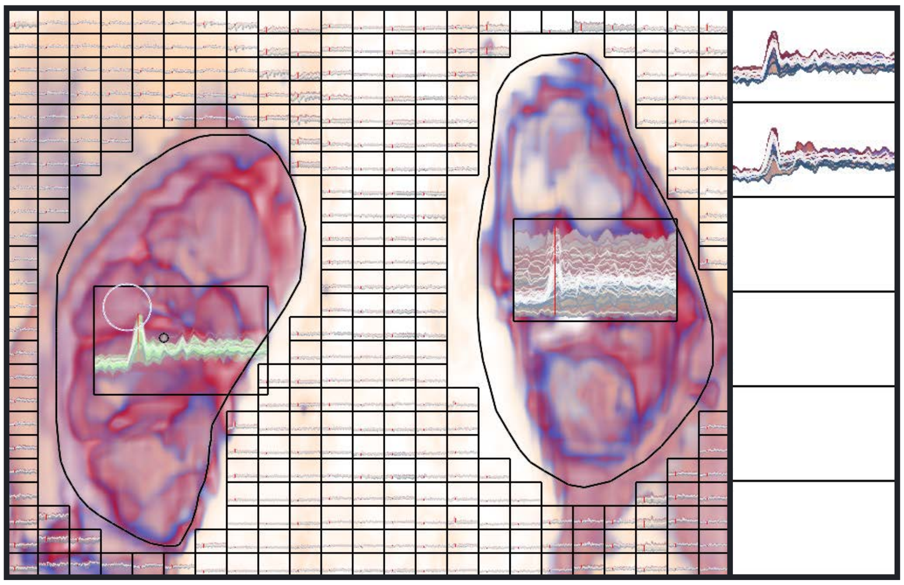


Figure A.11: The TICs from both kidneys are extracted and displayed in separate windows. We zoomed in on the left kidney in order to show the selection in more detail.

The TICs separate into several types, of which three are particularly distinct. These characteristic TICs are shown in more detail below the actual visualization, indicating that there are three regions with characteristically different temporal behavior. The region on the left shows high entropy in the first time steps, with a decrease towards the end. The region in the middle shows a high entropy field throughout the whole time period. The region on the right shows relatively low entropy in the beginning and high entropy in the last time steps. With volume rendering, we can confirm these developments, but multiple renderings are necessary, while our method captures this complex behavior in one view.

We do not claim that our method depicts the data more clearly than a volume rendering, which is essentially a Lagrangian representation of the flow, in this example, but it provides a fast overview of the whole time series in one image and manages to characterize the distinct areas with different behaviors. As such we see our technique as complementary to volume rendering, as it provides a quick overview of the data for identifying potential features of interest.

A.6 Discussion

In our experiments we found that using *graxels* in a screen filling manner provides a fast overview of the temporal development of the data. Distinct events manifest themselves as peaks in the TICs, which form visual clusters among the graxels. The focus lasso and curve selections allow fast and intuitive exploration of the data. Due to the choice to use an envelope of the TICs and the mean TIC rather than rendering all TICs at once, we reduced clutter but still provide some statistical information of the data. We do not see our method as a competitor to existing approaches, such as multiple coordinated views, as we do utilize these techniques as well. Our technique is meant to complement existing approaches and to provide a different view of the data. By overlaying our visualization over an animated volume rendering, for example, the temporal

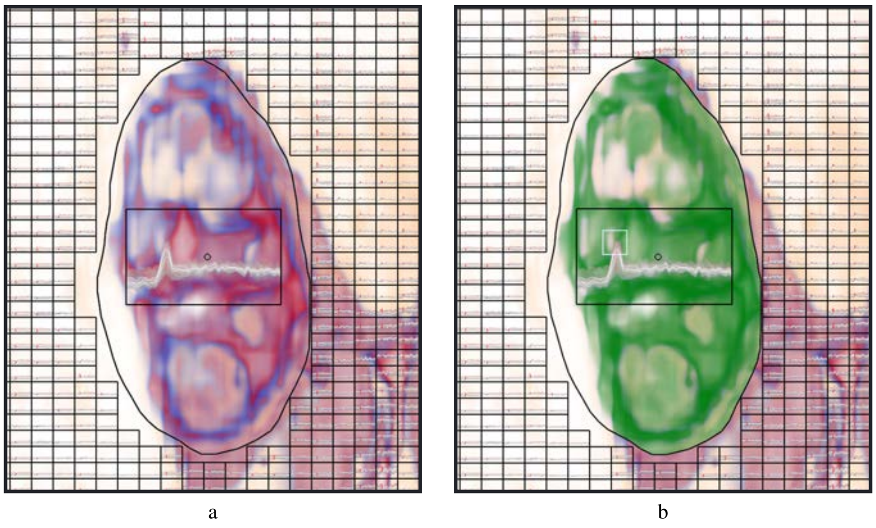


Figure A.12: By selecting the characteristic values of the kidney (a) we easily highlight the kidney in the rendered image (a, b). In (b), the volume corresponding to the selected values is highlighted.

evolution of different image regions can be made more apparent. This could also be used selectively or on-demand, for example as a type of magic lens or only during certain operations.

Our current implementation has some limitations. The need for atomic operations during tile casting is a bottleneck and limits the number of slabs when interactive performance is desired. However, tile casting only has to be performed when the view is changed. When using the presented selection methods, the performance is fully interactive. Also, when the transfer function is changed, only curve generation has to be performed. Hence, for most use cases the performance of our current implementation is sufficient and there are several options for further optimization.

While our visualization is quite dense, it nonetheless has shown to be effective in conveying even complex temporal patterns. As screen resolutions constantly increase, we believe that this type of visualization is promising to provide a concise overview of complex data. As pointed out by Tufte [215], humans are capable of a vast number of distinctions within a small area. Furthermore, one could display the graxels in a checkerboard fashion, thus cutting their amount in half. As already mentioned, hiding the graxels during interactions such as rotation and then gradually letting them fade in again can be used to improve orientation during viewpoint changes and also reduces the impact on computational performance. Since the graxels are placed in the screen space, they may not be aligned to boundaries in the spatial domain. This drawback can be overcome by either adjusting the tile size or by using the focus lasso which allows to define meaningful region borders.

The graxels are organized in slabs perpendicular to the viewing direction naturally many data sets contain structures which do not follow this orientation. These structures would have a better representation in deformed slabs, which align with the structures. We partially address these cases by allowing the user to define the number of slabs and the shown slabs freely.

We provide several interaction techniques with the data. Additional more complex

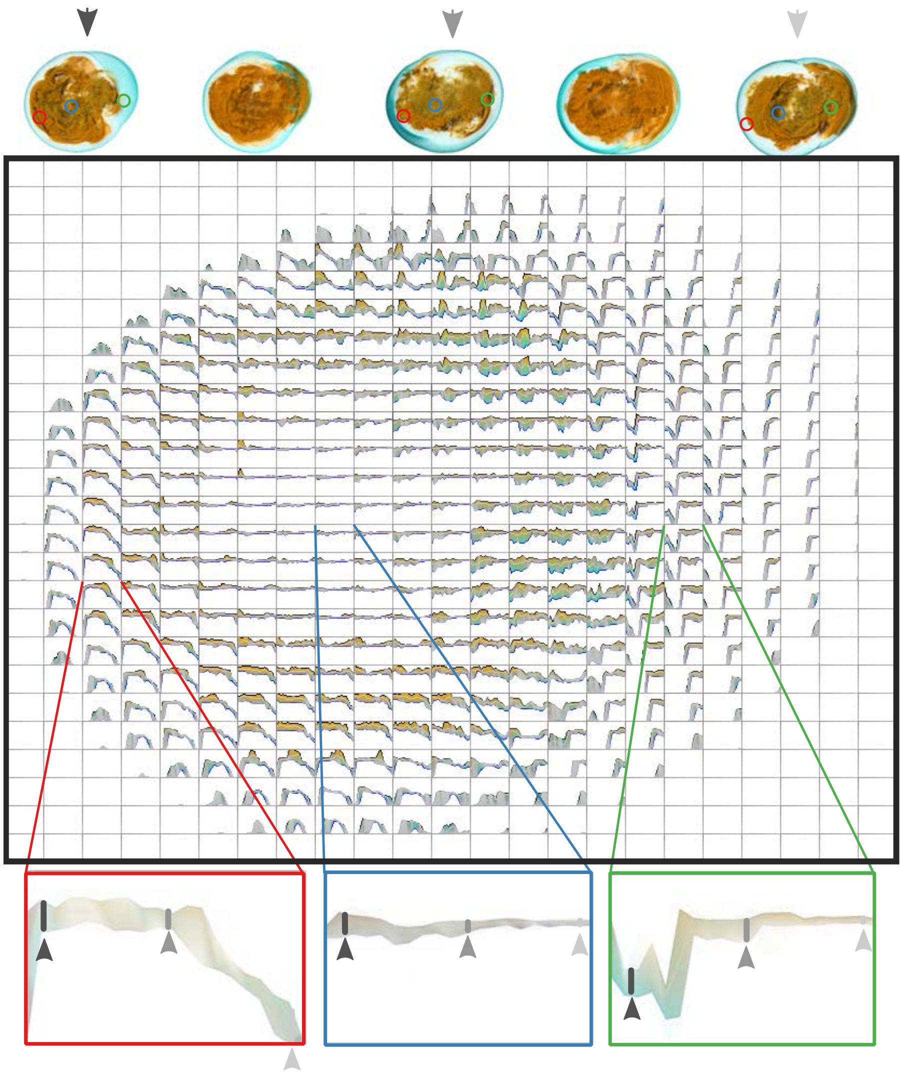


Figure A.13: Our method reveals different parts of the data set, with decreasing, constant and increasing entropy fields. These temporal developments are integrated in one single view.

techniques, such as selection of curves with a certain shape, can be easily integrated. It should be noted that our visualization is most effective when the size of the graxels is adjusted according to the screen resolution which is difficult to accurately reproduce in this paper as the maximum image size is limited by space restrictions. Hence, most of the images in this paper are best viewed by zooming in.

A.7 Conclusion

In this paper we introduced a flexible visualization technique for time varying volumetric data. Our method uses view-dependent small multiples of time-intensity curves, which we term graxels (graph pixels), to provide an overview of the spatio-temporal patterns in the data. We presented a GPU-based algorithm for generating these views and showed how they facilitate interactive exploration and analysis in space and time. Our method was designed for time-dependent volume data, but the approach is also applicable to other types of multivariate data.

A.8 Acknowledgements

This work has been supported by the MedViz lighthouse project IllustraSound.

A

Paper B

Vol^{2}velle: Printable Interactive Volume Visualization

Sergej Stoppel and Stefan Bruckner

University of Bergen, Norway

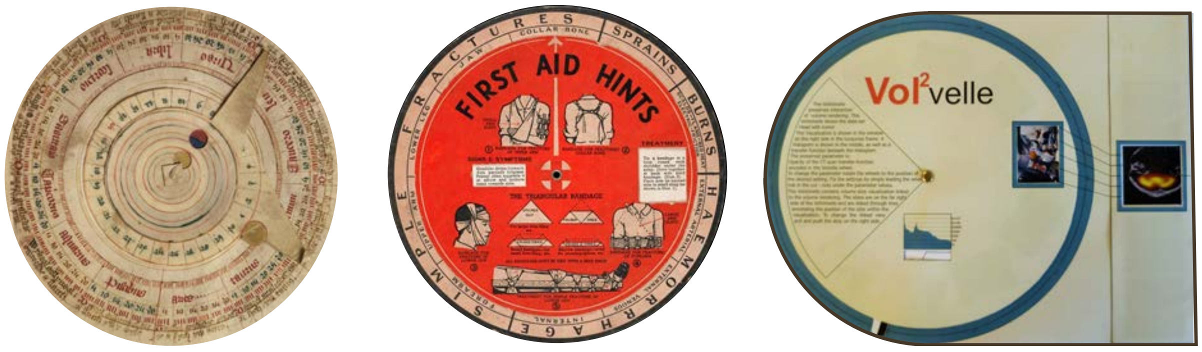


Figure B.1: Volvelles, carefully designed interactive wheel charts, have been used for centuries to present a wide variety of data. Our approach allows the simple creation of a volumetric Volvelle, or Vol^{2}velle (see rightmost image), directly from an interactive volume visualization setup. The leftmost and center image are courtesy of the National Library of Whales and the Welcome Library London (record ID = b1655695), respectively.

Abstract

Interaction is an indispensable aspect of data visualization. The presentation of volumetric data, in particular, often significantly benefits from interactive manipulation of parameters such as transfer functions, rendering styles, or clipping planes. However, when we want to create hardcopies of such visualizations, this essential aspect is lost. In this paper, we present a novel approach for creating hardcopies of volume visualizations which preserves a certain degree of interactivity. We present a method for automatically generating Volvelles, printable tangible wheel charts that can be manipulated to explore different parameter settings. Our interactive system allows the flexible

This article was published in Proceedings of *IEEE Transactions on Visualization and Computer Graphics* (2017). pages 861–870 mapping of arbitrary visualization parameters and supports advanced features such as linked views. The resulting designs can be easily reproduced using a standard printer and assembled within a few minutes.

B.1 Introduction

Volume visualization techniques for the exploration, analysis, and presentation of 3D scalar fields have been extensively studied and play an important role in many different applications such as medicine, biology, or engineering. More recently, volume visualization has also been employed in scenarios geared towards non-experts, for instance in the form of virtual exhibits in museums [110]. Interaction is often a critical component in enabling a better understanding of volumetric data through camera rotation, manipulation of the transfer function, placement of clipping planes, or the modification of light sources. When there is a need to create hardcopies of such visualizations, for instance if they are to be distributed to many people, however, the interaction aspect is lost. In this paper we present an approach for the creation of hardcopies which, to a certain extent, preserve the interactivity of the visualization. We draw our inspiration from the concept of Volvelles. A Volvelle, or wheel chart, is a type of circular slide chart. Volvelles are paper constructions with rotating parts and can be considered as an early form of an analog computer. Over the centuries, such charts have been carefully designed to accommodate scientific visualization and calculation in many different fields such as the study of planetary orbits or the presentation of multiplication tables [87]. Volvelles can be traced back to Arabic treatises on humoral medicine dating back to as early as 1000 BC and to the Persian astronomer Abu Rayhan Biruni, who made important contributions [180]. In the 20th century, mass-produced Volvelles experienced a resurgence as handy, cheap, and easy-to-use devices for supporting organization and calculation. Today, companies such as Datalizer [5] still design and produce Volvelles as custom data visualization tools for private customers and companies like AT&T, Comcast, or Verizon.

However, the creation of a Volvelle is a laborious, time-consuming, and mostly manual process performed by graphic designers. Depending on the complexity of the encoded information, the estimated time for the design process of a Volvelle ranges from several weeks to months. In this paper, we propose an approach for the instant generation of Volvelles directly from interactive volume visualization setups to create a printable, tangible volume visualization—a VolVolvelle or *Vol^{2}velle*—which still preserves a user-specified degree of interactivity. Clearly, paper-based visualizations are not meant to compete with handheld devices such as tablets or smartphones as a mobile solution to advanced interactive visualization. However, paper prints still have unique properties as they are very cheap, recyclable, and easily accessible, which makes them an attractive alternative for certain applications. Direct interaction with the material also increases the likelihood of remembering the encoded content [87, 203] and helps to understand the data better [202]. For instance, our approach can be used as a fast and easy way to generate personalized visualizations in museums or other public displays. In such a scenario, the user could interact with a virtual exhibit to design his or her own personalized visualization which can then be printed and assembled, creating a more engaging and memorable experience through the IKEA effect [158]. Other possible use cases include early education, product presentations, or marketing activities, where a paper-based Volvelle may also be used as an eye-catching way to distribute the URL to a fully interactive web-based visualization.

To the best of our knowledge, our approach is the first to employ the concept of Volvelles for the automated generation of interactive paper-based visualizations of scientific data. The main contributions of our work can be summarized as follows:

- We analyze the Volvelle design space and material constraints to identify a suitable subset for the mapping of visualization parameters.
- We introduce a set of strategies for parametrizing and encoding typical volume visualization parameters on a Vol^{2}velle.
- We present a flexible interactive approach for the semi-automatic generation of Vol^{2}velles which supports a large space of common volume visualization techniques.

Furthermore, while this paper focuses on volume data, our approach is general and can be easily extended to handle other types of visualization.

B.2 Related Work

Research inspired by art and design has a long tradition in visualization and computer graphics. The area of illustrative visualization, for example, aims to devise computer-based visualization methods inspired by traditional illustration techniques. An overview of the concept of illustrative visualization was provided by Rautek et al. [173]. A frequent common element of these approaches is the need to extrapolate from static, paper-based presentations of phenomena to interactive visualization approaches, in particular also in the context of volume data. Examples include cutaways or ghosted views [36, 220], exploded views [38], or deformation [49]. In some sense, the work presented in this paper can be seen as completing the circle by going back to paper while preserving some of the dynamic and interactive characteristics of computer-based visualization systems.

A further source of inspiration for our work comes from research on simplifying the interaction and the specification of visualization parameters. The seminal work of Marks et al. [144] introduced Design Galleries, a general concept for exploring parameter spaces by using random sampling. They applied their approach to problems such as transfer function design in direct volume rendering and lighting specification. König et al. [124] presented a user interface paradigm for semi-automatic transfer function manipulation which provides the user with suggestions and previews. The work of Rezk-Salama et al. [174] discussed high-level user interfaces for transfer function design which integrate semantic models. Ma [140] introduced a visualization system which presents information on how parameter changes affect the result image as an image graph based on data generated during an interactive exploration process. Subsequent work by Jankun-Kelly et al. [102] deals with the exploration of visualizations, but focuses on sharing and collaboration. The work of Scheidegger et al. [183] on the VisTrails system enables the creation and retrieval of visualizations by analogy using pipeline matching based on captured provenance information. Explorable images, introduced by Tikhonova et al. [211], aim to enhance the fidelity of remote visualization by recombining a set of server-generated images to allow parameter exploration on thin clients. Our approach employs several concepts as well as technical aspects of these works. In particular, our system allows for the flexible parametrization and sampling of different visualization algorithms to create a Vol^{2}velle which can then be used to further explore the data.

Several non-traditional interfaces for visualization have been presented which are related to our research. Holman et al. [93] used projections on physical paper to create interactive information displays. Spindler discussed two approaches for tangible paper displays, PaperLens [194], a spatially-aware paper display that resembles a magic lens, and Tangible Views [195], paper displays for information visualization with a set of interaction techniques. Jackson et al. [98] presented a tangible interface for thin fiber structure visualization, where the movements of a paper tube were translated into interaction input. Issartel et al. [97] used tangible volumes as hand-held fish-tank displays with pressure detection in an augmented reality visualization setup. Le Goc et al. discussed design considerations for physical visualizations [28] and introduced smart tokens [130], small tangible tokens that can sense multiple types of motion. While our work also aims to create a tangible interface for interaction, we see a clear distinction. We want to preserve interaction without electronic devices and only using standard printing equipment.

A detailed chronological list of physical visualizations without electronic devices is collected in physlist [57] of which a interactive wooden model of a 3D MRI scan [70] is conceptually closely related to our approach. While physlist does not contain any interactive paper models, the works collected there share our aim of converting digital data into physical visualizations. Several works have investigated the utility of physical visualizations. Stusak et al. [204] explored the impact of physical visualization as a reward for running activities, and also discussed physical bar charts and the information retrieval process of physical visualization [200]. A similar direction was taken by Jansen et al. [104] and Taher et al. [207]. Jansen et al. [103] also presented an interaction model for visualization beyond the desktop. Hogan et al. [90] showed the strengths of a paper-based participant-aided network diagram (sociogram) for the use in a field study and even used Volvelles and slide charts for graph representations. Huron et al. [96] performed a study on how people transform data into visualizations using tangible tokens as a basis for information visualization. The supportive character of physical visualizations and their impact on memorability was evaluated by Stusak et. al. [201–203] with the result that physical visualizations can support the analysis process and lead to significantly less information decay. We see these approaches as an inspiration to use physical interaction as an alternative medium.

An example for the generation of paper-based physical visualizations is the work of Li et al. [136] who introduced an algorithm for the automatic creation of popup illustrations from 3D models. Swaminathan introduced MakerVis [206], the first tool that integrates the entire workflow for the creation of physical visualization. Yeh et al. [242] introduced Gigapixel Prints, a set of techniques for supporting large paper media with interactive input, combining the complementary affordances of paper and digital media. Cherubini et al. [44] discussed the creation, deployment, and evaluation of a large-scale, spatially-stable, paper-based visualization of a software system.

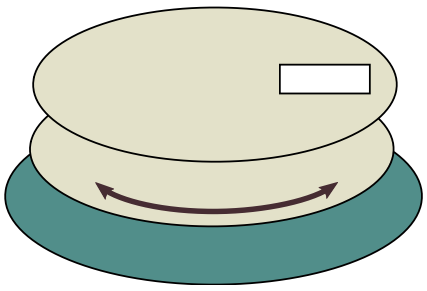


Figure B.2: The most simple Volvelle consists of a fixed front cover, a rotating wheel chart, and a fixed base.

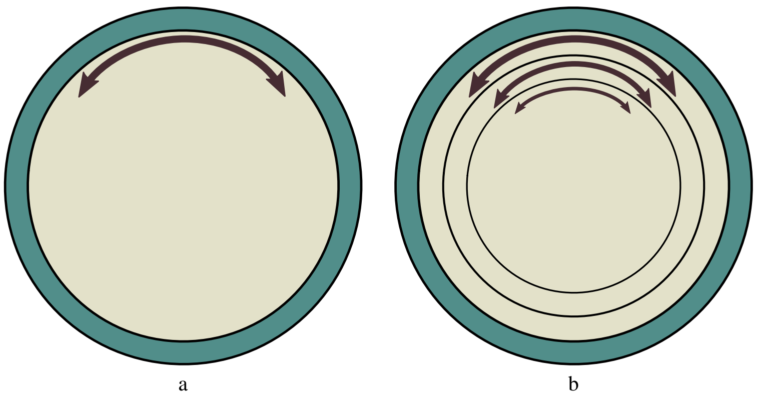


Figure B.3: The Volvelle can have either one rotating wheel (a) or multiple stacked wheels (b).

While these approaches used paper as a visualization medium, they do not focus on interaction, which is a key aspect of our work.

While the work presented in this paper shares some characteristics and goals with previous approaches which mostly focus on adding tangible interaction to computer-based systems, we see a clear distinction. Our aim is to preserve interactive characteristics without electronic devices and only using standard printing equipment, as a complement rather than a competitor to high-tech solutions for tangible and mobile visualization.

B.3 Volvelle Anatomy

Before presenting our approach for the generation of Vol^{2}velles, printed interactive volume visualizations, we want to briefly outline the possibilities and limitations of printed media. In the remainder of the paper, we use the term Volvelle to refer to wheel charts in general, while our use of the concept for the presentation of volumetric data is referred to as Vol^{2}velle. In this section we discuss the basic aspects of interactive paper models—the anatomy of a Volvelle.

A very basic Volvelle consists of three layers: a fixed front cover with a window, a rotating wheel, and a fixed base, as shown in Figure B.2. The amount of rotating wheels may vary, but most existing Volvelles consist of only one rotating wheel as depicted in Figure B.3(a) and in the examples in Figure B.1. In theory, stacking the wheels allows for more interaction degrees of freedom, but practical considerations constrain

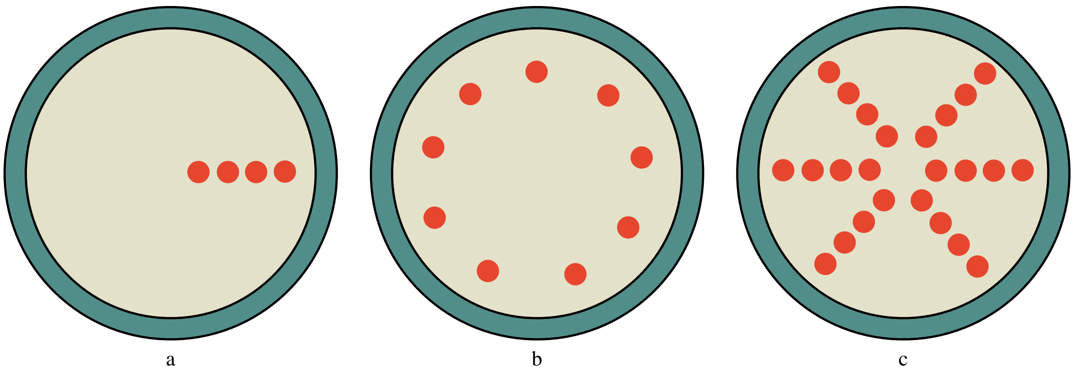


Figure B.4: The information layout on each wheel can be either concentric (a) or radial (b). Both layouts can be combined to a common concentric radial layout (c).

this amount as more wheels require additional space (see Figure B.3(b)). Moreover, each additional rotating wheel adds complexity, both in terms of assembly and with respect to the practical interaction with the final product. While the additional space requirements can be partially overcome if a transparent medium is used, the increased complexity remains a challenge. The information on one wheel can be encoded in two ways, concentrically (see Figure B.4(a)) or radially (see Figure B.4(b)). These layouts are not mutually exclusive and can also be combined as shown in Figure B.4(c).

Concentric layouts may have dynamic windows. Hence there are two possibilities for the a cover wheel, either a static cover with a fixed window for a radial layout (see Figure B.5(a)) or a freely rotating wheel with multiple windows for a concentric layout, as shown in Figure B.5(b). In the latter case, the rotating wheel can be covered by a fixed front wheel with a big window, which would hide the mechanics behind it. A circular Volvelle can also be combined with other types of interactive charts, such as a linear slide chart. A slide chart consists of an envelope-like cover with windows and a rectangular piece of paper which can be pushed and pulled through, as depicted in Figure B.6. Finally, it is also possible to use more complex Volvelle wheels. One example is a changing picture wheel chart as shown in Figure B.7. Such a chart is constructed out of two separate wheels, which are interleaved and composed in an iris-like manner such that rotating the two wheels can be used to control which one is shown. While there are several additional possibilities, we consider the outlined components to be most useful for our purposes.

B.4 Vol^{2}velle Generation

Our basic approach aims at preserving the simple and straight-forward process of regular printing from a user's point of view while enabling a flexible mapping between interaction parameters and Vol^{2}velle components. We integrated Vol^{2}velle generation as an extension to an existing volume visualization framework. In this system, the complete set of state parameters is represented by a set of properties which can be modified interactively, e.g., through mouse and keyboard commands such as camera manipulation, using user interface widgets, or programmatically. Any such modification may trigger the generation of a new output image, which can be regarded as a single sample

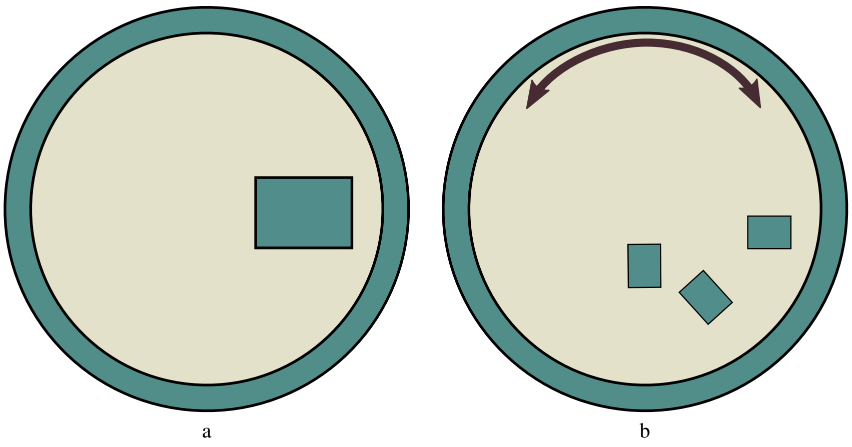


Figure B.5: A radial layout can be accomplished with a static window (a), a concentric layout requires a dynamic window (b).

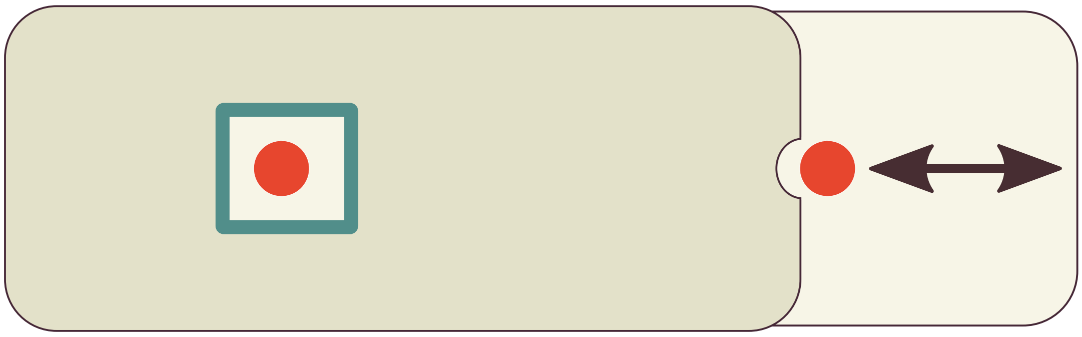


Figure B.6: Going beyond the circular layout, we can employ slide charts. A slide chart consists of an envelope-like cover and a rectangular paper piece which can be pushed through the envelope.

of this parameter space. Thus, the generation of a Vol^{2}velle involves sampling a subset of this parameter space according to a user-specified set of parameters, which are then mapped to a selected Vol^{2}velle model. A conceptual overview of this process is depicted in Figure B.8.

The user specifies a desired Vol^{2}velle model (see Section B.4.1) which provides a number of slots for visualization parameters that can be mapped. Next, the user chooses the desired set of parameters by assigning them to the available slots. As these parameters may be of any type, our approach offers a set of automatic parametrization functions to reduce complex parameters (e.g., transfer functions or rotation matrices) to scalar values as detailed in Section B.4.2. This determines the degree of interactivity that will be preserved in the Vol^{2}velle. Based on the Vol^{2}velle model and the parametrization, our system then creates a set of samples, i.e., output images of the visualization setup (see Section B.4.3). After the sampling process is complete, the samples are passed to the layout generator which is responsible for arranging the images on the available Vol^{2}velle elements (see Section B.4.4). This step produces the final printable document which consists of a template with cutting and assembly annotations.

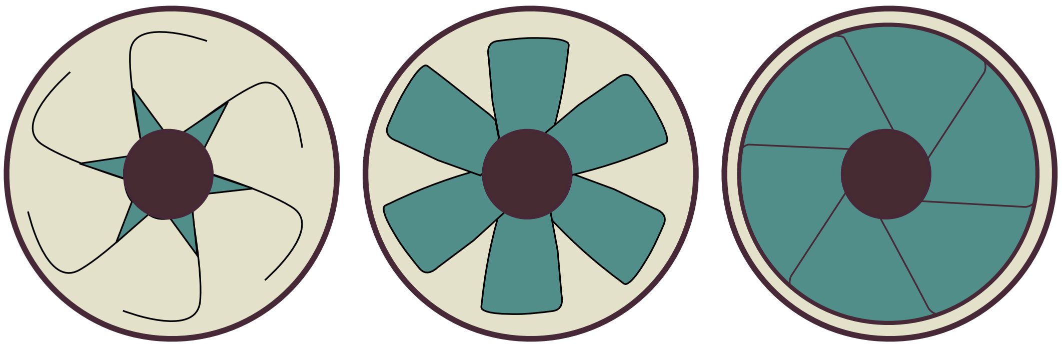


Figure B.7: By interleaving two wheel charts we can create a changing picture Volvelle which changes the appearance through rotation.

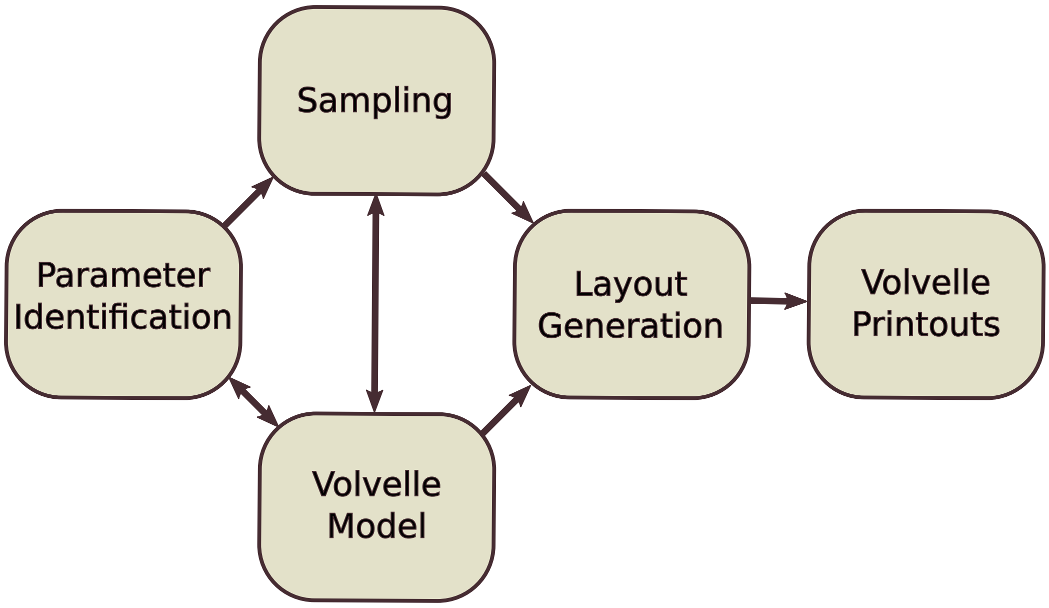


Figure B.8: Conceptual overview of our approach for Vol^{2}velle generation.

B.4.1 Vol^{2}velle Models

While we already discussed the basic components of a Volvelle in Section B.3, in this section we take a closer look at how this concept can be employed as a representation of a discretized function of multiple scalar parameters. Our approach allows users to choose between multiple Vol^{2}velle models which constrain the number and types of interactive parameters that can be represented.

We distinguish between a *paper model* and a *transparent model*. The paper model consists entirely of paper, while the transparent model uses transparent film for the individual wheels. In the case of the paper model, we can encode one- or two-dimensional parameter spaces, through a radial (Figure B.9(a)) or a radial/concentric (Figure B.9(b)) layout. The radial and concentric arrangements differ in two ways. First, the number of possible radial samples is typically larger than the number of concentric samples as more samples can be arranged around a circle than along the radial line. Furthermore, the number of radial samples increases with the radius. This means that if a uniform sampling of the parameter space is desired, space on the wheel will be wasted. In addition, we offer the possibility of using a *changing picture option*, which enables the

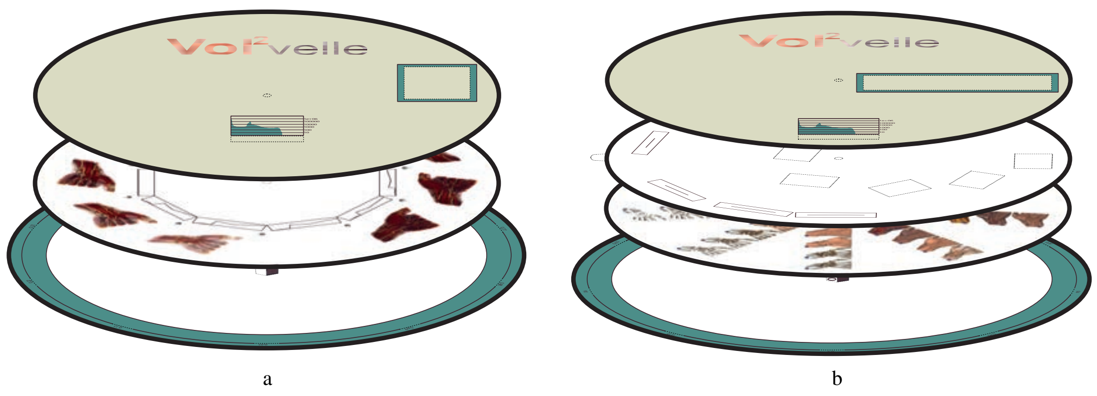


Figure B.9: The information on the wheel can either be encoded in one (a) or two dimensions (b).

encoding of a further binary parameter. The assembly of a changing picture Volvelle is illustrated in Figure B.10(a).

The transparent model consists of transparent film. This is particularly useful in the context of volume visualization, as such a *transparent model* can be used to overlay multiple independent layers. The final volume visualization can be composed similar to the multiplane camera used in early animation movies as shown in Figure B.11. The resulting assembly is shown in Figure B.10(b). As the order of the wheels is fixed, the presented data needs to follow the spatial ordering in the volume if correct occlusions are desired. While it would be possible to employ a decomposition similar to the work of Tikhonova et al. [211], we use a simpler approach where the layers are sorted according to their shortest distance to the eye point.

Finally, we allow a *slide chart extension* for both the paper model and the transparent model. The slide chart encodes one additional independent scalar parameter. It is used in addition to the Volvelle wheels to provide further information such as linked views. The slide chart is attached to the Volvelle base as depicted in Figure B.12. A natural choice for the slide chart is an additional slice view, but our system allows the use of arbitrary 2D and 3D visualizations.

In principle, our Vol^{2}velle allows for the representation of all three tabular data types: categorical, ordinal and quantitative data. Ordinal and quantitative data can be treated similarly, hence we simply refer to them as ordered. We can summarize the Vol^{2}velle models in the following way:

- Paper model
  - 1 Ordered Cyclic Parameter (radial layout)
  - 1 Ordered Sequential Parameter (concentric layout)
  - 1 Categorical Parameter (changing picture wheel)
- Transparent model
  - 3 Ordered Cyclic Parameters (radial layout)
  - 3 Ordered Sequential Parameters (concentric layout)

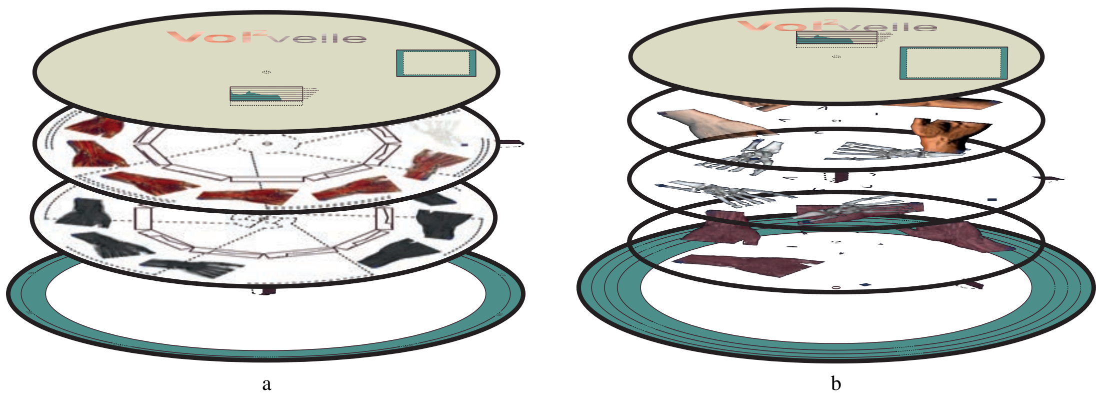


Figure B.10: (a) The wheels can be interleaved to a changing picture Volvelle. This introduces a new binary parameter but makes the assembly process slightly more complex. (b) Using a transparent medium allows superposition of multiple images.

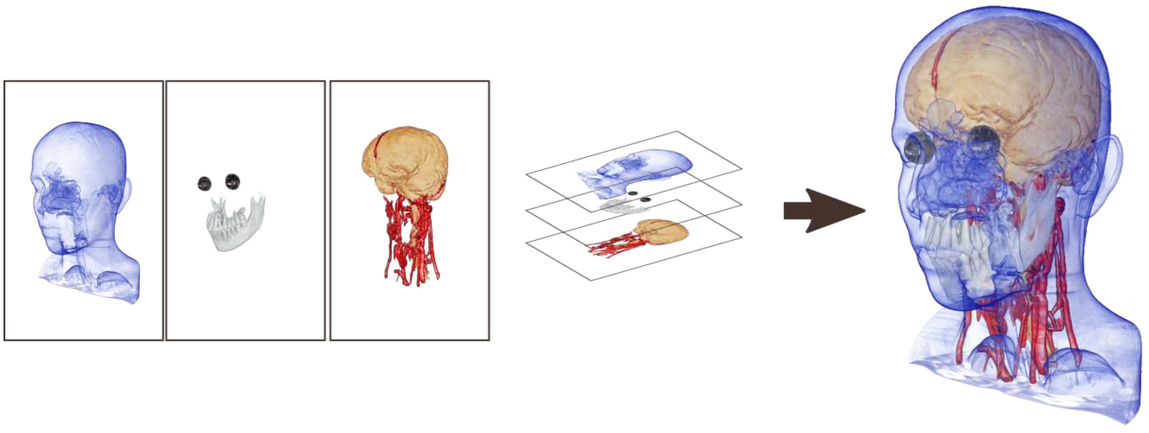


Figure B.11: The final image is composed out of individual layers.

- Slide chart extension
  - 1 Ordered Sequential Parameter (slide chart)

In our system, the user can choose between the *paper model* and the *transparent model*, select the number of desired wheels and enable or disable the *changing picture* and *slide chart* options. The result is a number of slots which can then be mapped to the desired interaction parameters. This choice depends on the intended result. If the user wants an approximation of volume rendering through layer composition then a transparent model is selected. The paper model can be used more freely. However, the user has to find a suitable mapping of the parameters. For example, a mapping of the cyclic parameter to the light position and the sequential parameter to a transfer function setting is more natural than mapping the light position to a sequential parameter. We discuss the parameter mapping in detail in the next section.

B.4.2 Parametrization

One crucial part of our approach is the conversion of an interactive visualization setup to a discrete Vol^{2}velle. In principle, we can regard this as a mapping between a visu-

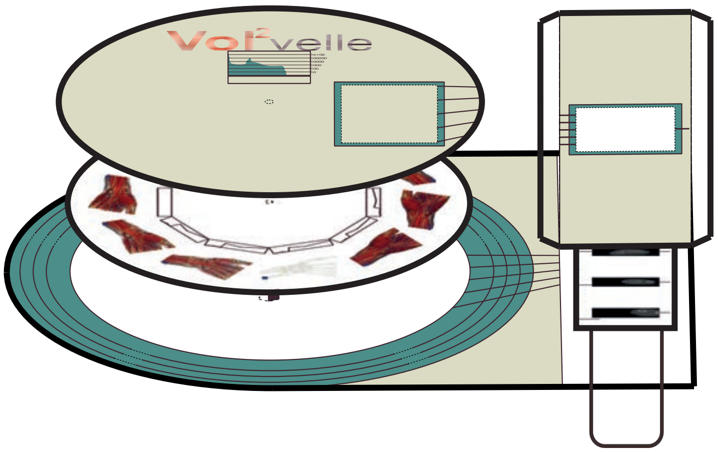


Figure B.12: A slide chart used for a linked view of volume slices.

Visualization space V	**Vol^{2}velle space** V'
High dimensional interaction space	Low dimensional interaction space
Continuous parameter variation possible	Discrete parametrization only
Visualization resolution/size is independent of interaction	Trade-off between the visualization size and parameter density

Table B.1: A brief characterization of the Visualization space and the Vol^{2}velle space.

alization space V and a Vol^{2}velle space V'. We characterize the spaces in Table B.1. Evidently, the visualization space is by far more complex and richer than the Vol^{2}velle space, and hence some fidelity will inevitably be lost. More formally, we can regard the two spaces V and V' as functions to an image domain and the Vol^{2}velle generation can be characterized as follows:

$$\begin{array}{ccc} V & \longrightarrow & Im \\ \uparrow F & & \downarrow S \\ V' & \longrightarrow & Im' \end{array} \tag{B.1}$$

As constructing a direct mapping from V' to the Vol^{2}velle image space Im' might be very complex, we subdivide the mapping into simpler parts. We first map the low dimensional Vol^{2}velle space to the complex visualization space. In the visualization space the mapping to the visualization images Im is already established. Finally, a sampling process S automatically converts the visualization image space Im according to the mapping F to the Vol^{2}velle image space Im'.

In principle, F could be constructed in a number of ways. For instance, one approach could be to employ a form of dimensionality reduction such as multidimensional scaling. However, as the user will typically have some specific communicative intent in the generation of a Vol^{2}velle, we propose a semi-automatic process which provides more control and is comparably easy to understand. We define F as:

$$\begin{array}{lr} F = (f^P_{Wh}, f_{Sl}) & \text{for the paper model} \\ F = (f^T_{Wh}, f_{Sl}) & \text{for the transparent model} \end{array} \tag{B.2}$$

where F_{Wh} denotes a mapping to the Vol^{2}velle wheel and F_{Sl} is a mapping to the slide chart. We furthermore distinguish between the paper model P and the transparent model T. In practice this means that the user has to define up to two functions f_{Wh} and f_{Sl}. The functions f_{Wh} and f_{Sl} are implemented as collections of 1D functions, i.e., the user selects parameters in V to which f_{Wh} and f_{Sl} map. Furthermore we allow some complex parameter ensembles, like settings of a transfer function, to be interpreted as a 1D parameter.

We have chosen to parametrize the Vol^{2}velle space on the unit interval. Each of these 1D function can have the parameters described in Section B.4.1 as its input parameters. More formally we define:

$$f^P_{Wh,k} : \{[0,1],[0,1],\{0,1\}\} \longrightarrow \mathbb{R} \tag{B.3}$$

$$f^T_{Wh,k} : \{[0,1]^3,[0,1]^3\} \longrightarrow \mathbb{R} \tag{B.4}$$

$$f_{Sl,k} : [0,1] \longrightarrow \mathbb{R} \tag{B.5}$$

where k denotes the index of the 1D function. The possible input parameters of these functions are described in Section B.4.1—they can be either one ordered cyclic parameter, one ordered sequential parameter, or one categorical parameter for the paper model. The six parameters of $f^T_{Wh,k}$ are the three ordered cyclic and three ordered sequential parameters of the transparent model. The single parameter of $f_{Sl,k}$ is the ordered sequential parameter of the slide chart.

Generally speaking every parameter of V can be discretized with a parameter of V' and the user is free to define these mappings. However, the nature of the Vol^{2}velle parameters leads to some natural mappings. For example, a Boolean parameter in V presents a perfect match with the binary parameter of the paper model. The cyclic parameter in V' can be naturally mapped to the angle of a light position. Many natural mappings can be found between V' and V, but of course this is dependent on the specific setup of the interactive visualization. We identified a set of functions as a meaningful selection of predefined mappings applicable to many common types of parameters in volume visualization:

$$p'_i \cdot (v_{max} - v_{min}) + v_{min} \tag{B.6}$$

$$\begin{cases} v_0\text{: for } 0 \\ v_1\text{: for } 1 \end{cases} \tag{B.7}$$

$$v_{span} \cdot sin(p'_i \cdot 2\pi) + v_{min} \tag{B.8}$$

$$v_{span} \cdot cos(p'_i \cdot 2\pi) + v_{min} \tag{B.9}$$

$$(p'_i)^{v_{pot}} \cdot v_{span} + v_{min} \tag{B.10}$$

where the numbers v denote user-defined values and p'_i is the $i-$th input variable of the function. Equation B.6 is a simple linear mapping between two values v_{min} and v_{max}—such a function is well suited for a mapping to opacity values. We use this function as a preset mapping. Equation B.7 defines a Boolean switch function for the changing picture Vol^{2}velle, but note that the values v_0 and v_1 are not required to be Boolean themselves. This can, for example, be used to map different light or cropping plane positions. Equations B.8 and B.9 can be used for cyclic mappings and are therefore well suited for the radial parameter on the Vol^{2}velle wheel. Equation B.10 allows for a nonlinear mapping between two values v_{min} and $v_{max} = v_{min} + v_{span}$. Such a mapping can be used for parameters with nonlinear behavior. It is possible to create more complex functions by combining the above operations or to use multiple input variables in one mapping. An example could be a light setting on two axes which can be controlled via two parameters on the Vol^{2}velle.

In addition to these basic arithmetic functions we provide two convenience functions which treat a predefined transfer function as a 1D parameter. These functions are:

$$Tinterpol_1(p'_1, trf_1, ..., trf_k) \tag{B.11}$$
$$Tinterpol_2(p'_1, p'_2, trf_1, trf_2, trf_3) \tag{B.12}$$

$Tinterpol_1$ linearly interpolates k transfer functions in a cyclic manner, i.e., it constructs a transition:

$$trf_1 \rightsquigarrow trf_2 \rightsquigarrow trf_3 \rightsquigarrow ... \rightsquigarrow trf_k \rightsquigarrow trf_1. \tag{B.13}$$

$Tinterpol_2$ interpolates between three transfer functions with two input parameters as:

$$trf = \left(trf_1 \cdot p'_1 + (1 - \cdot p'_1) \cdot trf_2\right) \cdot p'_2 + (1 - \cdot p'_2) \cdot trf_3 \tag{B.14}$$

The amount of possible predefined functions is of course countless. However, the presented set has proven sufficient for many common cases. The user also has the possibility to define custom functions via an integrated scripting language, which may also use one or several of the predefined functions. As all parameters have types and associated ranges, this information is used to infer a suitable default mapping (e.g., Equation B.6 for floating-point parameters).

In practice, the user selects the Vol^{2}velle model and options to obtain a number of slots for which the corresponding mapping functions need to be specified. The user can then select one or several interaction parameters for each slot and, if desired, customize the mapping (e.g., by modifying the value ranges in Equation B.6). Alternatively, a different predefined function may be selected or a custom function can be entered. Some additional details on this are also described in Section C.4.

B.4.3 Sampling

During the sampling process S the function F is discretized by sampling over the parameters p'_i. The circular layout of the Vol^{2}velle introduces some constraints into the sampling process. While it would be natural to uniformly sample all parameters, this is not an ideal match for the Vol^{2}velle's geometry as it may waste large parts of the limited space on a wheel, as illustrated in Figure B.13(a). In order to make use of the free space one can generate more samples with a growing distance from the center. This leads to a non-uniform sampling with a dense sample distribution on the Vol^{2}velle wheel, shown in Figure B.13(b). If there is only one parameter to sample, dense sampling leads to a simple radial layout as in Figure B.4(b). There are also parametrizations, for example common window level controls where a conical sampling strategy, as shown in Figure B.13(c), is advantageous. As additional semantic information would be required to automatically infer an appropriate sampling pattern, the user can choose between these strategies with a default of uniform sampling. The sampling process then proceeds by

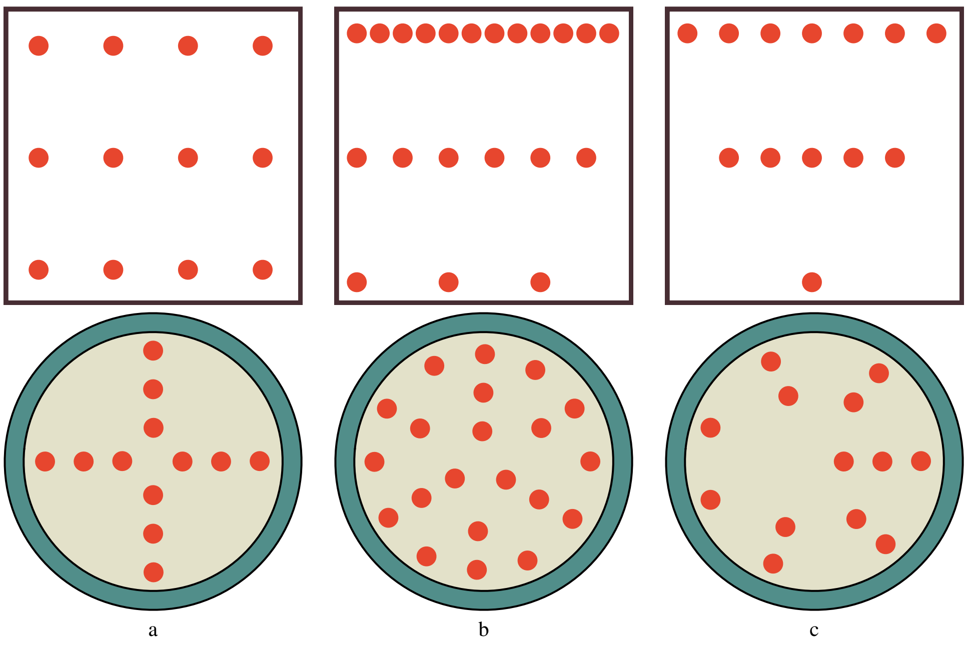


Figure B.13: The top row represents the sampling space, the bottom row represents the wheel layout. (a) A uniform distribution of the parameter, which leads to a sparse packing on the wheel. (b) Non-uniform sampling leads to a dense packing. (c) Some parameters are well suited for a conical sampling, which leads to slightly less dense packing.

generating a set of samples for each parameter p_i' of F according to the selected strategy and a chosen number of total samples. It then generates a single image for each combination of parameter values.

Particularly as a Vol^{2}velle only offers a limited capacity in terms of samples that can be placed, uniformity in parameter space may not be a good choice at all, as valuable space can be wasted with very similar images. For this reason, we also provide the option to perform perceptually uniform sampling based on the work of Lindow et al. [138]. This approach aims to equalize the distances between the output images as measured by an image difference metric.

The linearization of one parameter is performed by first creating a dense set of uniform samples which are then used to compute pairwise image differences:

$$\begin{aligned}&e(k) : K \mapsto \mathbb{R};\\&\text{where } e(k) = M(I(k-1), I(k)),\end{aligned} \tag{B.15}$$

with K being the number of samples. $I(k)$ is the visualization image with the parameter setting $p(k)$, and $M()$ is an image difference metric. Our current implementation offers the standard mean square error as well as the perceptually-based structural similarity metric (SSIM) [226]. The image differences are used to create a monotonically increasing function:

$$\begin{aligned}&h(k) : K \mapsto \mathbb{R};\\&\text{where } h(k) = h(k-1) + e(k) \text{ for } k > 0\\&\text{and } h(0) = e(0).\end{aligned} \tag{B.16}$$

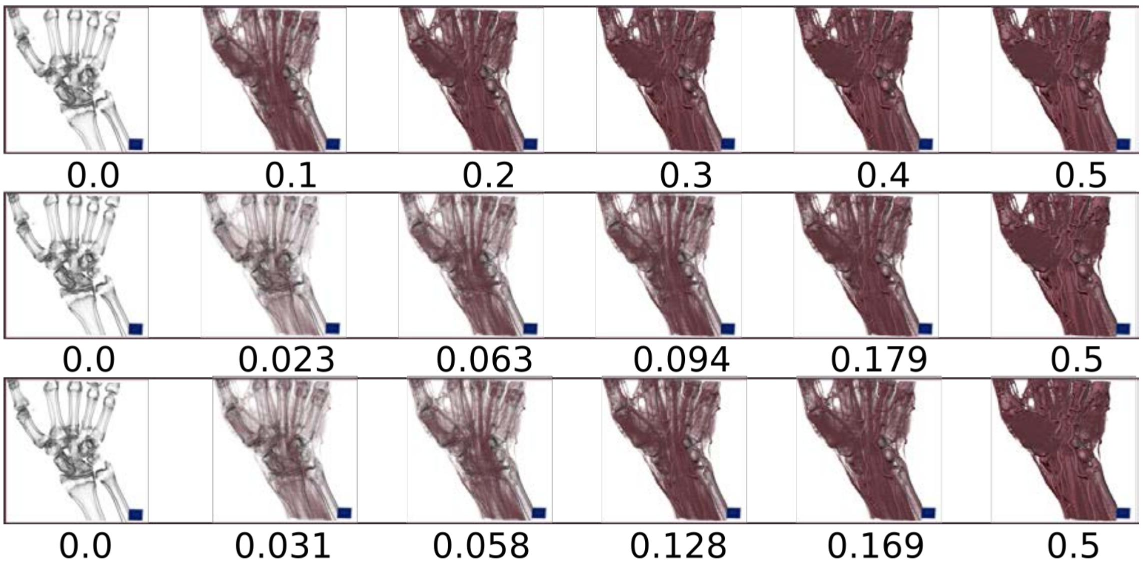


Figure B.14: **Top row:** Uniform sampling. The images look almost identical after opacity 0.2. **Middle row:** Perceptually linear parameter variation with the mean square metric. **Bottom row:** Perceptually linear parameter variation with the structural similarity metric. Both perceptually linear methods create a dense sampling for the lower transparency values. Otherwise both methods show similar results.

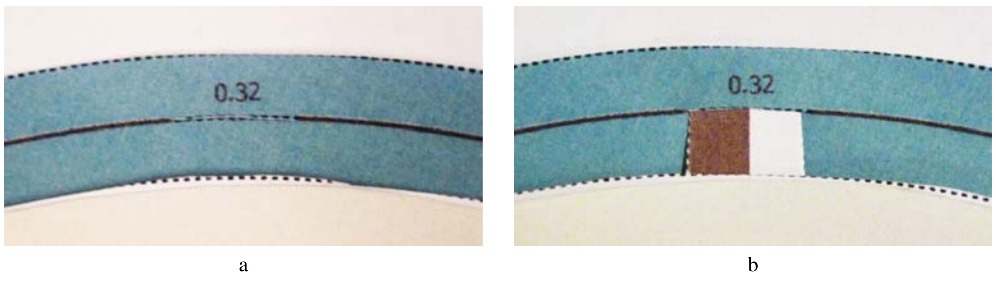


Figure B.15: (a) Values are placed radially on the base wheel. (b) In order to fix the parameter settings we add cuts under the values to fix the nobs of the rotation wheels.

The function $h(k)$ represents the perceptual change of the images for a linear parameter variation. By computing the inverse function of $h(k)$ one can find a new nonlinear parametrization which leads to perceptually linear image changes according to the given metric. The linearization of multiple parameters is similarly done by first performing a dense multidimensional sampling. By fixing one parameter dimension and computing the corresponding summed differences $h_j(k, j)$ for each fixed parameter $p(j)$, and then averaging over all the $h_j(k, j)$, we compute a nonlinear parameter variation which results in a perceptually linear variation on average. For further details we refer to the work of Lindow et al. [138].

Figure B.14 compares uniform sampling with perceptually-uniform sampling of a simple parameter (opacity) for the two different metrics, which in this case behave similarly.

B.4.4 Layout Generation

After the samples have been generated, the layout generator arranges the corresponding images on the Vol^{2}velle wheels and renders the Vol^{2}velle template as a multi-page PDF document which is then ready for printing and assembly. During layout generation, several additional graphical elements are added to each Vol^{2}velle. These include interaction guides such as tick marks and labels, elements designed to simplify the interaction, as well as additional data-dependent information. To ease the assembly, the Vol^{2}velle template uses solid lines for all graphical elements, dotted lines to indicate that cutting is required, and dashed lines where the paper should be folded.

All Vol^{2}velle models have a common basic design. Each Vol^{2}velle consists of a cover and a base which are fixed with respect to each other. The cover always contains a cutout window for the visualization, which is adjusted according to the image size. In the case of a two-dimensional sampling, the cutout window on the cover is enlarged to fit all samples. The base is mostly blank except for a ring on the edge. Labels for the parameter values are placed concentrically on the outer ring of the base wheel as shown in Figure B.15(a). In order to fix the wheels at certain parameter constellations we draw cutouts directly under the value display. These cutouts are used to hold the wheels in place as shown in Figure B.15(b). In the case of a changing picture Volvelle, the cutout geometry is placed over the two interchanging wheels (see Figure B.21). The shape of the base is either circular or a circle with an extension to a rectangular shape for the slide chart attachment. When the slide chart option is selected, the page after the last wheel consists of the linked views. We generate the slide chart as a paper strip and a trail as shown in Figure B.12. The trail has to be folded and glued onto the Vol^{2}velle base.

The common design of a Vol^{2}velle also includes a histogram of the underlying dataset on the front cover. If a transfer function is included in the parametrization, this histogram is augmented with an aligned depiction of the transfer function for the currently displayed image shown through an additional cutout. We also include the option of displaying additional text, such as a user-entered description or usage instructions, custom images, as well as meta data of the volume (if available), on the front cover.

Finally, to determine the maximum image size on the Vol^{2}velle wheels for arbitrary sampling patterns and aspect ratios, we use a force-directed layout approach.

B.5 Implementation

Our approach for Vol^{2}velle generation was implemented in C++ using the Qt toolkit as an extension to an the volume visualization framework *VolumeShop* [37] which provides several GPU-based rendering algorithms and interaction facilities. The Vol^{2}velle generation was implemented as a plugin for VolumeShop. The system includes a rich set of volume visualization methods including ranging from illustrative techniques to global illumination renderers, which are all available to the Vol^{2}velle plugin. Users can either load existing project setups, which include the specification of the desired visualization and interaction components, or create such a setup from scratch. The Vol^{2}velle module provides a simple GUI for selecting the desired model and options as discussed in Section B.4.1. The user can then interactively select any parameter that describes the system's current state to assign it to a slot of the Vol^{2}velle model. The module automatically selects an appropriate parametrization function based on the type and range of the parameter, which both can be retrieved from the environment. For more complex parametrizations, such as transfer functions, the user can simply interactively modify the current system state to define the respective ranges, similar to a keyframe-based animation approach. The parametrization function can also be modified manually and the user has the ability to define custom parametrizations using an integrated scripting language.

After the user is satisfied with the parametrization, the sampling and layout process can be initiated by pressing a button and the resulting Vol^{2}velle can be viewed and printed in an integrated PDF viewer. The whole setup can be stored and re-used for other datasets or for the generation of different Vol^{2}velles, by modifying any of the non-parametrized settings, e.g., to create a Vol^{2}velle for a different viewing angle or rendering algorithm. The system supports multiple render windows and linked views for the slide chart extensions can be specified by simply selecting the desired window. This flexible design allows our Vol^{2}velle generator to be used with any existing or future visualization module of the system.

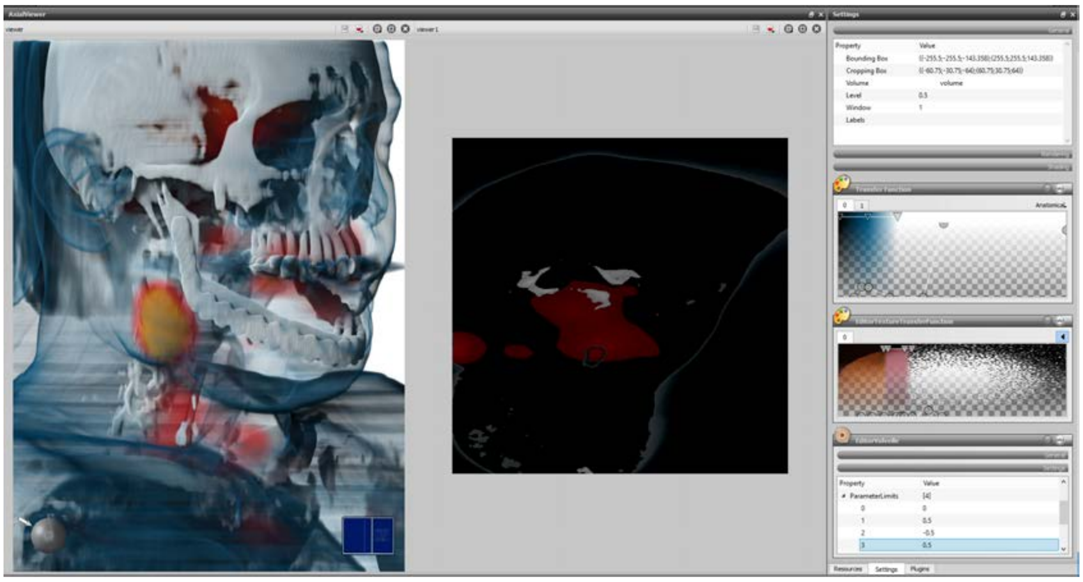
Our layout generator uses Qt's QPainter API, a powerful 2D vector graphics engine, for drawing the final Vol^{2}velle template. While currently there is only one basic Vol^{2}velle design in terms of general appearance, our implementation was performed with extensibility in mind and our software architecture allows for the straight-forward addition of alternative template designs. Additionally, certain settings such as colors or cutout positions can be exposed to the user for customization. In the future we plan to add several further templates, including ones inspired by examples of traditional Volvelles.

B.6 Results

In this section we will present several Vol^{2}velle examples to illustrate the capabilities of our approach. The corresponding Vol^{2}velle templates are also part of the supplementary material. The time for generating the Vol^{2}velle templates is primarily determined by the specific visualization setup such as the performance of the individual rendering algorithms, but for typical GPU-based volume rendering it ranges from a few seconds (without perceptual linearization) to approximately one minute (with perceptual linearization). The time needed for the assembly of a Vol^{2}velle is of course dependent on the manual skills of the user, but an untrained person is typically able to complete the build in 10 to 15 minutes. All presented examples were printed on medium heavy paper (120 g/m^2) in A4 format. We used a simple paper fastener to hold the Vol^{2}velles together.

B.6.1 Linked View Vol^{2}velle

Multimodal data depicting both morphological and functional information is common in many medical applications, such as the visualization of tumors in their anatomical context. In this example, we use multimodal volume rendering of a PET-CT data set



Property	Value
Bounding Box	([-255.5;-255.5;-143.358];[255.5;255.5;143.358])
Cropping Box	([-60.75;-30.75;-64];[60.75;30.75;64])
Volume	volume
Level	0.5
Window	1
Labels	

Property	Value
ParameterLimits	[4]
0	0
1	0.5
2	-0.5
3	0.5

Figure B.16: A screenshot of our system for the generation of Vol^{2}velles directly from interactive volume visualization setups.

of a human head with a large tumor on the right side of the chin. A linked slice view shows detailed information about the location of the tumor.

One example for the utility of such a paper-based visualization could be patient education before an upcoming surgery. The transfer function for the PET volume is fixed, and we map the transfer function of the CT scan to the cyclic parameter of the Vol^{2}velle. By changing the opacity of the CT scan the patient could gain more insight into the nature of the shown pathological structure. As such the Vol^{2}velle could be used as part of the physician's explanations for informed consent, as a type of patient-specific pamphlet that can be produced with little effort.

The Vol^{2}velle uses a slide chart extension to present the slice view, and the axial slice position is mapped to it. The slide chart can again increase the understanding of complex data. Both parameters use a simple linear mapping with uniform sampling. A screenshot of our system which depicts the interactive setup is shown in Figure B.16. The left view shows the Volume rendering and the slice view is shown on the right. The assembly of this Vol^{2}velle is illustrated in Figure B.17(a). We show photographs of the finished Vol^{2}velle in Figure B.18. On the left side, a photograph of the whole Vol^{2}velle is shown, and the right side depicts two different parameter settings for both opacity and slice position. On the lower right we show the volume visualization and the linked transfer function for the same parameter. It took us around 9 minutes to assemble this Vol^{2}velle.

B.6.2 Window Level Vol^{2}velle

A common interaction scheme for the visualization of imaging data are window level controls, where the center and width of a linear ramp function are used to adjust the brightness, contrast, and/or opacity mapping of a scalar field. In this example we use a CT scan of a hand. The two parameters, window and level, respectively, are linearly mapped to the radial and angular slots of the Vol^{2}velle using an interval of $[0.3, 0.7]$. A

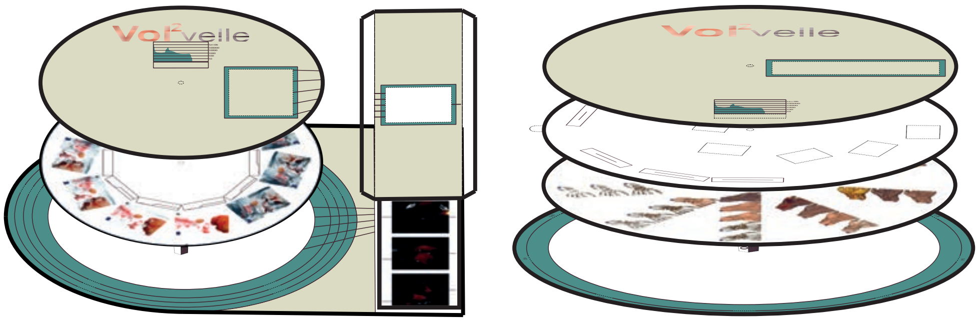
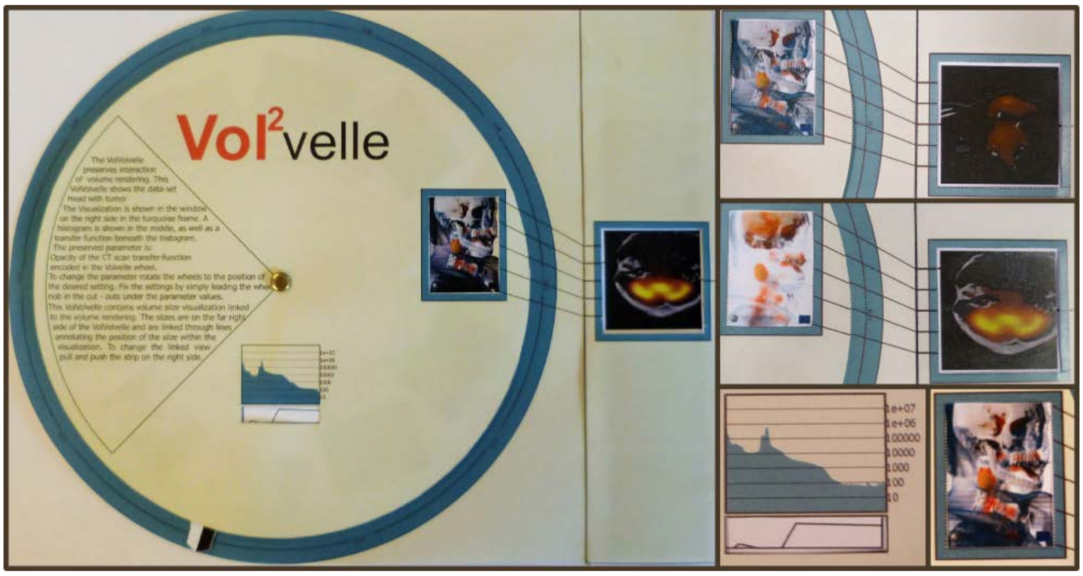


Figure B.17: (a) Linked view Vol^{2}velle with a slide chart extension. (b) Two parameter paper Vol^{2}velle with a dynamic window wheel.



The VolVolvelle preserves interaction of volume rendering. This VolVolvelle shows the data-set Head with tumor. The Visualization is shown in the window on the right side in the turquoise frame. A histogram is shown in the middle, as well as a transfer function beneath the histogram. The preserved parameter is: Opacity of the CT scan transfer-function encoded in the Volvelle wheel. To change the parameter rotate the wheels to the position of the desired setting. Fix the settings by simply leading the wheel nob in the cut - outs under the parameter values. This VolVolvelle contains volume slize visualization linked to the volume rendering. The slizes are on the far right side of the VolVolvelle and are linked through lines annotating the position of the slize within the visualization. To change the linked view pull and push the strip on the right side.

Figure B.18: Finished Vol^{2}velle with linked views. The slice position is indicated through lines. Rotation of the wheel changes the opacity and updates the linked transfer function as shown on the lower right.

standard GPU-based volume renderer with an emission-absorption optical model and gradient-based shading is used.

Due to the potential non-linearity of the differences between the resulting images, this is a good example for the utility of perceptually uniform sampling. We show the resulting information wheel in Figure B.19(b). In Figure B.19(a) we show the same mapping without perceptual linearization, which shows rapid transitions in the visibility of the skin. The assembly of the Vol^{2}velle is illustrated Figure B.17(b). In order to be able to interact with two parameters individually, a freely rotating wheel chart with windows is placed right after the front cover.

We show a photograph of the finished Vol^{2}velle in Figure B.20, with the complete Vol^{2}velle on the left side and detailed views of the visualization images with different settings on the right side. This Vol^{2}velle contains 25 samples in total which leads to relatively small images, hence this layout is less suitable for setups with high-frequency changes in the images. The assembly of this Vol^{2}velle took about 8 minutes.

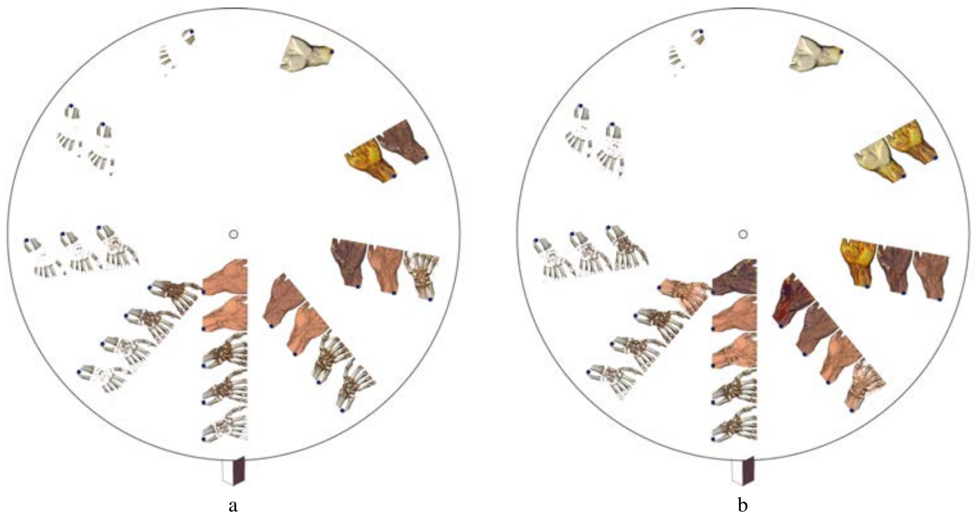


Figure B.19: (a): Linear sampling. (b) Perceptually linear sampling shows the change of the skin more prominently.

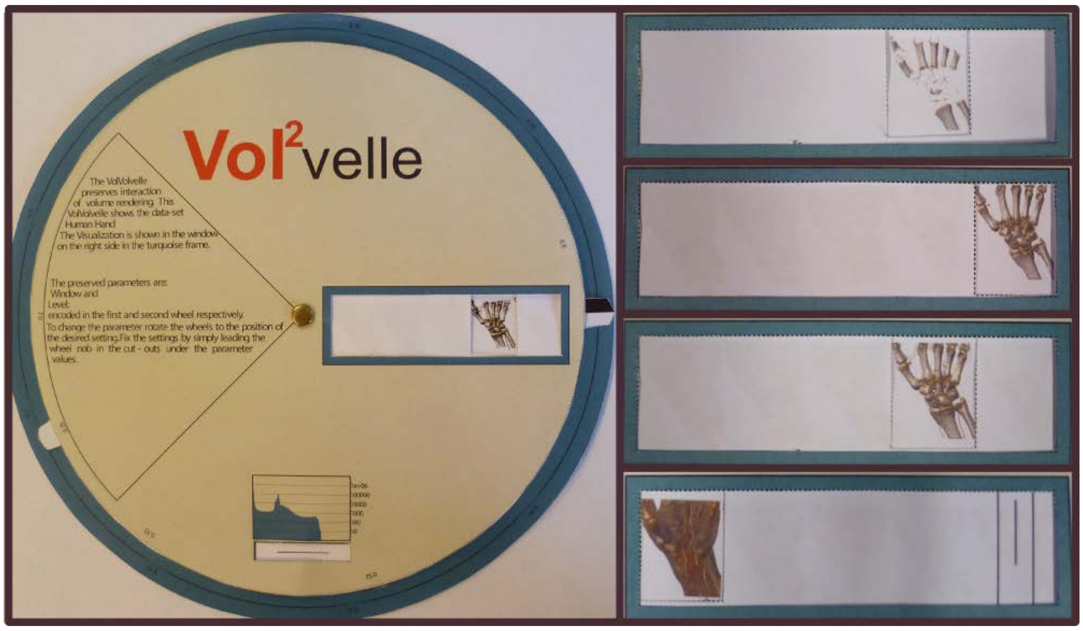
B.6.3 Light Position Vol^{2}velle

A very natural choice for the cyclic parameter is the position of a light source. This example uses a microCT scan of a new millipede species [153] and represents another interesting scenario for the use of Vol^{2}velles. One could imagine the presentation of such a data set as a virtual installation in a museum or exhibition. The visitors would have several interaction options such as light position, a cutting plane to sweep through the volume, rotation of the specimen, and maybe some interaction with the transfer function. As a souvenir, the visitor has the option to create a small version of this interaction. To create this particular Volvelle the user selects the light position as a preserved interaction and defines two positions of the cutting plane to create a changing picture Vol^{2}velle as shown in Figure B.22(a). We show photographs of the Vol^{2}velle window in Figure B.23. Some visualization images must be cut in order to employ the changing picture mechanism and the cutting lines are placed directly over the images (see Figure B.21), however those cuts are only noticeable under strong light at relatively low angle to the paper.

The changing picture Vol^{2}velle is the most complex in terms of assembly—the presented example took around 17 minutes to build. However, the additional parameter on the wheel makes this Vol^{2}velle particularly enjoyable as the mechanics present an element of surprise.

B.6.4 Transparent Vol^{2}velle

Sometimes visualization is not focused on providing an accurate representation but rather on conveying the idea of the data. This is often the case for illustrative visualization. In this example we create an expressive Vol^{2}velle for an illustrative visualization of a human head using style transfer functions [39]. We use a segmented CT scan of a human head. The segmented parts (head, mandible, eyes, brain and blood vessels) have a fixed style with opacity as an interaction parameter. We group the mandible and the eyes as well as the brain and the blood vessels and link the Vol^{2}velle parameters to



The VolVolvelle preserves interaction of volume rendering. This VolVolvelle shows the data-set Human Hand
The Visualization is shown in the window on the right side in the turquoise frame.

The preserved parameters are:
Window and
Level:
encoded in the first and second wheel respectively.
To change the parameter rotate the wheels to the position of the desired setting.Fix the settings by simply leading the wheel nob in the cut - outs under the parameter values.

Figure B.20: A Vol^{2}velle with two parameters on one wheel, with window as concentric and level as radial parameter. The wheel contains 25 images in total which limits the size of the images.

their respective opacity settings. Each of the groups is printed on a separate transparent wheel. Through the arrangement of the wheels in the order of their closest distance to the viewpoint, the composite volume rendering can be approximated. The effects of this approximation can be seen in Figure B.24. The assembly of the Vol^{2}velle is illustrated in Figure B.22(b). Such a Vol^{2}velle can be seen as a direct approximation of volume rendering. The user is able to change the transparency of the segmented parts resulting in an intuitive Vol^{2}velle with a pleasing appearance through the blending of individual layers.

Figure B.25 shows a a photograph of the finished Vol^{2}velle. On the left we show the whole Vol^{2}velle and on the right we show detailed views for different parameter constellations. The visualization images on the Vol^{2}velle are more transparent compared to

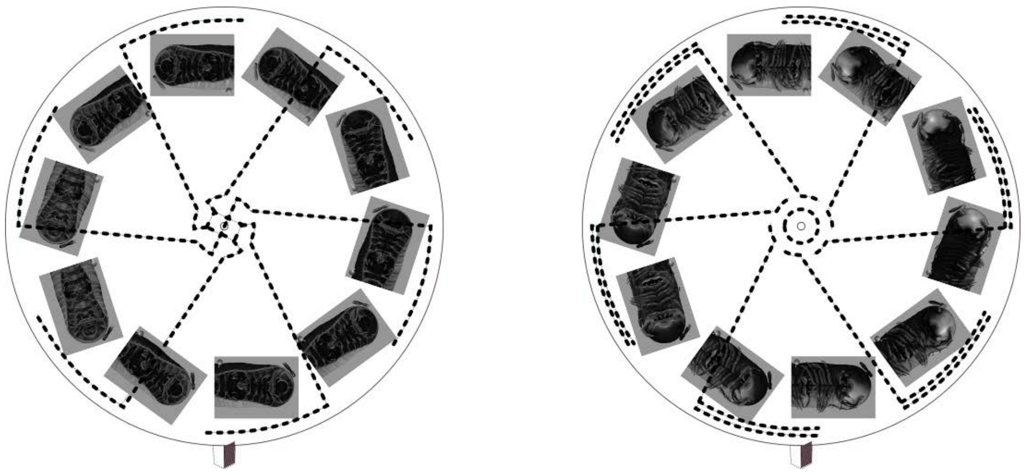


Figure B.21: The cutting lines intersect the images on the interleaved wheels.

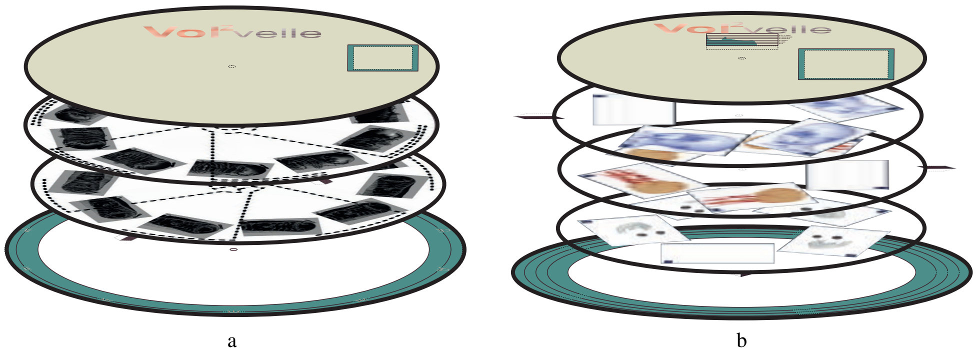
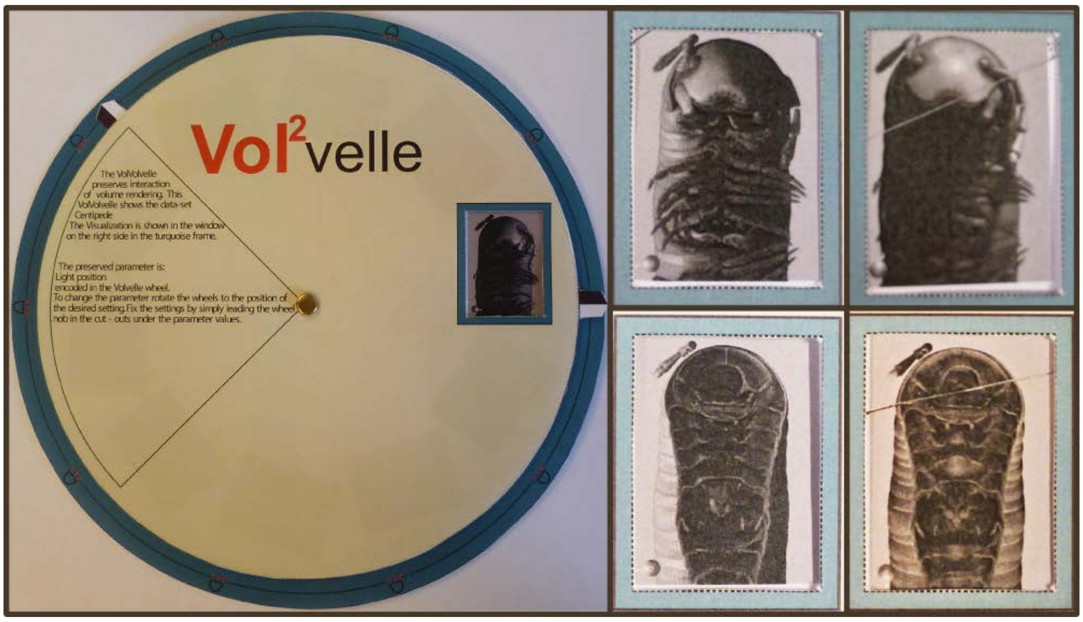


Figure B.22: (a) The assembly of a changing picture Vol^{2}velle. The light position can be changed by rotation, and a cutting plane can be turned on or off using the iris mechanism. (b) The assembly of transparent medium Vol^{2}velle.



The VolVolvelle
preserves interaction
of volume rendering. This
VolVolvelle shows the data-set
Centipede
The Visualization is shown in the window
on the right side in the turquoise frame.

The preserved parameter is:
Light position
encoded in the Volvelle wheel.
To change the parameter rotate the wheels to the position of
the desired setting.Fix the settings by simply leading the wheel
nob in the cut - outs under the parameter values.

Figure B.23: The finished changing picture Vol^{2}velle. Because of the interleaved wheels, some images have to be cut.

the computer generated visualization, but they still manage to convey the relationship between the individual structures well.

B.7 Discussion

Physical media naturally have considerable limitations in their capacity for representing complex interaction. The limitation of data storage on printed media is evident, information can only be stored on the visible parts of the printable medium. This leads to a trade off between the image size and the number of samples that can fit on a wheel. In our examples the number of images per wheel ranged from 5 to 25, Vol^{2}velles with 41 images are possible with a dense sampling. In our experience more samples are typically not practical on an A4 sheet.

In addition to space limitations, the dimensionality of the parameter space is also

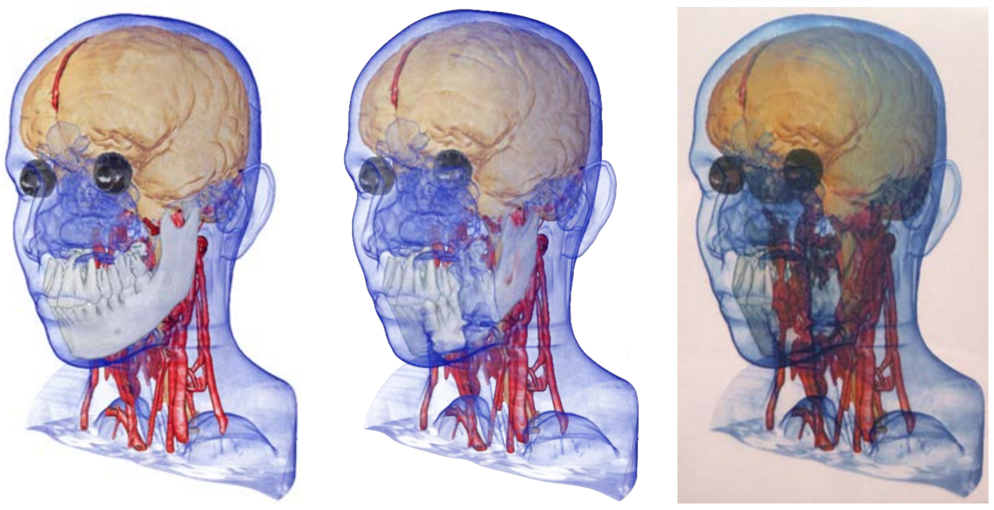


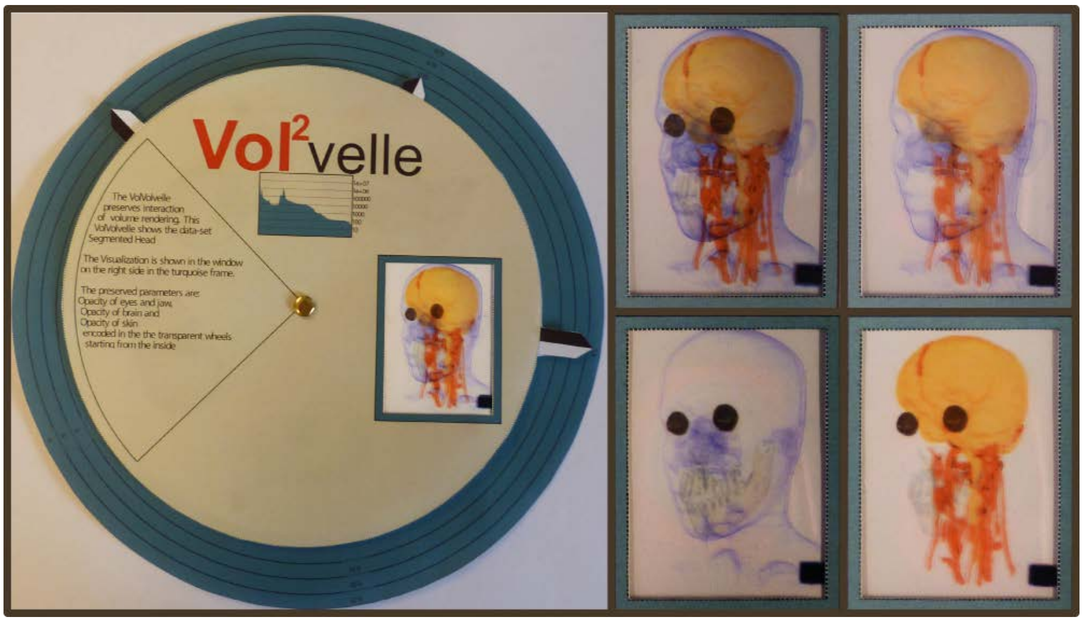
Figure B.24: (a) Direct volume rendering with style transfer functions. (b) Image-based blending of the individual layers. (c) Physical blending with transparent films. Because most printers are not capable of printing white color, the white jaw is more transparent in the physical blending.

heavily constrained. While stacking of wheels can be used to partially overcome this, the additional complexity quickly outweighs the gain. Moreover, the full potential of stacked wheels can only be exploited in an unconstrained design space, where the information representation is not bound to a spatial context, for instance in the case of textural information. Text snippets can be distributed over the whole Volvelle and multiple strategically placed cutouts can be used to integrate them as shown in the examples in Figure B.1.

The Vol^{2}velle should be printed on white paper as most printers are not able to print white color. Many printers are not able to reproduce a rich set of gray colors, leading to rather dark images as in Figure B.23. As we take a direct snapshot of the visualization for the Vol^{2}velle generation, the background of the visualization is preserved. Naturally the quality of the visualization image is dependent on the used printer, however we did not experience drastic differences between Vol^{2}velle printouts for different printers.

We allow the use of transparent media for the Vol^{2}velle generation. While transparent media potentially increase the number of possible parameters, they carry some disadvantages as well. As mentioned above most printers are not capable of printing in white color. Furthermore, prints on transparent media are fully opaque for black color only. This leads to discrepancies between the virtual and the physical visualization as depicted in Figure B.24. However, there are professional printers that do not suffer from these limitations. Another issue in the context of transparent media is that our current solution requires a uniform depth ordering for the individual layers. In the future, we plan to introduce a more automatic solution for this, but the physical restriction in the number of layers means that this will still only be an approximation in most cases unless a "natural" layering exists in the underlying data.

To summarize, Vol^{2}velles can by no means be considered to be a serious competitor for tablets, augmented reality, or other advanced solutions to allow tangible interaction with visualizations and this was never our aim. However, we have shown that there are



The VolVolvelle preserves interaction of volume rendering. This VolVolvelle shows the data-set Segmented Head

The Visualization is shown in the window on the right side in the turquoise frame.

The preserved parameters are:
Opacity of eyes and jaw,
Opacity of brain and
Opacity of skin
encoded in the the transparent wheels starting from the inside

Figure B.25: Finished Vol^{2}velle with transparent medium.

viable use cases where the low cost, simplicity, and recyclable character of a Vol^{2}velle present an interesting addition to other means of presentation. While we have focused on volumetric data and wheel charts in this paper, we also believe that other types of visualization and infographics offer interesting opportunities for further exploration of interactive paper-based interfaces.

B.8 Conclusion

In this paper we presented a method for the generation of Vol^{2}velles, paper-based visualizations of volumetric data with interaction capabilities inspired by traditional wheel charts. Based on an analysis of the wheel chart design space, we identified a basic set of Vol^{2}velle models which are well-suited for the presentation of volumetric data. We discussed an approach for the mapping of interactive visualizations to the limited degrees of freedom of the Vol^{2}velle and presented a flexible interactive system which enables the easy specification of this mapping. Our results show that our approach can be used to create tangible, recyclable, and interactive visualizations of volumetric data and we outlined several potential use cases. Furthermore, the concept of interactive paper-based interfaces may also stimulate future research in other areas of visualization.

Paper C

Smart Surrogate Widgets for Direct Volume Manipulation

Sergej Stoppel and Stefan Bruckner

University of Bergen, Norway

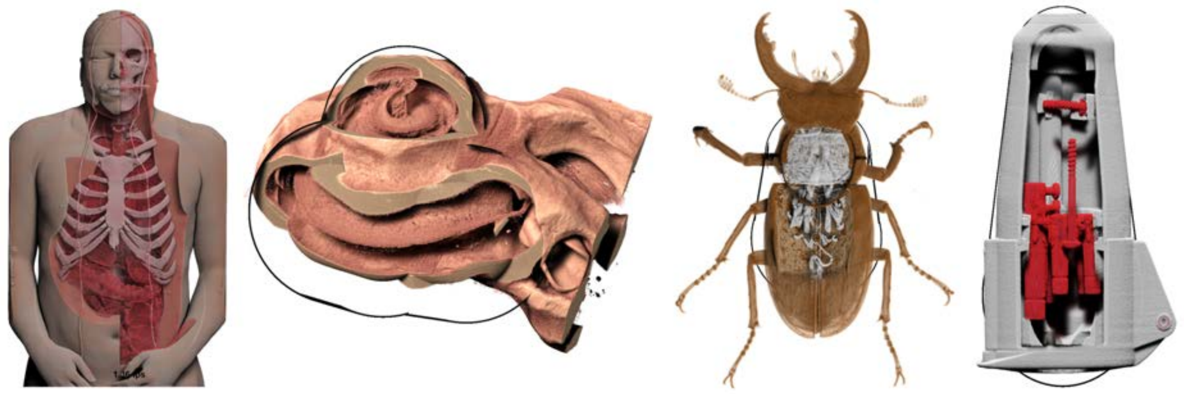


Figure C.1: Illustrative visualization techniques such as cutaways or ghosting views are used to emphasize important concealed structures. Typically such structures are carefully segmented prior to the visualization. Our approach allows for the simple creation of illustrative visualizations without prior processing of the data. The leftmost image shows the visible human data set manipulated with our technique to reveal the skeleton and inner organs. The next image shows an illustrative visualization of the human cochlea. The second image from the right shows an illustrative cutaway of a beetle. The rightmost image shows a cutaway illustration of a high voltage power outlet.

Abstract

I nteraction is an essential aspect in volume visualization, yet common manipulation tools such as bounding boxes or clipping plane widgets provide rather crude tools as they neglect the complex structure of the underlying data. In this paper, we introduce

This article was published in Proceedings of 2018 IEEE Pacific Visualization Symposium (PacificVis) (2016). pages 36-45 a novel volume interaction approach based on smart widgets that are automatically placed directly into the data in a visibility-driven manner. By adapting to what the user actually sees, they act as proxies that allow for goal-oriented modifications while still providing an intuitive set of simple operations that is easy to control. In particular, our method is well-suited for direct manipulation scenarios such as touch screens, where traditional user interface elements commonly exhibit limited utility. To evaluate out approach we conducted a qualitative user study with nine participants with various backgrounds.

C.1 Introduction

Volume visualization is a well established method for the exploration, investigation and representation of 3D scalar fields, with various applications in medicine, engineering, physics, architecture and biology. In the recent years volume visualization techniques have been increasingly targeted at non-expert scenarios, for instance in the form of museum installations, and novel interfaces such as touch-based tabletop displays are becoming more common [243]. However, interacting with volume visualizations still remains a complex task requiring training and expertise, and new developments in terms of intuitive interaction design are needed.

Interaction interfaces for volume visualizations primarily come in two forms: as spatially-detached widgets allowing for complex and detailed manipulations, e.g., for the specification of transfer functions, and as widgets with direct spatial semantics, that tend to be very general (e.g., bounding boxes for the specification of clipping operations). Since direct manipulation was proposed by Shneiderman [190] in 1982, it had a major influence on interaction design. While direct interaction has its weakness in managing abstract or attribute-rich objects, the directness particularly benefits the novice user in terms of intuitiveness and seamless interaction results.

For these reasons, we present a direct volume manipulation approach targeted at non-expert users inspired by the notion of surrogate objects as discussed by Kwon et al. [127]. We propose visibility-driven smart surrogate widgets that are automatically constructed by fitting a model onto the visible structures of the current view. We allow for several interactions on the surrogate objects which are directly translated to operations on the current visualization. More specifically, we follow four main principles of Shneiderman [190]:

- Continuous representation of the object of interest
- Physical actions instead of complex syntax
- Rapid, incremental, reversible operations whose impact on the object is immediately visible
- Layered or spiral approach to learning that permits usage with minimal knowledge

Our contributions in this paper are the following:

- A novel approach for visibility driven surrogate widget generation

- Incremental, flexible and invertible construction of widgets, integrated directly into the volume visualization
- Seamless translation of simple gestures into volume manipulations
- Intuitive widget design that allows for complex illustrative manipulation through simple, incremental interactions without prior volume segmentation

C.2 Related Work

Research inspired by art and illustration is a well established subfield of visualization and computer graphics. Illustrative visualization, for example, aims to reproduce and extend traditional illustration techniques in computer-based visualization methods. Much research has been devoted to developing methods for visualizing occluded structures in their spatial context [219]. Hauser et al. [86] introduced two-level volume rendering, an approach that allows for the flexible combination of different rendering techniques for subsets of the data. Bruckner et al., for instance, introduced several illustrative techniques for volume visualizations such as cutaways [36] and exploded views [38]. Viola et al. [220] proposed importance-driven feature enhancement in volume visualization in order to steer the application of abstraction methods. Based on these ideas, Viola et al. [218] presented a system that determines the most expressive view for a user-selected feature. Li et al. [135] proposed a novel cutaway design based on the occluder geometry, where previous approaches focused only on the occluded object. Lawonn et al. [129] presented an approach for the occlusion-free visualization of blood flow and wall thickness in vascular data. Elmqvist et al. [63] discussed a taxonomy of the design space for occlusion management techniques. Birkeland et al. [33] presented a view-dependent technique for the generation of peel-away views. These approaches focus on the visualization of occluded structures. While our approach also can be used to generate cutaways or ghosted views, our primary aim is to support the specification and manipulation of specific features of interest by the user instead of requiring a pre-processing step such as volume segmentation to identify relevant structures.

The benefits of direct manipulation in contrast to command languages were already discussed by Shneiderman [190] in 1983. For more than three decades, the direct manipulation paradigm has influenced interface design, encouraging designers to minimize indirection and to transform domain objects directly into intuitive widget interfaces. A common design pattern for data exploration is the concept of magic lenses. Magic lenses reveal secondary structures through dynamic lens widgets. For a detailed survey on interactive lenses for visualization we refer to Tomisnki et al. [213]. The design of our smart surrogate widgets incorporates the concept of magic lenses by providing focus regions for which alternate optical properties can be specified.

Kniss et al. [123] described a set of direct manipulation widgets for intuitive transfer function manipulation, focusing on probing operations to ease the establishment of correspondences between data attributes and visible features. Guo et al. [80] developed a sketch-based approach for volume rendering manipulation, allowing for quick and easy adjustment of properties such as such as color, transparency, contrast, or brightness. Gerl et al. [76] proposed an interactive approach for the explicit specification of semantics in volume visualization, to visually assign meaning to both input and output parameters of the visualization mapping.

Weiskopf et al. [228] introduced techniques for depth-based clipping in texture-based volume rendering, allowing for the efficient use of complex clipping geometries. Our work instead focuses on the easy creation and modification of clipping geometries, exploiting the information already present in the current visualization state. For this purpose, as proposed in the work of Kwon et al. [127], we use the concept of surrogate interactions that enable the user to interact through easily-understandable proxies instead of more complex domain objects. Yu et al. [245] presented FI3D, a direct-touch data exploration technique for 3D visualization. While they primarily targeted global operations such as rotation, zooming, and translation, their approach also incorporated simple cutting and selection tools.

Instead of considering the underlying data in its entirety, our approach is deliberately based on visibility and hence our widgets are constructed according to the current visualization state, i.e., the structures that are currently visible and identifiable, allowing users to manipulate the objects that they see. This is in contrast to data-driven approaches, that attempt to extract additional semantics irrespective of the current state. Gagvani et al. [74], for instance, presented a technique to animate volumes using a volumetric skeleton tree. McGuffin et al. [147] explored an alternate strategy for volumetric cutting and peeling tools. By using predefined semantic information, the user can apply deformations to selected layers using pop-up menus and integrated 3D widgets. Birkeland et al. [32] developed an illustrative clipping technique with automated feature preservation using an elastic membrane that adjusts itself to salient structures. Le Muzic et al. [152] presented Visibility Equalizer, visualization approach for mesoscopic biological models that applies cutting planes only to certain parts of the data.

While data-driven methods can be very useful for global manipulations, e.g., in the context of transfer function design [221], more localized operations greatly benefit from the incorporation of visibility and other information. Selecting three-dimensional shapes through a two-dimensional interface can be challenging, and structure-aware or context-aware interaction techniques can be employed to overcome some of the inherent ambiguities. These techniques try to deduce the user intent based on the underlying data and the current state of the provided visualization. Owada et al. [162] solved the challenge of three-dimensional selection by asking the user to draw the perceived outlines of the displayed volume and deducing well-suited segmentations from the sketch. This approach was later extended by preprocessing the 2D domain before applying a 2D-to-3D stroke elevation algorithm [163]. Ropinski et al. [177] proposed an approach for stroke-based transfer function design, where the transfer function is computed to emphasize user-specified features of interest selected by simple sketches.

The work of Wiebel et al. [231] proposed methods to select 3D positions for 2D picking operations based on visibility information. They later extended this approach to map 2D strokes to the most visible features in a 3D volume [230]. Based on this idea, Stoppel et al. [199] extended the method to select visible surface patches in direct volume rendering. Yu et al. [244] presented a family of gesture-driven Context-Aware Selection Techniques (CAST) for the interaction and analysis of large 3D particle clouds. In addition to using contextual information to infer the likely meaning of user interactions, our approach aims to incorporate this information already in the construction of interaction widgets, offering additional guidance to novice users.

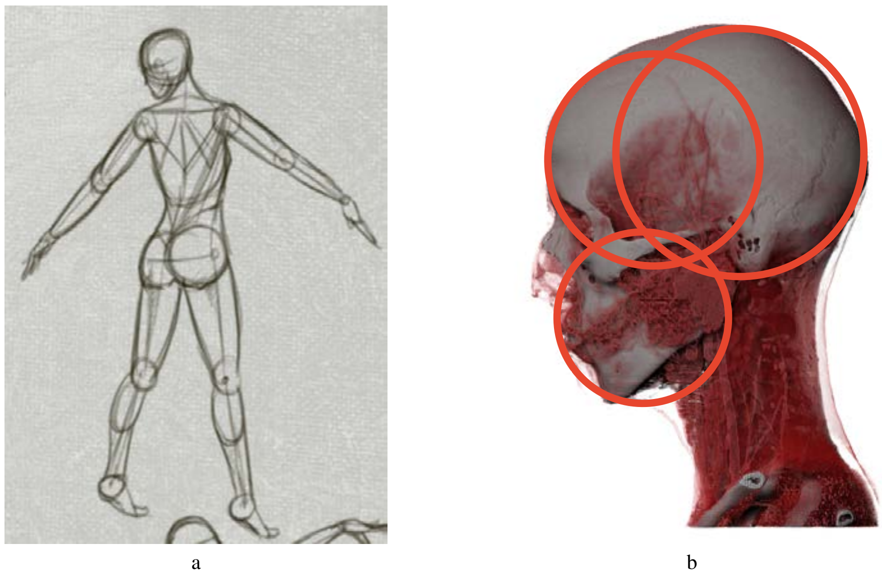


Figure C.2: (a) Illustrators often approximate complex shapes with a set of circles and the connections between those (Illustration by Paulina Kawenka). (b) A set of only three spheres already gives a good approximation of the shape of a human head.

C.3 Smart Surrogate Widgets

A central aspect of our approach is to provide users with easily understandable yet sufficiently powerful and flexible interaction widgets based on the content of the visualization as seen on screen. When confronted with visualizations of volumetric data, novice users frequently seek for ways to "open up" visible structures in order to look inside, but common interaction widgets such as bounding boxes, due to their uniform structure, can make it difficult to specify a desired operation. In particular, the initial placement of such tools can be challenging, as 3D manipulation is often difficult for novice users. For such users, ideally, a system should "know" the spatial structure of visible features and should provide them with an "expected" set of operations.

While it is of course difficult to capture such subjective notions, we can look at existing cutaway and breakaway illustrations for inspiration. Such illustrations frequently use simple and symmetric shapes to partially remove occluding structures, allowing the viewer to mentally reconstruct the entire object. Furthermore, when sketching an object, artists and illustrators often approximate complex structures with a set of simple shapes such as spheres, as shown in Figure C.2(a). Motivated by this, we propose surrogate widgets as constructions of primitive shapes such as spheres and tapered cylinders. In fact, medical and biological scans can often be approximated with a relatively small number of spheres as illustrated in Figure C.2(b).

A general overview of our approach, which we detail in the remainder of this section, is shown in Figure C.3. First, we discuss the automatic construction of our widgets in Section C.3.1 and then proceed to discuss the available interaction mechanisms in Section C.3.2.

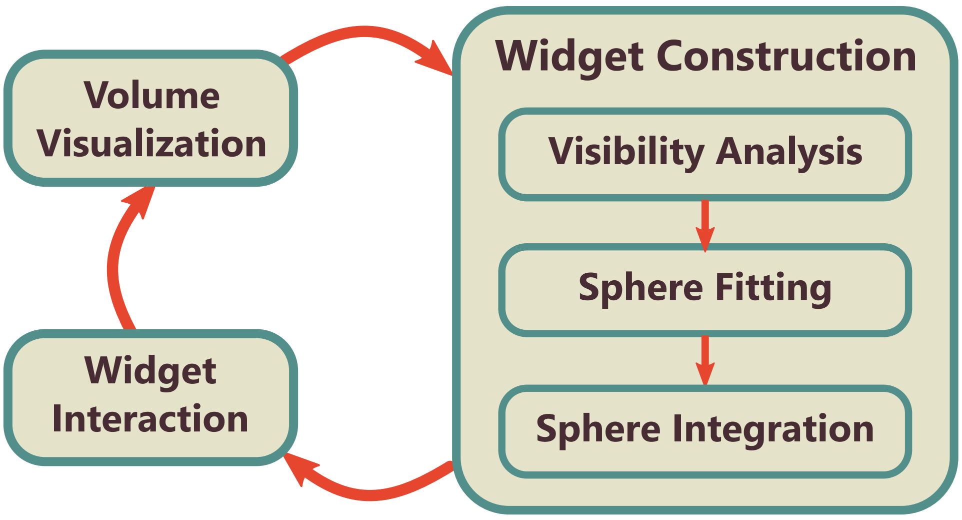


Figure C.3: Overview of our approach for smart surrogate widgets and its integration into the volume visualization pipeline. To fit the surrogate widget into the rendered image, we analyze the visibility and compute a set of prominent points for the given visualization state. In the next step, we fit a sphere into the computed point set. The new sphere is combined with the existing widget. The updated widget can be used for further volume manipulation.

C.3.1 Widget Construction

The main idea for the construction of our widgets is the aim of representing the most prominent visible structures using a set of simple primitive shapes that can be easily interpreted and manipulated. Topologically, our widgets are undirected acyclic graphs, where the nodes correspond to spatially anchored spheres with varying radii and the edges correspond to tapered cylinders that connect them. The widgets are constructed incrementally whenever triggered by the user. When requested, our system initiates the construction of a new widget, initially consisting of a single sphere. If more complex operations are desired, additional widget components can be added, which will be connected to the existing spheres via a tapered cylinders based on spatial proximity. Likewise, the user can remove spheres when they are no longer needed or discard the entire widget.

The creation and extension of the widgets requires a reliable method for fitting a sphere into the rendered data. To achieve this, we employ a random sample consensus (RANSAC) model [66]. RANSAC is an iterative learning algorithm that estimates parameters of a mathematical model by randomly sampling the observed data designed to operate on point cloud data. The algorithm works by iteratively identifying outliers in the data and estimating the model with data not containing the outliers. As such RANSAC is tolerant to noise in the data and is commonly applied to similar scenarios such as object recognition.

As mentioned, the RANSAC algorithm requires a point cloud of object surface points, for estimating the model. Therefore, we have to compute surface points for the visible structures in the rendered image. Inspired by the point picking approach proposed by Wiebel et al. [231], we estimate the surface points of visible structures as the most prominent points, i.e. points with the highest opacity gain inside each structure. As illustrated in Figure C.4, we evaluate the accumulated opacity α for each

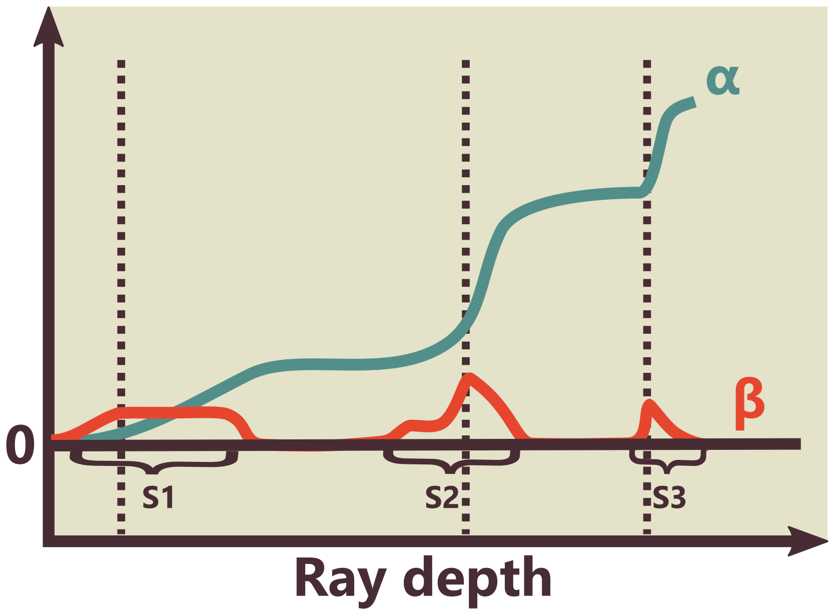


Figure C.4: Borders of structures S_k are detected by local maxima of β. This defines a point cloud for the whole scene which is used for widget fitting.

point p along the ray. If the point is already inside a widget, its opacity is considered to be zero. From the function α, we compute its derivative function β. Note that the derivative β is positive throughout the ray profile, being zero at plateaus of α. We can then divide β into strictly positive segments, where each segment represents a visible structure along the ray. Following Wiebel et al., we identify the surface points of the visible structures as the most prominent points $p_{k,max}$ of each structure along the ray (indicated by dotted lines in Figure C.4). For each structure S_k, the most prominent point $p_{k,max}$ is then defined as:

$$\beta(p_{k,max}) \geq \beta(p) \qquad \forall\, p \in S_k \tag{C.1}$$

$$d(p_{k,max}) \leq d(p) \qquad \forall\, p \text{ s.t. } \beta(p) = \beta(p_{k,max}). \tag{C.2}$$

where $d(p)$ is the distance of point p to the camera. Thus, we define the surface point of each structure as the nearest point with the highest opacity gain inside each structure. Because the RANSAC algorithm considers all points to be equal it would not distinguish between very transparent or very opaque structures when fitting the model. To address this we simply save the surface points of the most prominent structures along each ray by computing the visibility contribution of each structure. This approach provides us with a coherent estimate for the visually most prominent point for each pixel, resulting in a point cloud covering the whole rendered volume that can be used to find the best fitting sphere into the visible data.

After finding the best matching sphere based on the visibility information, it is either integrated into the existing widget or is declared to be the initial widget, if no widget exists so far. To integrate the new sphere into an existing widget, a tapered cylinder is constructed between the new sphere and the nearest sphere in the existing widget. If a new sphere is added to a widget with more than three spheres, it may be connected with a sphere with more than one existing connection. In this case the widget will branch out, as illustrated in Figure C.6.

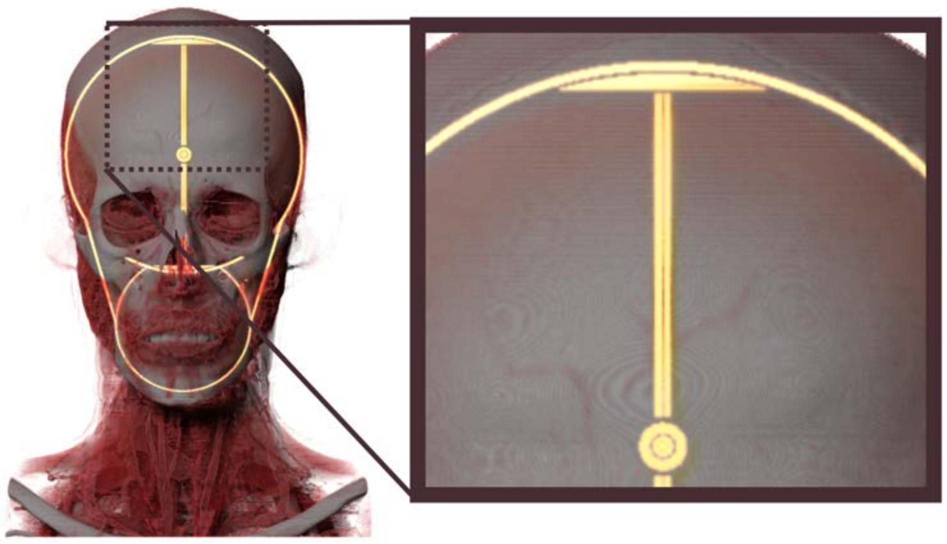


Figure C.5: Smart surrogate widget rendered into the visualized volume. The widget is emphasized through a glow in this image.

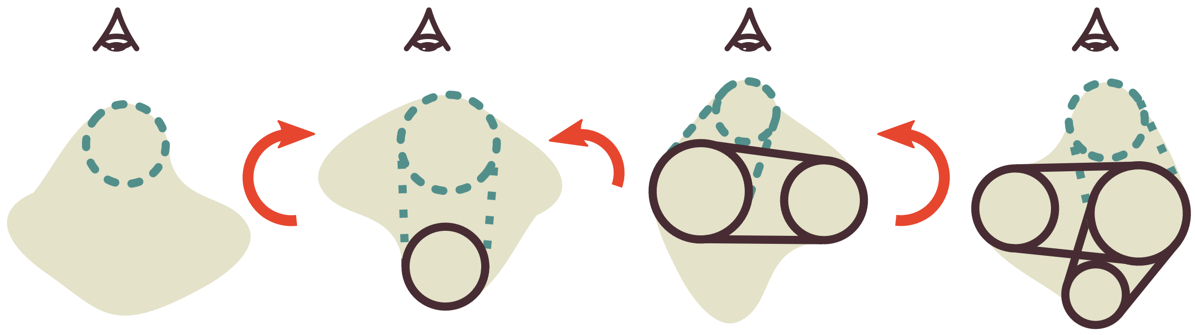


Figure C.6: When a new sphere is added to the widget, it is connected to the closest existing sphere. As such the widgets can branch, as shown in the rightmost image.

The radius of fitted spheres is limited to an interval of $[s_{\text{min}}, s_{\text{max}}]d$, where d is the smallest extent of the volume's bounding box. The experimentally determined default values in our implementation are $s_{min} = 0.03$ and $s_{max} = 0.4$ (preventing the matching of overly large or very small objects), which were used for all results in the paper and the video, but the user is able to adjust these values freely.

C.3.2 Interaction Design

Our approach is designed to provide instant and context-dependent means for performing localized manipulations of volume data, including operations such as cutting or highlighting. By adjusting viewing parameters like rotation and zoom, users indicate what they are interested in, and our approach for widget construction automatically provides manipulation facilities based on the visible structures. We deliberately selected a set of simple and easy-to-interpret operations, which still provide considerable flexibility.

Graphical editor software such as Photoshop typically divide manipulation tools into two components. The first component allows direct interaction via surrogate ob-

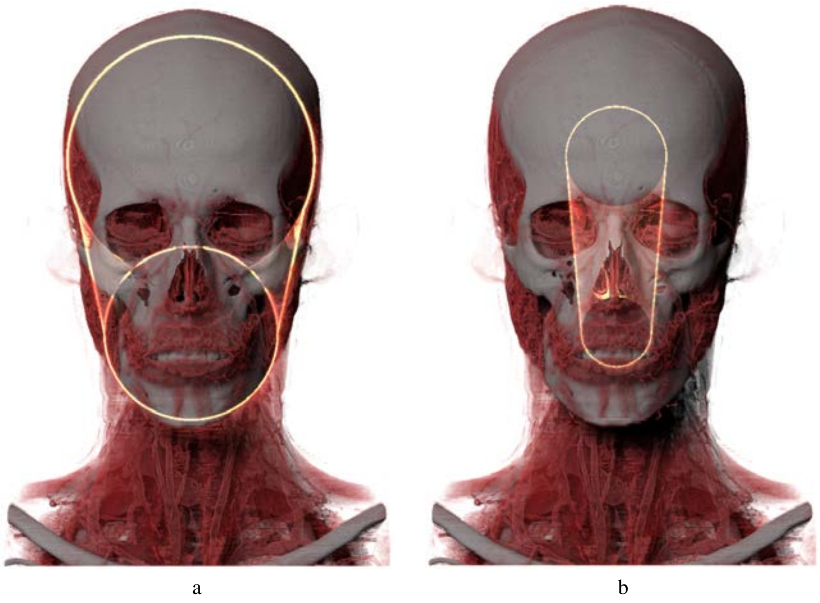


Figure C.7: The widget size can be changed by pulling on the outer frame. The selected widget part is emphasized with a glow.

jects that are embedded into the image, while the second component represents control panels or dialogs that do not have a direct spatial interpretation, such as opacity manipulation or contrast adjustment. The surrogate widgets in graphical editor software typically consist of a frame around the focus object with handles for drag interactions, such as rotation or deformation. Additional properties are controlled via user interface elements such as sliders or text fields which are typically arranged around the image. Our smart surrogate widgets are designed to conform to this common model and consist of a spatial representation shown as part of the rendered image (space widget) and a floating panel displayed in the upper part of the viewport (detached widget).

The region covered by the space widget is subdivided into the focus region and the background. Initially, the focus region is empty, i.e., the widget does not affect the visualization. By clicking on a sphere of the widget, it will be activated and can be resized by dragging its contours, as demonstrated in Figure C.7. The focus region can be manipulated by two simple tools, an iris tool and a wedge tool. Changes of the focus region are propagated along all cylinders attached to it, and the final focus region is formed by the union of these subregions. The user can rotate the iris by moving a small circle on a sphere (see Figure C.8). To change the size of the iris, users can click and drag a ring around the position circle. The focus is defined by the region enclosed in the cone defined by the iris ring and the sphere center as apex, as shown in Figure C.8. The wedge tool can be controlled by moving two independent arcs, and the focus region is set to the volume between the two arcs, as shown in Figure C.9(a). The two arcs can be rotated around the view vector by pulling on their poles (see Figure C.9(b)).

The main purpose of the detached widget is to control the appearance of the focus region and to provide access to additional global operations. First, we provide a slider to limit the extent of the focus region, essentially generating an inner boundary. Modifying this slider inflates a volume from the center of the widget that is not part of the focus region. By switching between two radio buttons, the inner volume can either be

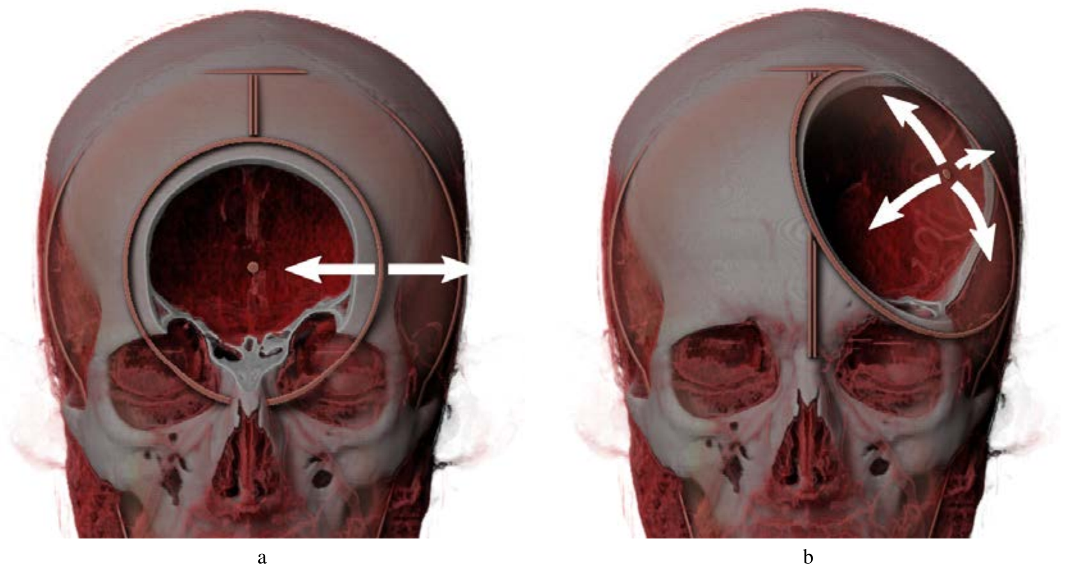


Figure C.8: (a) The iris size can be changed by pulling on the circular iris opening. (b) The iris can be rotated by moving a handle representing the center of the iris opening.

displayed with the same transfer function as the rest of the data (see Figure C.10(a)) or with the alternate transfer function (see Figure C.10(b)). Furthermore, we provide a slider that can be used to control an opacity multiplier for the focus region. A multiplier with the value 0 renders the focus completely transparent, resulting in a cutting operation, while a value of 1 results in an unmodified rendering with the selected transfer function. In addition to modulating the opacity, the user can also switch to display the focus region with an alternate transfer function. This is illustrated in Figure C.11, where a widget consisting of two spheres covers a CT scan of an armadillo. By using a combination of the iris and wedge tools, the body is opened and the opacity inside the focus region is reduced. In Figure C.11(a), the focus area is rendered with the initial transfer function, whereas in Figure C.11(b) the focus area is rendered using the alternate transfer function. In both cases a smaller volume inside the smart surrogate widget is unaffected by the opacity changes, which allows for better orientation and comparison.

To ensure the third main principle of Shneiderman (rapid, incremental and reversible interaction with immediately visible impact), the widget contains two buttons for adding a sphere and removing the last sphere. This simple mechanism enables the incremental construction of complex widgets through rotation of the volume and fitting the new widget parts to previously covered areas. Of course our system also allows for the manual repositioning of widget parts. When initiated, the user can move the selected sphere on a plane perpendicular to the view direction. However, for all examples shown in this paper, we used the initial automatic placement, as described in Section C.3.1.

C.3.3 User Interface

As we aim for direct interaction our user interface is mostly directly integrated into the rendered volume. A standard setup of out smart surrogate interface can be seen in

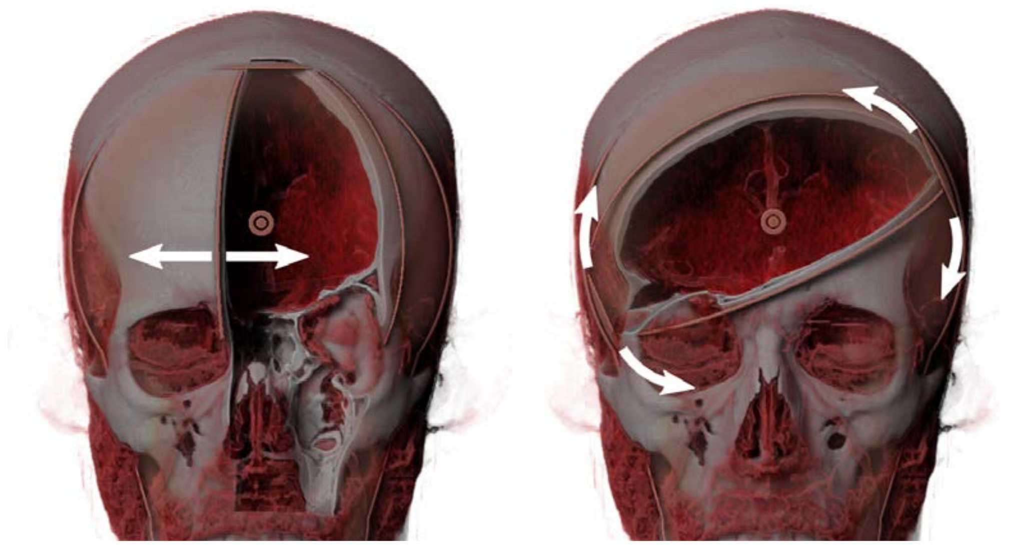


Figure C.9: The wedge can be adjusted through two independent sphere arcs (a) and can be rotated around the view vector by pulling on the poles of the wedge (b).

Figure C.12. The center component of the user interface is the smart surrogate widget that is directly integrated into the volume (1) in Figure C.12. The detached widget has three components shown in the upper region of the viewport. Component (2) consists of a slider for controlling the size of the inner volume that is unaffected by the focus region and two radio buttons to change between the main and secondary transfer function for the inner volume. Component (3) consists of two buttons for adding a new sphere to the widget and removing the last added sphere, as well as a radio button that can be activated to move the widget spheres manually. Component (4) is a slider for the opacity of the focus region and two radio buttons to switch the rendering of the focus region in the primary or secondary transfer function. Further parts of the user interface of our system are a control panel for advanced and expert settings that are hidden by default (5) as well as a transfer function editor (6).

C.4 Implementation

Our approach for the generation of smart surrogate widgets was implemented in C++ and OpenGL as an extension to an existing volume visualization framework [37]. The system provides several GPU-based rendering algorithms and the widget interface was implemented as part of a plugin. When a widget is created or extended, the visibility information is written to a separate buffer during volume rendering. The widget construction is then performed on the CPU in an asynchronous background thread, and the workflow is not interrupted. When the construction is finished, the new widget is displayed to the user. For sphere fitting, we use the RANSAC implementation of the Point Cloud Library [181].

As widgets can be geometrically complex, intersection testing is performed in two steps. First, for every frame we generate a layered depth image (LDI) from the spheres and tapered cylinders composing the widget. The LDI stores the intersection points

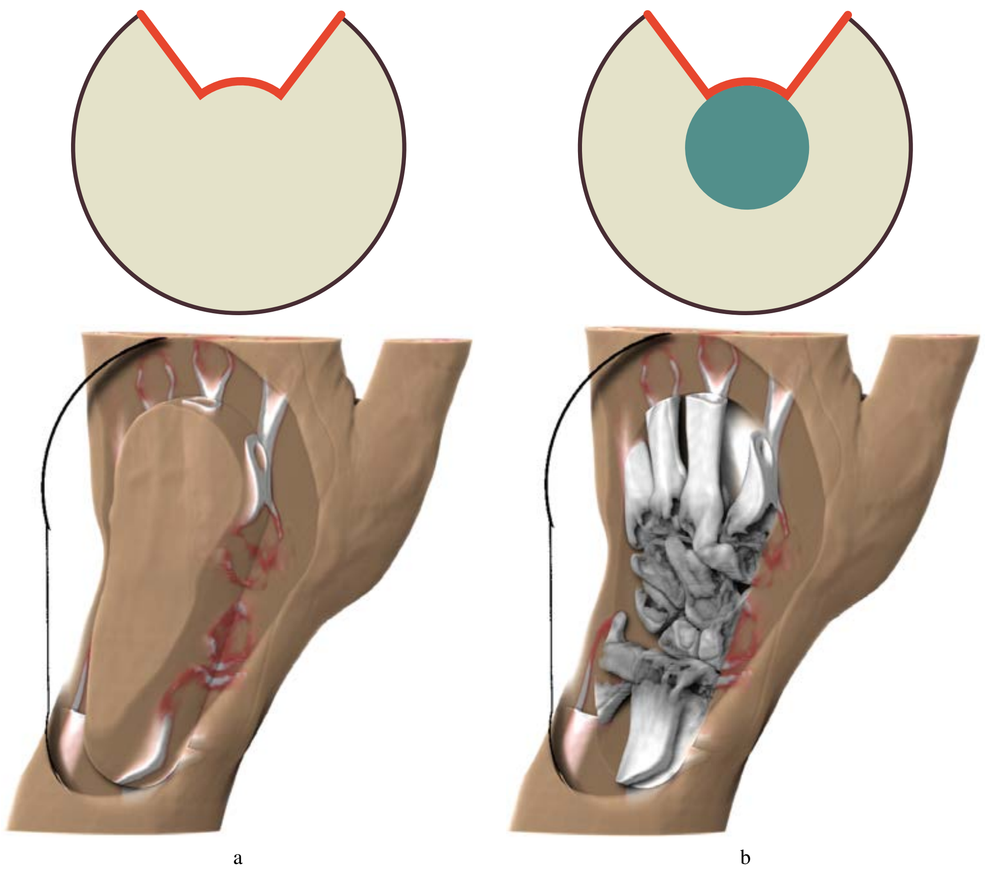


Figure C.10: The upper row shows a simplified cross section of the widget. The widget is opened with the wedge tool and the focus area is rendered transparent. The size of an area inside the smart widgets that is unaffected by the cone and the wedge manipulations can be changed with a simple slider. The computed volume can either be shown in with the same transfer function as the remaining data (a) or with a second transfer function (b).

with the individual widget elements as well as the corresponding element IDs. The intersection points are evaluated analytically, alleviating the need for tessellating the geometry.

This LDI is then used during rendering to test whether a sample position falls within the region covered by the widget. A uniform buffer containing the state of each widget element (its wedge and iris attributes, opacity multiplier, and transfer function setting) is passed to the corresponding shader. When a position is determined to lie within a particular widget element, the corresponding state attributes are retrieved using the element ID and used to test whether the point is part of the focus region as determined by the wedge or iris properties. For this, we construct a vector from the intersection point P to the sphere center or the closest point on the cylinder axis C (see Figure C.13). The dot products between the normalized vector CP and the wedge normals LN and RN give us enough information to identify the tool positions. Computing the dot product of CP and the iris axis allows us to determine if a point is covered by the iris or not. In the illustration in Figure C.13, the wedge and iris tool positions are highlighted in orange.

The generated LDI is also used on the CPU when an interaction is initiated by

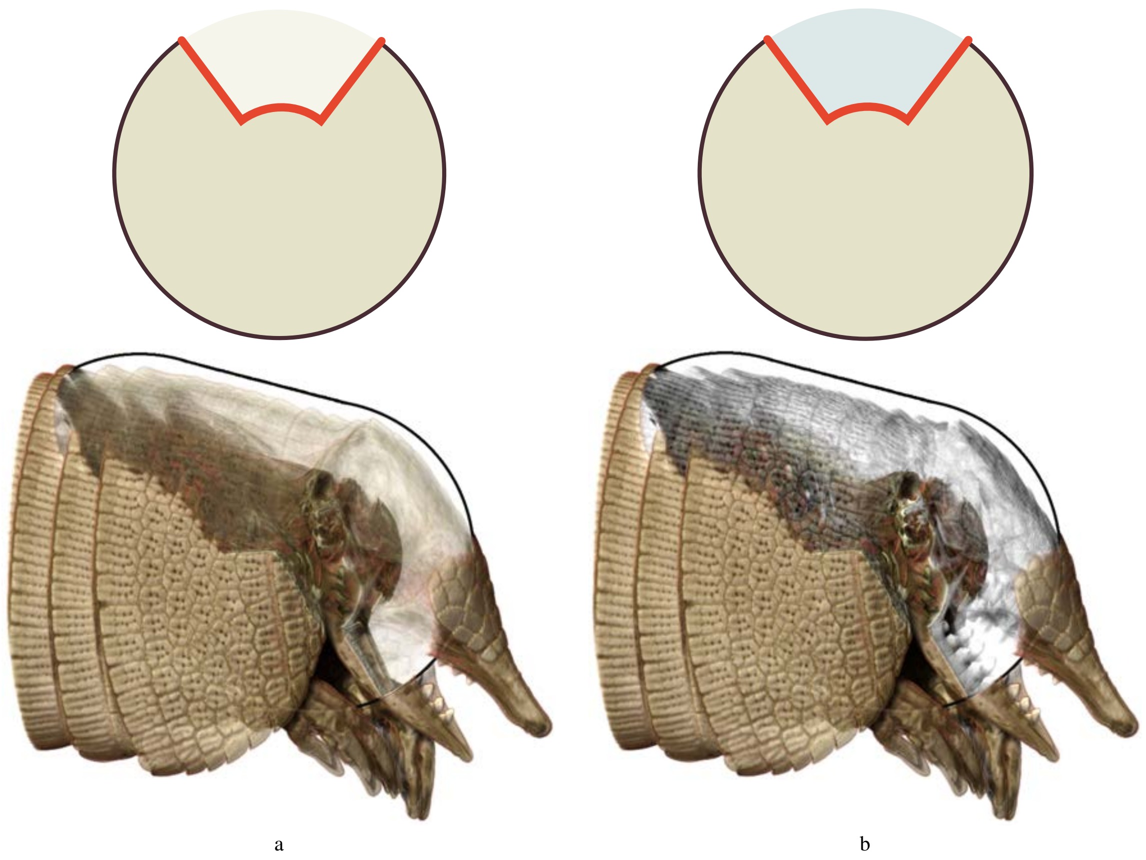
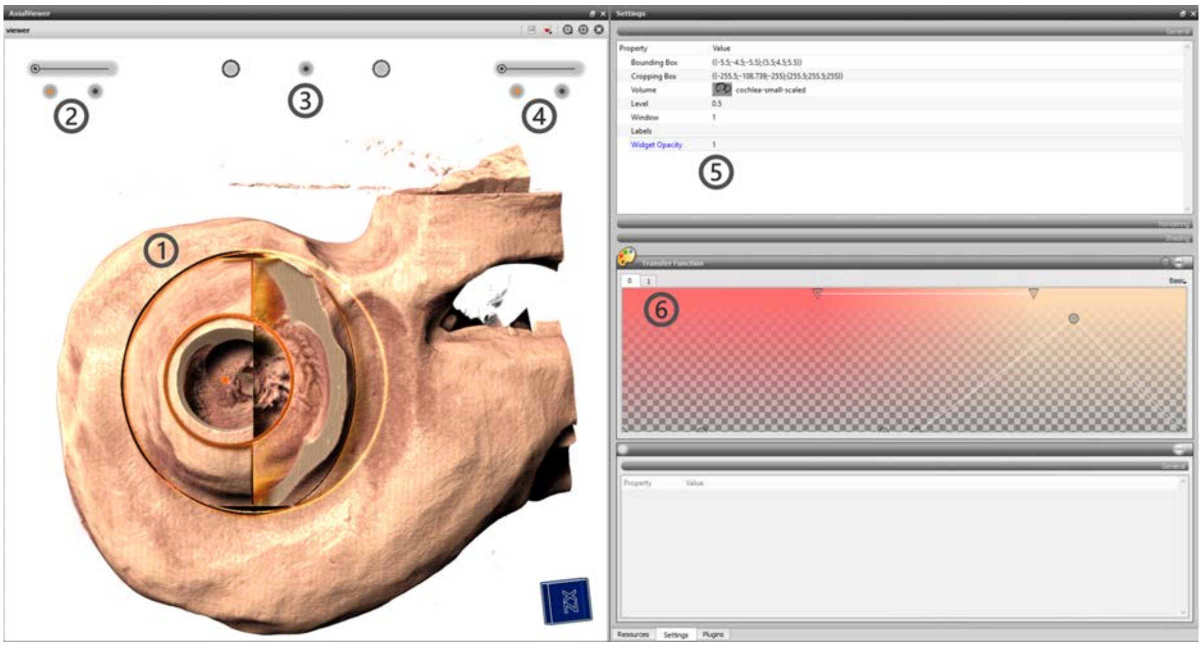


Figure C.11: The upper row shows a simplified cross section of the widget. The upper part of the armadillo is covered with a smart surrogate widget consisting of two spheres. Through a combination of cones and wedges it is now possible to look into the body by reducing the opacity inside the widget using (a) the initial transfer function or (b) the alternate transfer function.

the user. To modify the widget state according to an interaction event, we perform a lookup at the corresponding image position to check if the point lies within a sphere, a cylinder, or both. We construct the vector *CP* where *C* is either the center of the sphere or the closest point on the cone axis. Similar to the procedure shown in Figure C.13, we compute a set of dot products to determine the relationship of the point to the interaction tools and modify their positions according to the specific event.

This approach is simple and flexible, and can be easily integrated into existing volume rendering applications. For the results in this paper, we used a renderer based on the algorithm described by Patel et al. [165], but our method supports any standard volume rayasting or slicing approach. In terms of performance, the overhead introduced by our approach is comparably small. Whenever a widget is added or extended, the visibility needs to be evaluated and sphere fitting needs to be performed. The performance of this is dominated by the CPU-based RANSAC fitting. On average, the entire process takes around 300 to 400 milliseconds for the presented data sets. This could potentially be reduced by using a GPU-based implementation for the fitting, but due to the fact that this is performed asynchronously in the background, allowing the user to continue interacting without interruption, the delay is hardly noticeable and was not mentioned by any of the users during our evaluation. Furthermore, for each frame the generation of



Property	Value
Bounding Box	((-5.5;-4.5;-5.5);(5.5;4.5;5.5))
Cropping Box	((-255.5;-108.739;-255);(255.5;255.5;255))
Volume	cochlea-small-scaled
Level	0.5
Window	1
Labels	
Widget Opacity	1

Figure C.12: The standard setup of the smart surrogate interface. The numbered interface components are explained in detail in section C.3.3.

Data set	Dimensions	Regular	Widgets
Beetle	$832 \times 832 \times 494$	11.87 fps	10.67 fps
Cochlea	$527 \times 434 \times 531$	10.14 fps	9.11 fps
Pawpawsaurus	$958 \times 646 \times 1088$	4.38 fps	4.17 fps
Outlet	$425 \times 551 \times 895$	9.33 fps	8.23 fps
Supernova	$432 \times 432 \times 432$	4.87 fps	4.32 fps

Table C.1: Comparison of normal volume rendering (regular) to our approach with additional handling of smart widgets (widgets) as measured on an NVidia GeForce GTX 780 GPU with a viewport size of 967×967 pixels.

the LDI and the additional tests during rendering incur a slight overhead, as shown in Table C.1. As can be seen, there is a reduction in the frame rate, but the overall impact of our approach on the performance is not substantial. As already mentioned, we use a volume rendering algorithm that includes advanced illumination including dynamic soft shadows, which is why the frame rates in Table C.1 are generally lower than with a simpler standard ray caster.

C.5 Results

In the following, we illustrate the benefits of our approach for five different scenarios and demonstrate how smart surrogate widgets can be used to easily manipulate different types of volumetric data sets. All the examples discussed here, as all the results presented in this paper, only use automatically determined widget placement without manual adjustment.

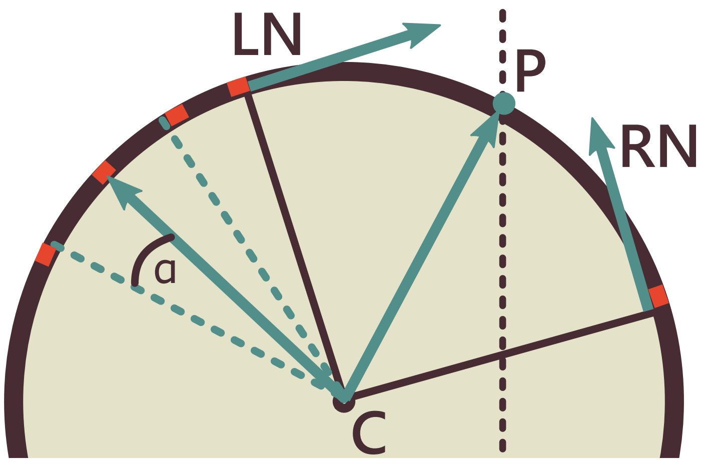


Figure C.13: To compute the positions of the interaction tools on the sphere, we construct a vector *CP* and compute the dot product with normalized vectors *LN* and *RN* to estimate if the point *P* lies in the focus region defined by the user interface tool.

C.5.1 Stag Beetle

Medical and biological illustrations often use cutaway views, where the obscuring tissue such as skin is not drawn to reveal the structures underneath. In the this example, we use a CT scan of a stag beetle. The beetle is rotated to be viewed from above. Our smart surrogate widget covers the thorax and the abdomen. With a combination of the iris tool on the thorax and a wedge on the abdomen, we can easily obtain a cutaway illustration with emphasis on the inner structures of the beetle (see Figure C.14). Note that the sphere centers are positioned below the beetle, such that the sphere surface closely matches the exoskeleton, as shown in Figure C.15. The inner volume of the widget was inflated to cover the whole widget and was displayed in a transfer function that shows the internal structures of the beetle.

C.5.2 Cochlea

The human cochlea is the auditory portion of the inner ear. It is a spirally-shaped cavity divided into three fluid-filled parts. The complex structure of the cochlea makes it an interesting subject for medical illustrations. Because of its non-regular shape, illustrators often depict the cochlea using a number of non-planar cutaways as shown in Figure C.16(a). During our evaluation, we asked the participants to recreate a similar illustration with our technique on a CT scan of the cochlea. One result of this process generated by a non-expert participant of the study can be seen in Figure C.16(b). To generate this visualization, the participant created a branched widget with four spheres in total. The cochlea was opened with the wedge tool and a volume that is unaffected by the tool inside the widget was inflated. To achieve this or comparable results the participants needed 7:12 minutes on average ranging from 3:37 minutes to 13:42 minutes. Although the data set does not show strong symmetry, seven out of nine participants stated that they found the that the task of creating this illustration was easy to complete. We discuss further details of our evaluation in Section D.6.

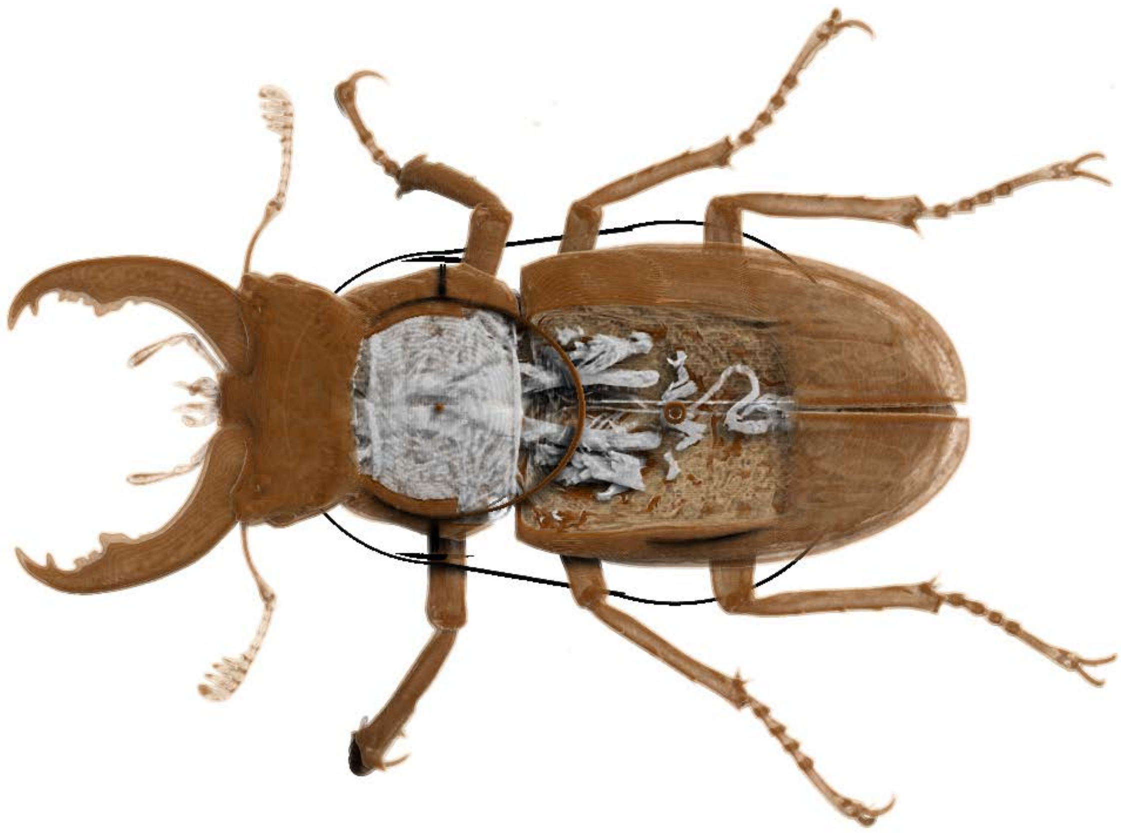


Figure C.14: Example of a cutaway illustration generated using a combination of the wedge and iris tool to reveal the inner structures of the beetle.

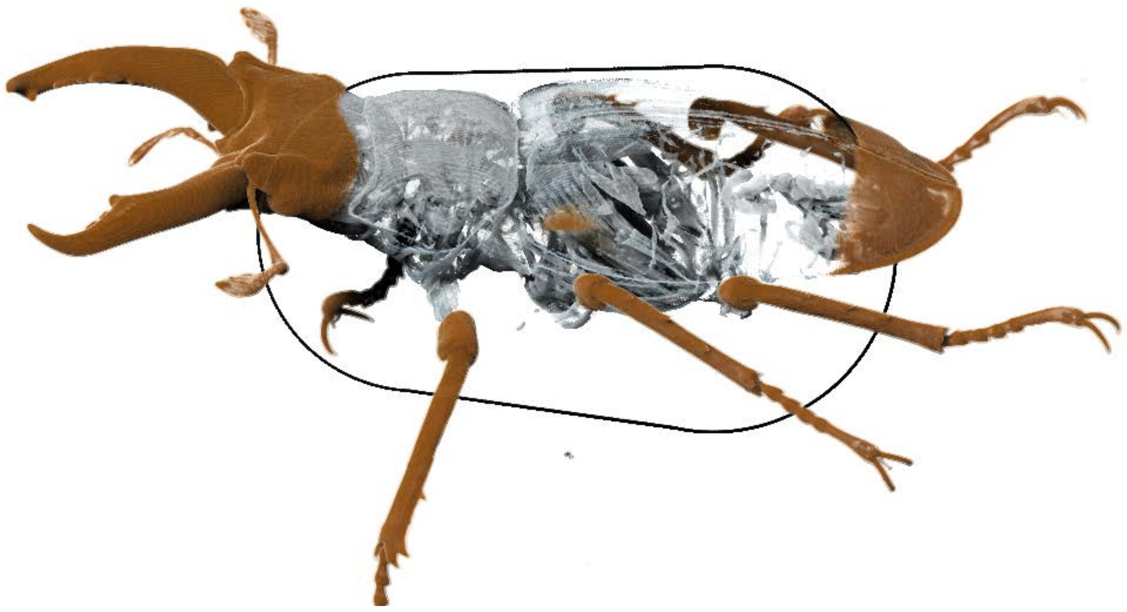
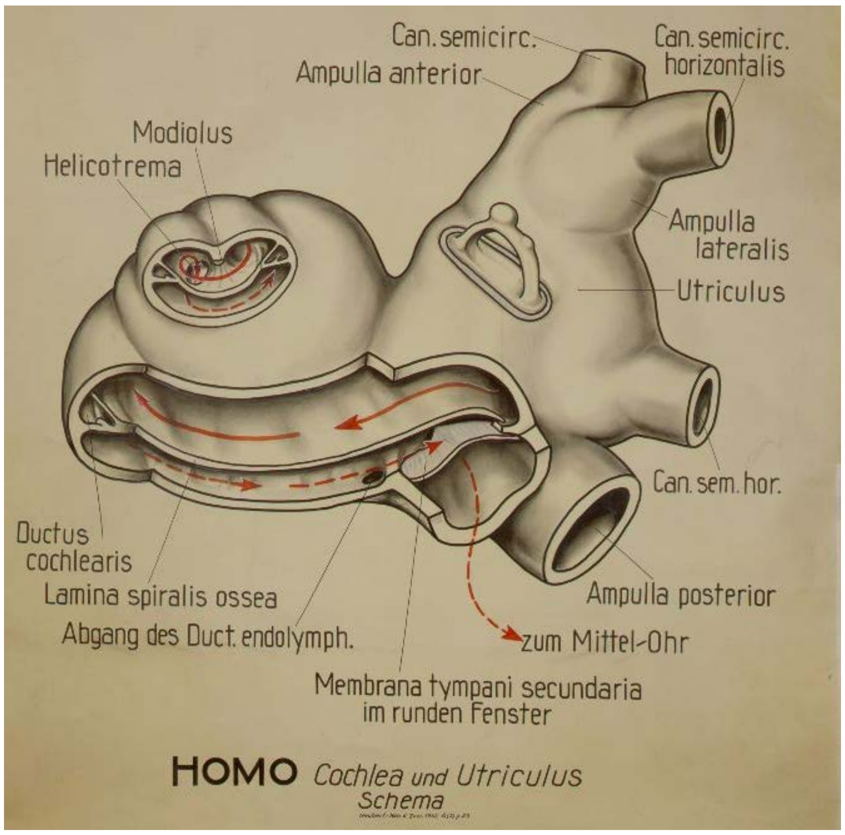
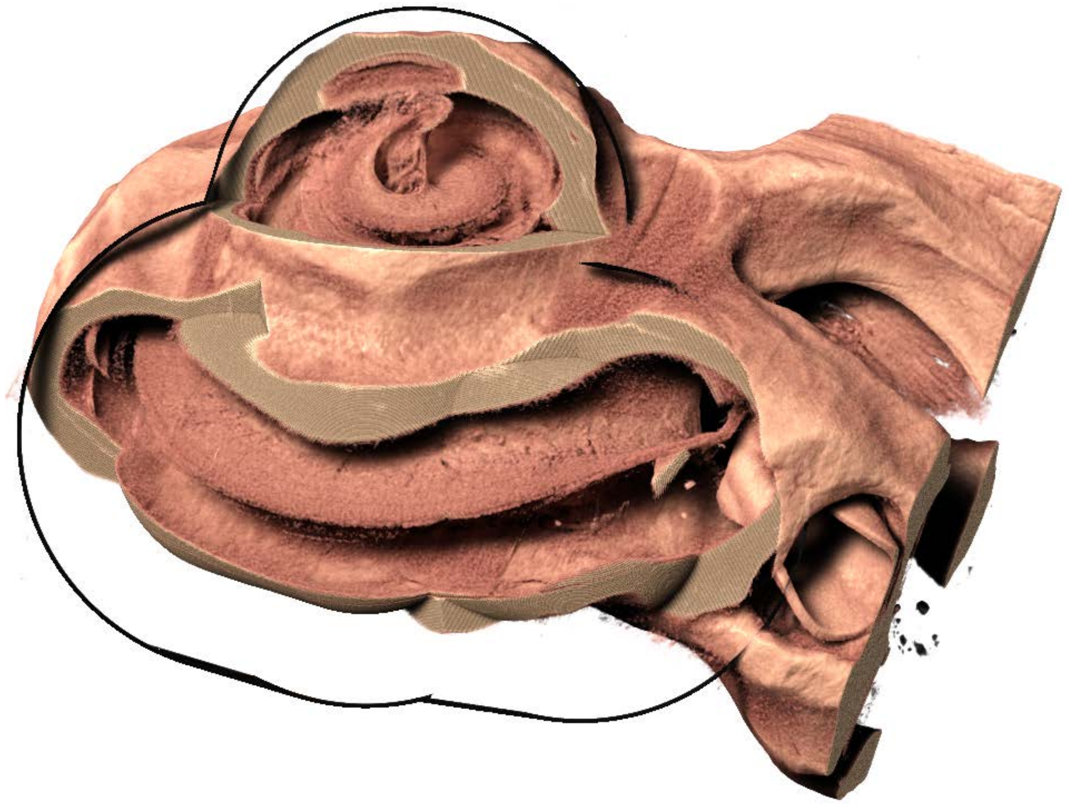


Figure C.15: Our algorithm tries to position the spheres close to the selected boundary. When flat surfaces are present this results in spheres that are bigger than the covered structure.



a



b

Figure C.16: (a) Medical illustration of the cochlea from Humboldt-University Berlin. (b) An illustrative visualization obtained with our technique by a first-time user during the user study.

C.5.3 Pawpawsaurus

Paleontologists spend the majority of their time in the field excavating and evaluating fossils. To evaluate new findings and to estimate possible kinship between the new finding and known species, they commonly compare the found bone fragments with already classified findings in a locally stored database. This can be a challenging task as subtle details can be very important for the classification. Structures such as the nasal cavities or the cavities of the inner ear and brain give indications about the relationships between species and help to reconstruct the anatomical features. The state-of-the-art procedure to solve this task is to use cutting planes, but it can be difficult to align them appropriately. The skull of the pawpawsaurus [1] is a relatively well-preserved specimen that was scanned with a high-resolution CT scanner in 2014. Multiple cavities and structures of interest in this data set are obstructed by the jaw, as shown in Figure C.17(a). To reveal the nasal cavities, we open an automatically-placed sphere with the wedge tool as demonstrated in Figure C.17(b) and (c). Next, a second sphere is placed automatically further back at the location of the brain cavities. We use the iris tool to remove obstructing bones, as shown in Figure C.17(d) and (e). The final result, which depicts the nasal cavities, brain cavity, and inner skull fragments, can be seen in Figure C.17(f). Experts in paleontology who evaluated our tool particularly valued the possibility to locally change the opacity in the wedge and iris tools, as this helped to quickly switch between detailed inspection and evaluation of the structural relationships between the fragments.

C.5.4 Outlet

Technical illustrations often use section views, which exploit symmetries in order to depict inner structures. Without additional information such as segmentation, this is difficult to achieve in volume visualization. Our widgets allow us to quickly achieve similar results without any preprocessing. We illustrate this on a CT scan of a high voltage power outlet. Using standard volume rendering, it is hard to show the inner structures and their positions inside the volume. The transfer function can be set to show a clear outer geometry hiding the inner structures (see Figure C.18(a)), to show the inner structures (see Figure C.18(b)), or it can be set to show the outer geometry semi-transparently (see Figure C.18(c)). In either case the spatial relationships between the outer and inner parts are not entirely clear. Using our smart surrogate widgets, which are automatically are placed on the top and bottom of the volume, we can open the outlet with the wedge tool and the iris tool, as shown in Figure C.19(a). The volume covered by the wedge and iris tool can be shown in a transfer function emphasizing the inner parts as depicted in Figure C.19(b). Furthermore, we can inflate an inner volume inside the widget to be displayed in the same transfer function as the wedge as shown in Figure C.19(c). This visualization depicts the structure and position of the inner parts clearly, and the shape of the outer parts can still be inferred.

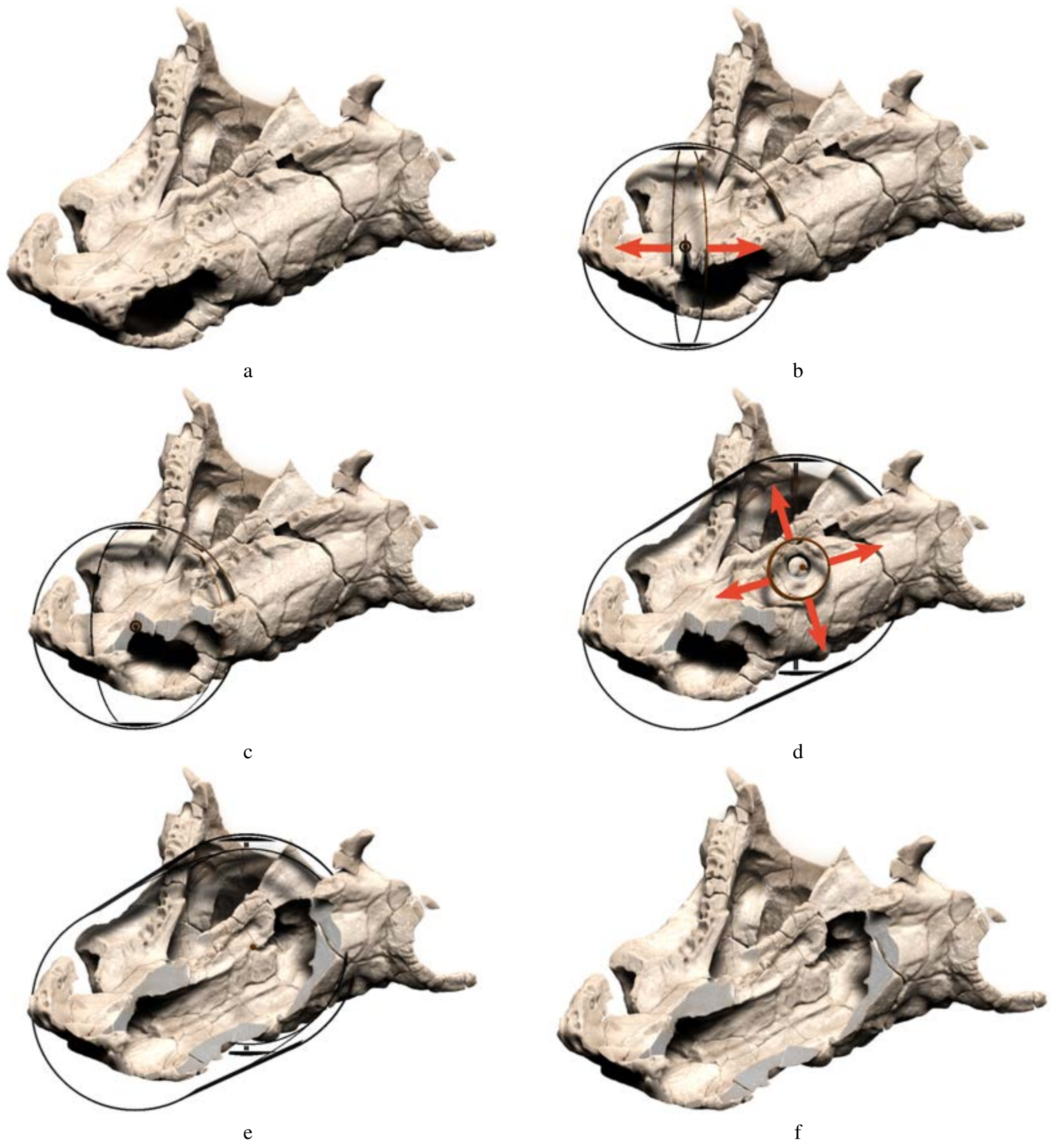


Figure C.17: (a) The skull of a pawpawsaurus seen from underneath, cavities of interest are not visible. (b) The first sphere of the widget is created, and we open the wedge tool. (c) The wedge tool is opened to reveal the nasal cavity. (d) A second sphere is added to the widget and iris tool is opened. (e) The iris tool is opened to reveal the brain cavity. (f) The final illustration showing the nasal and brain cavities.

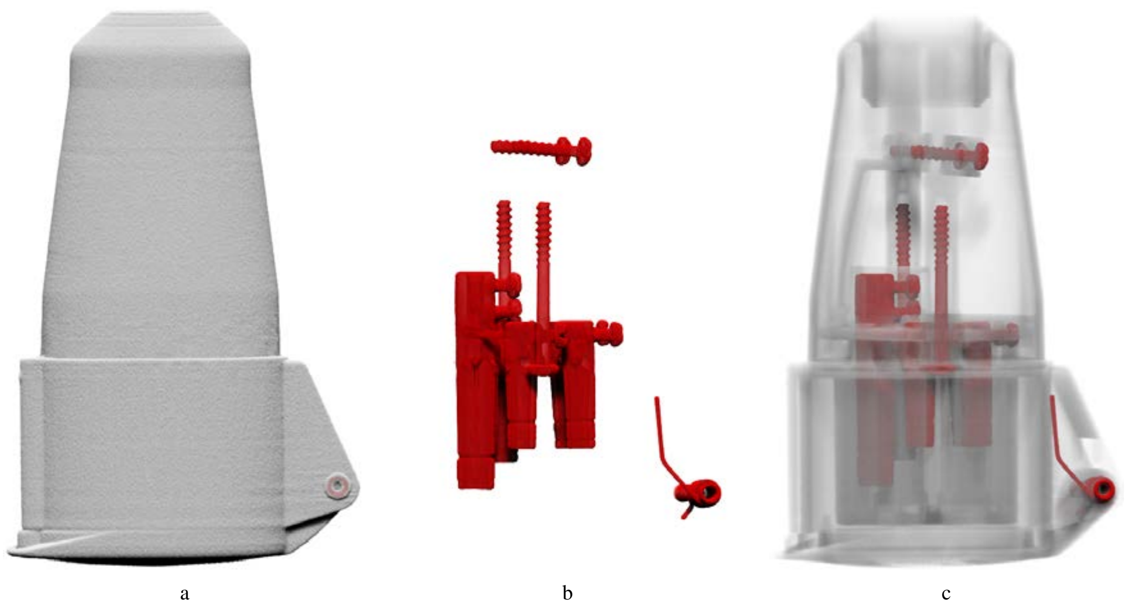


Figure C.18: Using standard volume rendering methods the transfer function can be set to (a) show the outer geometry, to (b) show the inner structures, or to (c) show both using transparency.

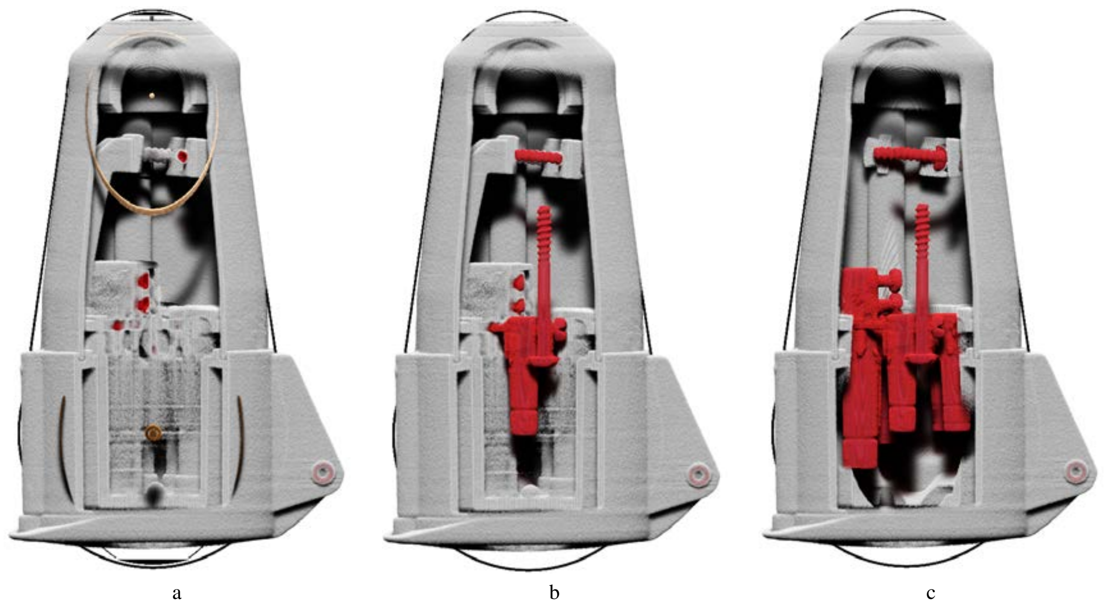


Figure C.19: (a) The smart surrogate widget covers the outlet with two spheres first fitting to the rounder top part and the lower end. To open the volume, we used the wedge tool for the lower sphere and the iris tool in the upper sphere. Both spheres are active in this image. (b) The volume covered by the iris and the wedge is shown with a different transfer function to depict the structure in front of the cut. (c) An inner volume is inflated and shown with the same transfer function as used for the wedge and iris tool.

C.5.5 Supernova

In this example, we use an entropy field from single time step of a supernova simulation made available by Dr. John Blondin at the North Carolina State University through the US Department of Energy's SciDAC Institute for Ultrascale Visualization. The data set consists of multiple interleaving layers with varying entropy levels. A volume rendering of the entropy can be seen in Figure C.20(a). Physicists are interested in the structural development of the different entropy levels over time. To study the different entropy levels, a sophisticated transfer function has to be constructed that clearly separates them. However, due to the high degree of occlusion, local manipulations can help to improve the clarity of such visualizations.

The automatically-placed widget represents a good match for the overall structure of the dataset, as shown in Figure C.20(b). Using the iris tool, we can open the data and display the inner volume with a copy of the original transfer function that is set to zero opacity except for one entropy level (see Figure C.21(a)). We can also inflate an inner volume in the widget to be displayed in the same transfer function as the volume covered by the iris tool, to see the structure of the single entropy level more clearly (see Figure C.21(b)). Changing the size of the inflated volume isolates the single entropy level, giving the user insight into where this entropy level is located in the data as well as its relationships to other entropy levels.

C.6 User Feedback

To gain feedback on the utility and usability of our approach, we conducted a qualitative study with 9 participants with ages ranging from 26 to 52 years and various backgrounds, including paleontology, scientific illustration, biology, and museum curation. All participants received a quick introduction to the basic concept of smart surrogate widgets and our prototype. They could then freely experiment with the tool and ask further questions if needed. Afterwards, they performed a task of recreating an illustrative visualization of the cochlea (see Figure C.16) and evaluated the automatic fitting of the widget against manual fitting. We then performed a semi-structured interview with questions about the usability of our tool and general impressions of the approach.

All participants with at least some experience in 3D software were able were able to recreate the cochlea illustration to their satisfaction, with completion times ranging from 3:37 to 13:42 minutes (7:12 minutes on average). However, two participants, who had no prior experience with 3D software at all, were overwhelmed by the combination of possibilities of the individual widgets tools, and commented that they would need to spend more time with the system in order to achieve the desired result. All participants stated they would consider to use the tool in their work or for educational purposes.

The second task was to evaluate the automatic widget fitting against manual fitting with position manipulation through sliders. The participants were asked to rate to the quality of the fitting on a scale from 1 ("bad fitting") to 5 ("according to my intentions"). Seven out of nine participants rated the automatic fitting with 5 and two rated it as 4. Two participants explicitly stated they were surprised how well the placement of widgets matched their intentions. Interestingly, only five out of nine participants rated the manual fitting with 5. The remaining four participants rated it as 4, 3, 2 and 1, and

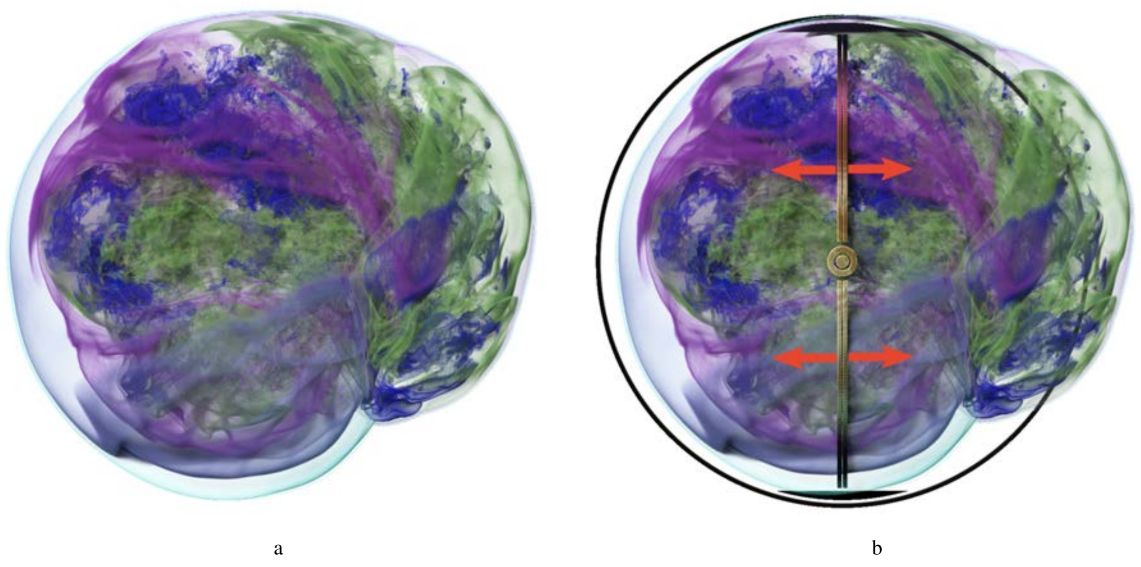


Figure C.20: (a) Direct volume visualization of a supernova simulation entropy field. (b) Automatically placed smart surrogate widget.

stated that they perceived the manual fitting as a very difficult task. To complete the manual fitting, the participants required from 2:03 to 3:17 minutes with an average of 2:13 minutes.

After the participants finished the tasks, they were asked to give qualitative feedback on the concept and our prototype. When asked if the participants would consider using smart surrogate widgets for their work one participant answered: "Yes. This a very useful for identifying hidden or rather tricky morphological features. It is a good additional tool to the cutting plane. Cutting planes can be useful for larger scale structures. This tool would be more useful for intricate structures such as the skull or smaller bone fragments". A professional illustrator stated: "Yes, I would use it at two stages of illustration: First - for research, so that I understand the shape, anatomy and functions of the object. Second - as a template for the illustration, so that I can get the right 3D shapes, angles and textures". Two participants that are specialized in multimedia exhibitions for museums stated that the smart surrogate widgets would have "high applicability" as an educational tool for museums "in a simplified version for children. For instance only with the wedge functionality".

When asked how intuitive the participants perceived the interaction with the widgets, six rated the interaction as very intuitive, stating "The interaction is very intuitive. The widget behaves as I intend it to and has a logical structure" or "The interaction is very intuitive. It is nice that you can start illustrating without thinking of where to put the sphere". One participant rated the intuitiveness as medium, stating "Activating is the widget is not intuitive, perhaps a set of buttons would be better. A button each for the action you will perform: wedge, iris, move, and scale. However, when actually using the tool, it is easier when you can toggle on and off the widgets in the interface". Two participants rated the interaction as "not very intuitive" and "not intuitive", suggesting that a short animation of the functionality when hovering over the widget parts would improve the intuitiveness. With respect to the functional completeness of the

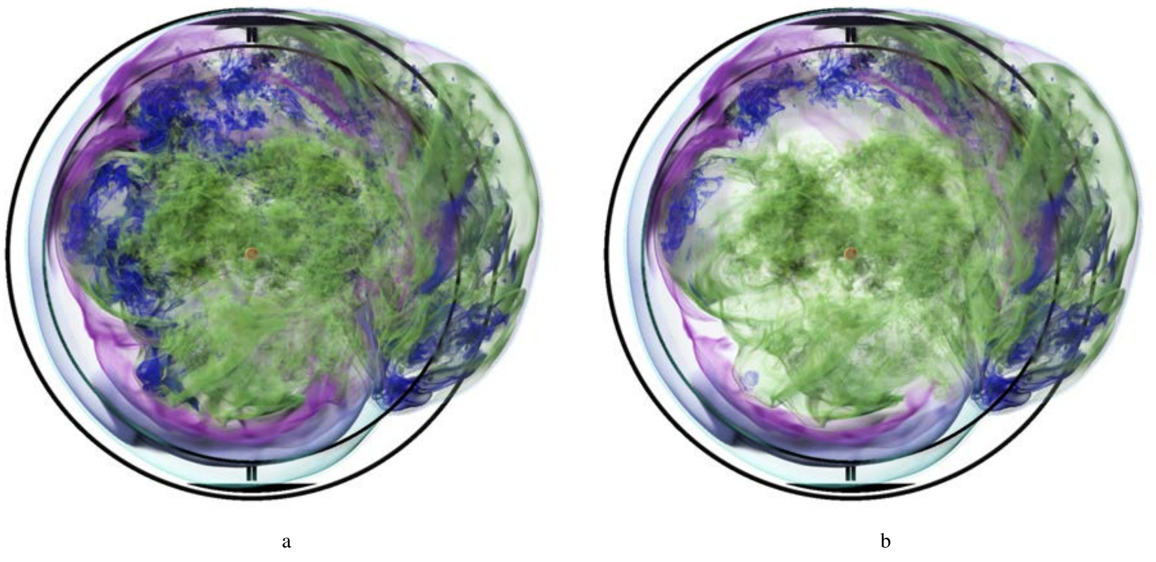


Figure C.21: (a) The smart surrogate widget is opened to remove outer layers of the data. (b) The inner volume of the widget is displayed with only a single entropy level.

smart surrogate widgets, most participants agreed that the functionality is sufficient. One experienced participant stated that he would like to be able to change the transfer function for each sphere separately. Another participant wished for an undo or reset button for the widget tools, so that he could easily close the whole volume again.

One negative comment noted by several participants was that they felt that the interaction handles were slightly too small, making it somewhat difficult to hit them when interacting quickly. Furthermore, many participants commented that they also consider the interface to be well-suited for touch-based interaction. While the feedback was gathered using conventional mouse interaction, our system actually supports touch interfaces as well. While this was not part of the task description, we noticed that some participants used the smart surrogate widgets as an exploration tool similar to a magic lens. With only one sphere, the widget was opened with the wedge tool and moved around in the volume to study the inner structure.

Overall, the feedback was very encouraging and we believe that our approach can provide a simple yet powerful tool for interacting with volumetric data. We are currently exploring the possibility of creating an interactive museum installation based on our prototype, and hope to use this opportunity to gather further data from a much larger user group.

C.7 Discussion

In our experiments, we found out that smart surrogate widgets provide an intuitive and fast method for the creation of expressive visualizations as well as data exploration. We deliberately chose a minimalist design, avoiding the introduction of additional clutter or occlusion, and only provide a limited set of interactions. Nonetheless, we found that the simple tools presented in this paper can be efficiently combined to quickly perform complex operations that would be difficult to achieve using conventional approaches. While our widgets are composed of two simple primitive shapes (spheres and tapered cylinders), they adapt well to a variety of spatial structures. In fact, initially we attempted to incorporate a larger set of basic widget elements, such as ellipsoids or rectangular prisms. However the resulting interface was significantly more complex without a noticeable increase in functionality.

Our approach can be easily integrated into existing visualization pipelines, and only introduces a small overhead. While the sphere fitting computation can take several hundred milliseconds, it is an infrequent background operation that does not negatively affect overall usability. As widget construction is based on visibility, our approach is not restricted to any particular type of data or transfer function model. Moreover, even though in this paper we only considered scalar data sets, we also see great potential for using our widgets in the context of multimodal data. For instance, the focus region of a widget could be used to display a second modality instead of using an alternate transfer function.

C.8 Conclusion

In this paper we presented the new interaction concept of smart surrogate widgets for direct volume manipulation. Inspired by the notion of interaction through surrogate objects, we designed a visibility-driven approach for incrementally building a complex volume-covering widget consisting of only two primitive parts. We discussed our design on several examples and showed how it facilitates quick and easy volume manipulation. To evaluate our approach we conducted a qualitative user study. Our results show that smart surrogate widgets represent a promising and powerful approach suitable for non-experts in various scenarios.

Acknowledgements

The research presented in this paper was supported by the MetaVis project (#250133) funded by the Research Council of Norway.

Paper D

LinesLab: A Flexible Low-Cost Approach for the Generation of Physical Monochrome Art

Sergej Stoppel and Stefan Bruckner

University of Bergen, Norway

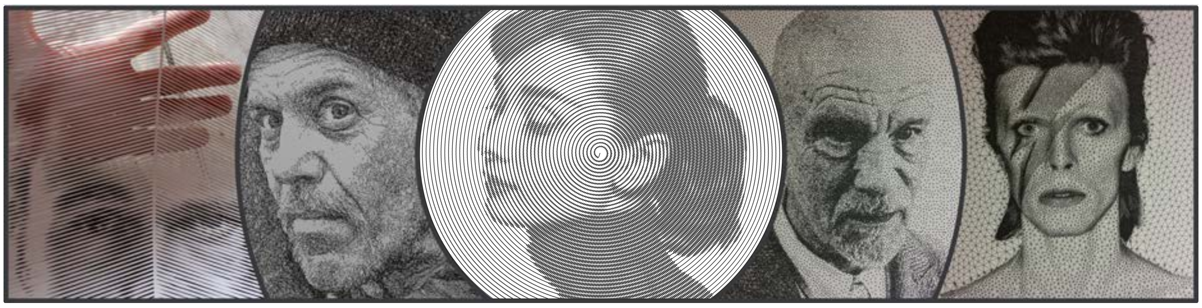


Figure D.1: Artistic examples generated with our method. From left to right: Paper cutout, dashes drawing, single line spiral drawing, stippling, triangulation drawing.

Abstract

The desire for the physical generation of computer art has seen a significant body of research that has resulted in sophisticated robots and painting machines, together with specialized algorithms mimicking particular artistic techniques. The resulting setups are often expensive and complex, making them unavailable for recreational and hobbyist use. In recent years, however, a new class of affordable low-cost plotters and cutting machines has reached the market. In this paper, we present a novel system for the physical generation of line and cutout art based on digital images, targeted at such off-the-shelf devices. Our approach uses a meta-optimization process to generate results that represent the tonal content of a digital image while conforming to the physical and mechanical constraints of home-use devices. By flexibly combining basic sets of positional and shape encoding strategies, we are able to recreate a wide range of artistic styles. Furthermore, our system optimizes the output in terms of visual perception based on the desired viewing distance, while remaining scalable with respect to the medium size.

This article was TODO

D.1 Introduction

Early after the development of the computer, artists discovered computational art as a new discipline. Mathematicians like Georg Nees [7] and Frieder Nake [6] advanced computational art by generatively creating complex geometric figures and drawing them with an axial plotter on a paper canvas. As the technology advanced, the output devices for computational art became more complex, such as industrial devices equipped with a multi-axial arm in the case of *e-David* [137] or sophisticated 3D printers that output multiple layers of paint as used in *The Next Rembrandt* [10].

While these advances in technology are capable of generating remarkable results, they rely on complex and expensive customized hardware setups. Several manufacturers addressed the needs of hobbyists and developed affordable axial plotters for home use. While these devices are more limited than their high-end counterparts (e.g., with respect to medium support and process feedback), they nonetheless are capable of generating high-quality results at often only a small fraction of the cost.

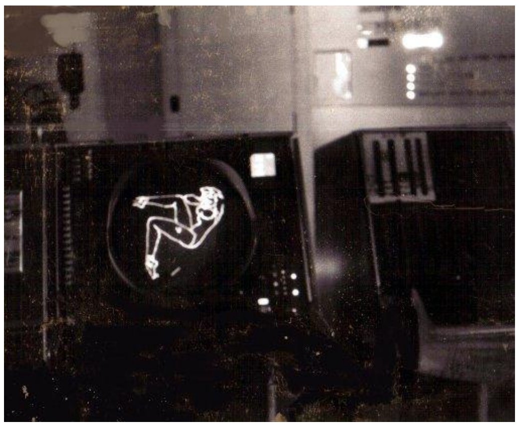
This new generation of hobbyist devices has given rise to a renewed interest in customized computer art for recreational purposes that increasingly penetrates the mainstream. However, there is still a lack of capable software that addresses the needs of this market, provides a wide degree of stylistic choices, while still taking into account the limitations of the available devices.

We present a flexible automated system that transforms images into stylized drawings or cutouts, by creating a set of commands that can be interpreted and performed by conventional hobby plotters and cutting machines, such as Sihlouette Studio [13] or Cricut [4]. The main contributions of our work can be summarized as follows: We identify the limitations and capabilities of home-use plotters and analyze the design space for line art and paper cuts. Based on the resulting constraints, we present a modular automated system that is capable of creating a wide variety of art-inspired styles while being scalable with respect to the medium size. Our system supports the synthesis of new styles by combining sets of positional and shape encoding strategies, and can generate line and cutout art. Furthermore, being targeted at the generation of physical artwork, our approach can optimize its output based on the targeted viewing distance.

D.2 Related Work

The areas of computational art and computational aesthetics have been extensively studied over the years. Since we focus on physical artwork, we first cover drawing machines in art and then continue to discuss related work on non-photorealistic rendering methods and image stylization.

Simple devices to support the drawing process were constructed already in the early 15th century. Later automatic drawing machines that could produce complex geometric patterns with ease, such as the Harmonograph, were introduced. Jean Tinguely [9], for example, created a number of sophisticated drawing machines with complex repeating stroke patterns. While these machines could only generate fixed patterns, this changed with the development of computers. The roots of computer-generated art can be traced back to the late 1950s. The pin-up girl at the SAGE air defense system, shown in Figure D.2(a), was probably the first drawing to appear on a computer screen. The



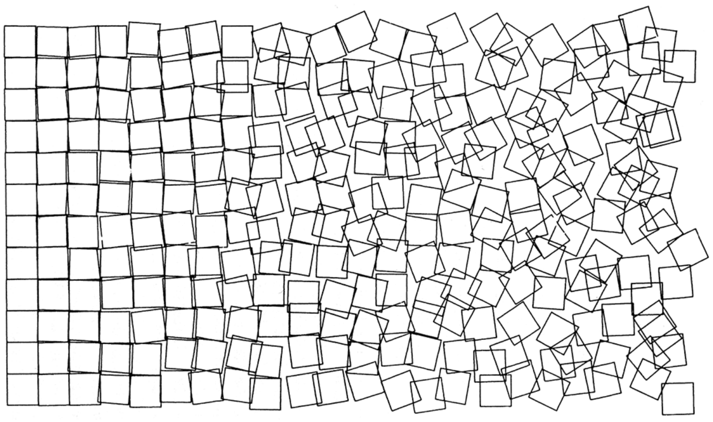


Figure D.2: (a) The pinup girl of SAGE, the first documented case of computer generated art. (b) Schotter (1965), by Georg Nees – one of the first physical drawings made with by a drawing machine.

term computer art was first used by Edmund Berkeley in 1962, which inspired the first Computer Art Contest in 1963. This annual contest propelled the development of computer-generated art. Early pioneers of computer graphics like Georg Nees [7] or Frieder Nake [6] used plotters to create the first artworks that were constructed digitally and then physically drawn by a machine (see Figure D.2(b)). Over the subsequent years a number of artists used machines to create – often abstract – artwork.

In recent years, artists started to develop means for more realistic machine-generated paintings. Harold Cohen [8] built sophisticated painting machines that could represent digital image content, but his main focus was still on abstract art. Pindar Van Arman [11] experimented with AI-controlled painting robots, collectively referred to as *cloudpainter*, to create original compositions. Tresset and Fol Leymarie [214] described *Paul the Robot*, a robotic installation for face drawings. A more general setup was achieved with *e-David*, a feedback-guided painting robot, whose rendering techniques were discussed by Deussen et al. [53] and Lindemeier et al. [137].

Galea et al. [75] presented a stippling method with flying quadrotor drones. Prévost et al.[171] used an interactive optimization process to guide the user in creation of large scale painting with spray paint. Jain et al. [99] developed a force-controlled robot that was able to draw on arbitrary surfaces. A similar approach was taken by Jun et al. [112], where a humanoid robot capable of drawing on a wall was developed. Calinon et al. [41] used existing tools to create a humanoid robot to draw artistic portraits for entertainment purposes. Inspired by these results, many hobbyists and developers created algorithms and machines for drawing purposes. The project *Caravaggio* [3] by Michele Della Ciana is particularly noteworthy, as it served as one of the inspirations for our work. *Caravaggio* is a custom-built polargraph drawing machine that produces artwork composed of a single continuous path. A single illustration created with this approach requires 12 to 24 hours to complete. One of our aims was to reproduce the compelling results generated by this approach for arbitrary digital images on simple and affordable consumer devices.

Research in the areas of non-photorealistic rendering and computer-based image stylization are closely related to our work, as they typically aim to reproduce different types of artistic techniques or media. A common strategy to approximate an image is to use stroke-based rendering such as brush stroke techniques. Most of these techniques use low-level abstraction based on local image properties, such as done by Wen et al. [229] for color sketch generation through a segmentation algorithm. In some approaches, the stroke placement is influenced by higher-level parameters. Shugrina et al. [191] presented "empathic painting", an interactive painterly rendering approach whose appearance adapts to the emotional state of the viewer in real time. A similar approach was taken by Colton et al. [48], who developed a stroke-based rendering algorithm which heightens the emotions of the depicted person. Ostromoukhov and Hersch [161] used a multi-color dithering approach to generate color images made of artistic shapes. Many approaches convert a raster image into vector graphics, such as work done by Jeschke [106], who introduced a generalized formulation for Diffusion Curve Images. Durand et al. [58] introduced an interactive system that supports the user in the creation of drawings from photographs in a variety of styles.

Our work instead focuses on monochrome styles such as halftoning, stippling, or hatching. Pnueli and Bruckstein [170] among the first to discuss gridless halftoning techniques. Ahmed et al. described two line-based halftoning techniques, via recursive division [18] and line amplitude modulation [20]. Ahmed [19] also addressed the general brightness/contrast problem of line-based halftoning methods and proposed to use error diffusion as preprocessing step. Pang et al. [164] presented a structure-aware halftoning technique, that aims to preserve the structure and tone similarities between the original and the halftone images. Later Chang et al. [43] used an error-diffusion approach to create visually similar results as in the method by Pang et al. but with significantly decreased computation time. Pedersen and Singh [166] addressed the synthesis of organic maze structures, and Xu and Kaplan [239] discussed a set of algorithms for designing mazes based on images. Kaplan and Besch [114] described how to construct drawings with a continuous line by solving the traveling salesman problem. Chiu et al. [45] extended the TSP algorithm to a tone- and feature-aware path creation method for circular scribble art. Hiller et al. [88] discussed generalized methods for computer-generated stippling by exploring primitives other than points. Bartesaghi et al [30] proposed an approach for generating hatching drawings based on multiple images with fixed positions and angles to the camera. The work of Son et al. [192] introduced the notion of feature-oriented structure grids for directional stippling to determine the position and orientation of rendering primitives. Xu et al. [240] presented a technique for procedurally-generated two tone paper cut designs with guaranteed connectivity. For an extensive overview of artistic stylization techniques, we refer to the state of the art report by Kyprianidis et al. [128].

Several studies have been performed to evaluate the aesthetics of stippling algorithms compared to human artists, as well as to each other. Maciejewski et al [141] compared hand-drawn and computer-generated illustrations using image processing techniques. Spicker et al. [193] investigated if the perceived abstraction quality of stipple illustrations is related to the number of stipples using a crowdsourced user study. A comprehensive study of digital stippling was done by Martin et al. [146].

While our approach is able to generate similar results to some of the work above (e.g., line-based halftoning), we specifically focus on the physical realization of the stylized image and hence must account for technical restrictions of a plotter that were not considered in previous work. Furthermore, our system is based on a unifying frame-

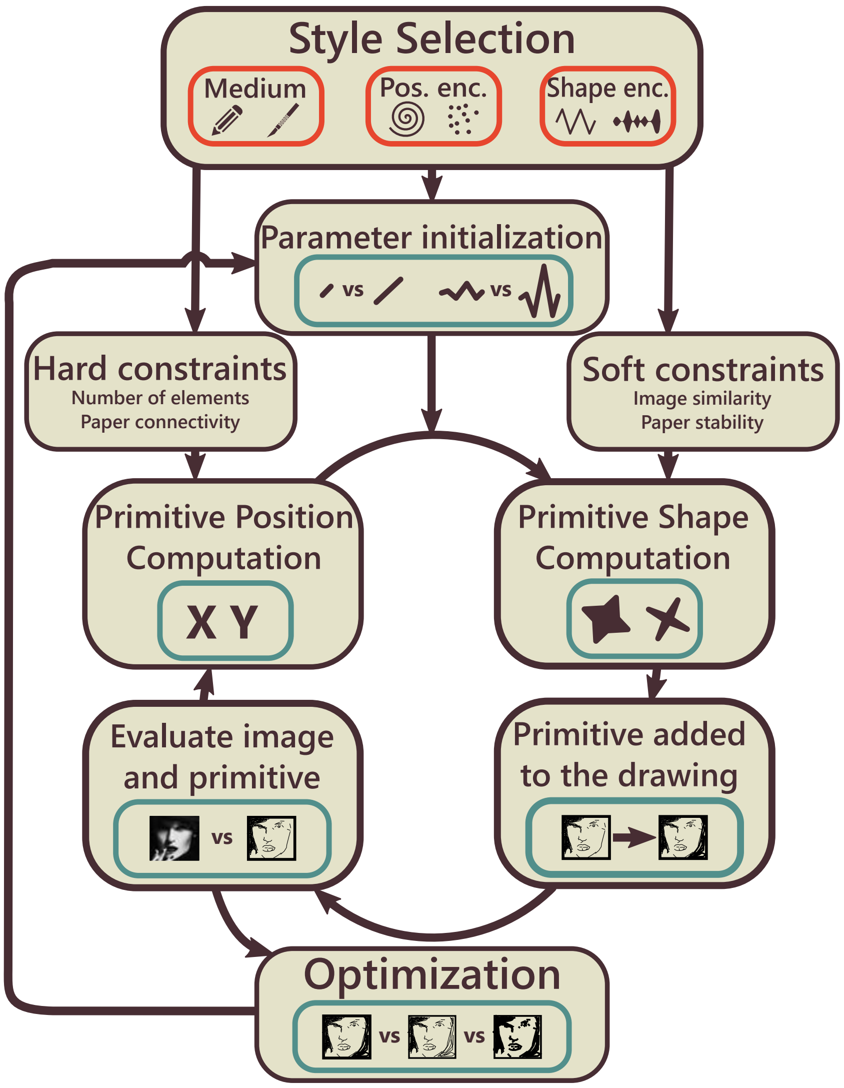


Figure D.3: Overview of our system. The user input is annotated with orange boxes, turquoise boxes illustrate the algorithm results.

work that allows for the creation of a wide range of styles as well as the easy integration of new abstractions.

D.3 LinesLab System

The LinesLab system was designed for users with the intent of creating computational artworks with little programming knowledge. Computational artists typically use dedicated software such as Processing [12] to create their works. While such tools are very flexible, they usually require the user to write the code from scratch and offer no support in finding suitable parameters for the visual primitives. This can lead to a long trial and error process where the user learns how to create appealing art through experimentation. Our goal is to create a system that offers comparable flexibility for the creation of monochrome line art or cutout styles, while simultaneously reducing the user workload of finding suitable parameters for the visual representations. Furthermore, we aim for a system that is well suited for novice users, by allowing for the synthesis of a wide range of artistic styles from a relatively small set of simple basic modules that can be easily extended by the user.

To achieve these goals, we constructed the LinesLab system as a set of exchangeable plugins for style selection. An overview of our system is depicted in Figure D.3 with arrows annotating the information flow in the system. The orange boxes denote user input or user-defined processing steps. To create an artwork in a particular style, the user has to select the used medium, as well as the medium size and encodings for the shape and positions of the visual primitives. The LinesLab system automatically optimizes the visual primitive parameters in a nested optimization loop to create an artwork perceptually close to the input image, while following constraints set by the plotter hardware.

The first step in our system is style selection. This is the only part of the system that actively involves the user. First, the user is required to select the medium (drawing or cutout), which defines a basic set of global constraints for the optimization process. Next, the user can define the style through a choice of positional encodings, such as predefined positions on a regular grid or iteratively computed positions, and shape encodings, such as circles, points or lines. Entirely novel encodings can be added by creating new plugin modules. If a new encoding is introduced, the user has to denote which parameters are to be optimized in the optimization loop and the range of the parameter space. For example, parameters that define the density of the samples or the size and shape of the primitive are natural choices for the optimization. Our system is able to handle any function as long as the output can be described by a set of curves. In addition, the user is able to modify global settings such as the paper size, viewing distance, pen tip width, or maximum number of primitives. As default values, the system uses a paper size of A4, with a viewing distance of 1 m, a pen tip width of 0.5 mm, and no upper limit on the number of primitives.

The optimization process then proceeds as the execution of the two nested optimization loops depicted in Figure D.3. The outer loop samples the style parameters and simulates the drawing process for each of the samples. The inner loop then seeks to place primitives and adjust their shapes such that they maximize image similarity (and paper stability, in the case of cutouts). During drawing simulation, in the first step of the loop a suitable position for a primitive is found. In the case of predefined positions (e.g., primitive placement on a fixed grid), the system computes the positions only once based on the current parameter settings. If the positions are computed iteratively, the system evaluates if the new position would violate the hard constraints and computes a different position if it does. In the next step, the shape of a primitive is computed according to the local properties of the input image and the soft constraints. After the position and shape of a primitive have been computed, the system evaluates the resulting primitive for violations of the constraints. If the primitive fulfills all the constraints, it is added to the drawing. If an upper limit for the complexity was specified by the user, our system checks if this limit has been reached and terminates the loop accordingly. If the limit has not been reached, our system evaluates the difference between the simulated drawing and the input image. If a sufficient similarity has been achieved, the drawing is considered to be finished and the drawing simulation loop ends. The resulting drawing is then evaluated against all previously generated samples in the outer loop.

In the following sections, we discuss the concept of the visual encoding utilized in

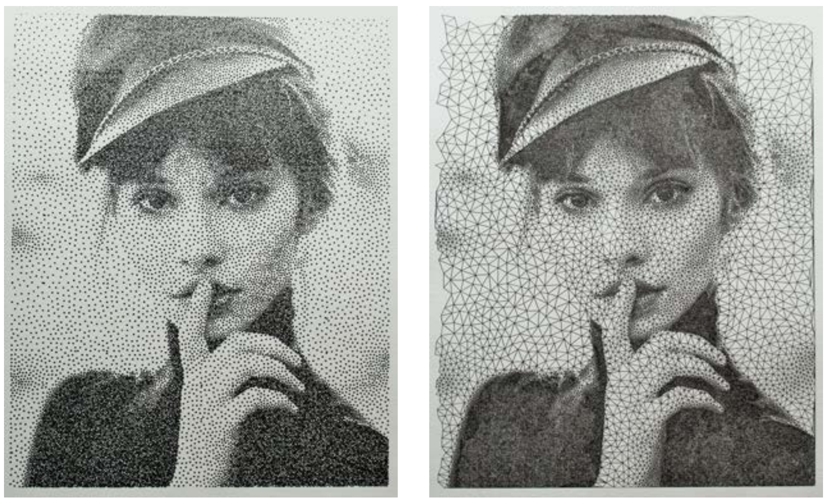


Figure D.4: Two examples of positional encoding: (a) Stippled image, (b) triangulation of the stipple positions.

LinesLab and review the individual parts of our system in detail. We provide examples of different styles and explain their formulation as a pseudocode in the results section.

D.3.1 Encoding of Visual Primitives

As noted before, our goal is a flexible system for the creation of a wide range of monochrome line art and cutouts. In order to reproduce a wide variety of artistic styles, our approach aims to represent the content of a digital image (in terms of its tonal content) using a set of generalized primitives. The basic strategy for approximating the tonal content of the source image is then determined by a user-specified choice between a set of encodings that control the subsequent global optimization process.

Many visual encodings exhibit a significant degree of overlap. Halftone images, for example, are usually generated by placing different sized circles on a fixed grid. Often, only the grid will vary for different styles while the encoding as circles stays fixed. To avoid repetitive definition for the visual primitives and to further reduce the encoding complexity, we split the definition of visual primitives into two parts. Conceptually, our approach is a stroke-based method, and we draw our inspiration from traditional line art. Most drawing techniques consist of two main abstractions, the position of a pen stroke and the drawn patterns. In fact, art students often train their skills by restricting themselves to one specific abstraction. Following this concept, we distinguish between two basic types of encodings for visual primitives: *positional encoding* and *shape encoding*.

Positional Encoding

Positional encoding denotes all techniques that recreate the original image with geometric shapes with varying positions, while the essential shape of the drawing primitive stays fixed and is not dependent on the image intensity values. The density of the primitives conveys the gray values of the image. A stippled image, as shown in Figure

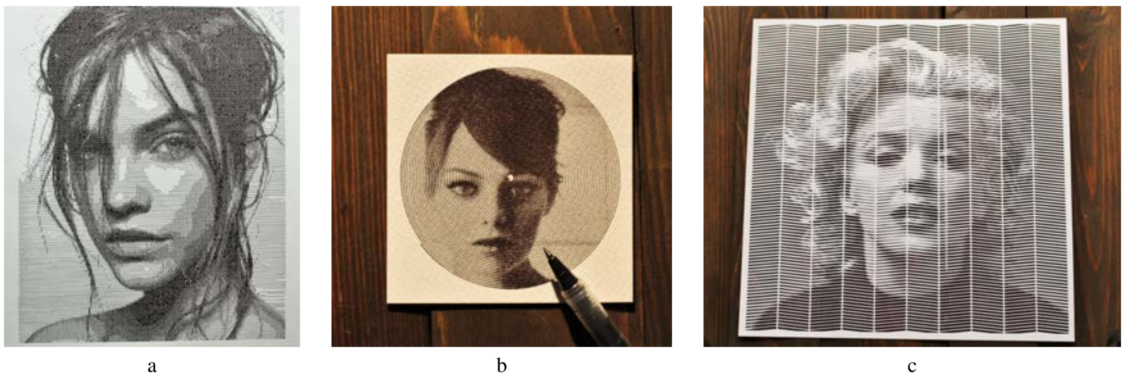


Figure D.5: Three examples of shape encodings: (a) Halftone image, (b) amplitude-modulated halftoning, (c) width-modulated halftoning realized as a paper cutout. Note that the positions of the graphical elements are fixed. The image is encoded through shape difference only.

D.4(a), is the most basic example of positional encoding. However, positional encoding can have more complex forms, as shown in Figure D.4(b), where primitive positions are triangulated to create a new style.

Shape Encoding

With shape encoding we denote techniques that use a direct mapping of the image intensity in a local neighborhood to a geometric property, such as size or orientation, on a fixed position. Halftone images, as shown in Figure D.5(a), are simple examples of such direct encodings. However, shape encodings can have various forms and the position of the encoded object does not have to be aligned to a regular grid. In Figure D.5(b), the input image is encoded through an amplitude-modulated cosine function on an Archimedean spiral. Another example for shape encodings are paper cutout images, where the illustration is created by cutting out segments of varying thickness from the paper, as shown in Figure D.5(c). The image intensity values are directly encoded as thickness of the cutout.

Design Combinations

Based on these two types of encodings, we can proceed to generate new drawing styles by combining positional and shape encodings. This allows us to express a large number of different artistic styles as combinations of positional and shape encoding with varying degrees of freedom. As illustrated in Figure D.6, most monochrome line art can be represented as a combination of position and shape encoding with varying degrees of freedom for each encoding. For instance, the images in Figure D.4 can be seen as combinations with a high degree of freedom for the positional encoding and a low degree of freedom for the shape encoding. In contrast, the images shown in Figure D.5 have a high degree of freedom in the shape encoding and a low degree of freedom in the positional encoding. Likewise, it is possible to generate styles that have a high degree of freedom in both positional and shape encoding, and their exact weighting results in countless possibilities. For example, combining positional encoding in form of stippling and shape encoding in form of circles sizes yields a drawing as in Figure D.7(a).

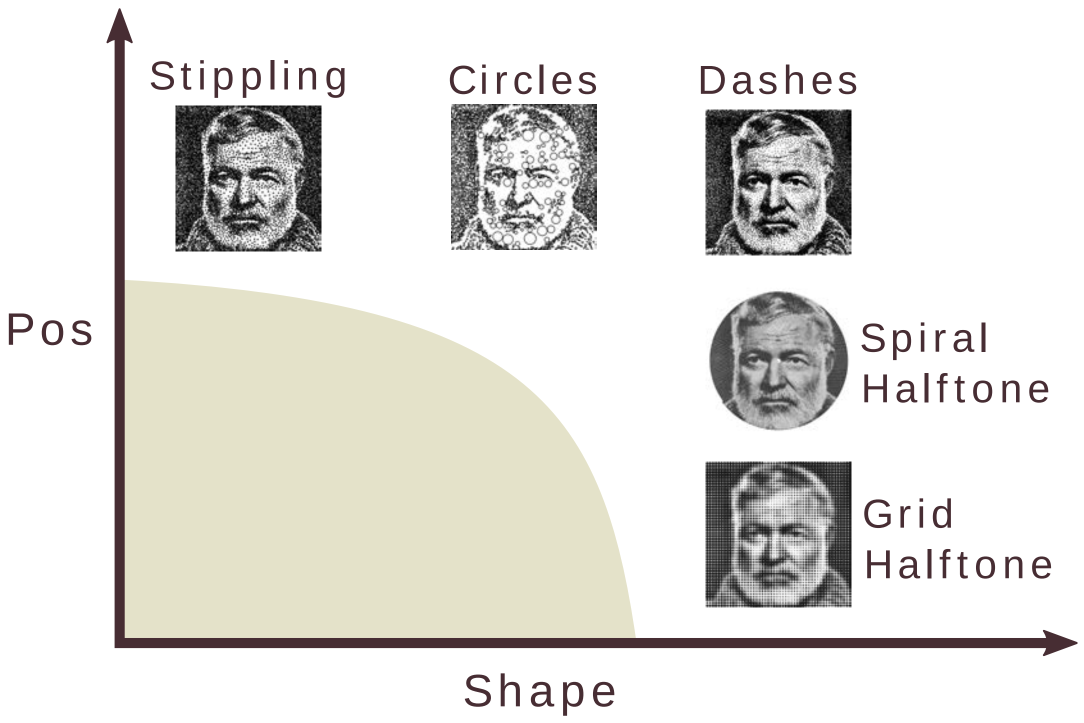


Figure D.6: Most drawing styles can be represented as a combination of positional and shape encoding with varying degrees of freedom. Designs with low degree of freedom in both encodings (gray area) are not suitable for closely representing input images.

Even expressive paintings, such as the works of Vincent van Gogh, can be seen as combinations of positional encodings and shape encodings, where the position, orientation and the size of the brush stroke carry information about the color and structure of the depicted object. Translating Van Gogh's style to a monochrome style produces a drawing as depicted in Figure D.7(b). In this example, the spatial encoding of stipples was combined with a shape encoding of lines, where the orientation of the line is dependent on the local variance in the input image.

D.3.2 Device Restrictions

Having discussed the definition of the visual primitives, we want to briefly outline the possibilities and limitations of a hobby plotter for visual representations. The design of a hobby plotter is conceptually very similar to an XY-plotter with an on/off state for the Z-axis. In contrast to common XY-plotters where the drawing instrument is moved freely in X and Y direction, most hobby plotters move the blade or the pen along the X axis only. The processed medium is moved along the Y axis. The working area of hobby plotters is typically limited to the size of approximately 30 cm $\times$ 300 cm, where 30 cm is a natural constraint by the machine dimensions and 300 cm is a common artificial constraint of the plotter software. Some techniques such as those used for *Paul the Robot* [214] require process feedback of the current drawing state. Because the processed medium is constantly moved in a hobby plotter, it is not easily possible to track this process with a camera and to use this information in a feedback loop. Therefore, all commands for the plotter must be precomputed.

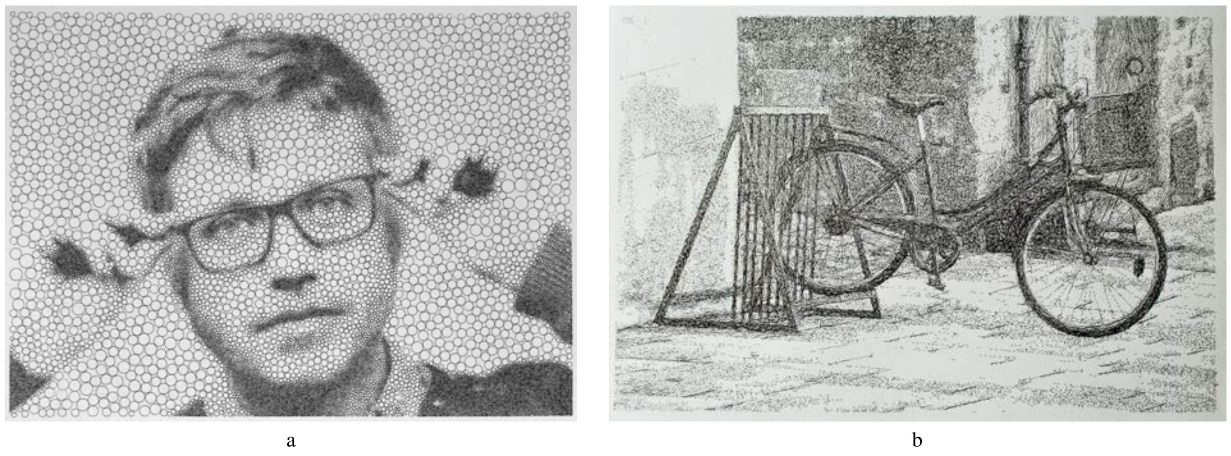


Figure D.7: Examples of a design combination. In (a) the image is encoded through the position and the size of circles, in (b) the image is encoded through position, orientation, and length of straight lines.

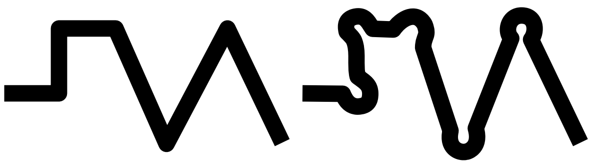


Figure D.8: Connected lines are processed differently by a plotter depending on the tool used. While the pen preserves the shape of the input lines (left), the blade tool rounds up the edges to ensure that the blade can rotate smoothly (right).

Naturally, a XY-plotter processes its input as a collection of paths. Many hobby plotters allow for two different processing tools, a pen and a small freely rotating blade. While the path is unchanged for the pen, the transitions between individual line segments are rounded for the blade as illustrated in Figure D.8. This ensures that the blade has enough space to rotate appropriately. Many methods from computational aesthetics, such as hatching, employ varying line thickness to emphasize darker regions. Such techniques are not possible with a plotter as a pen can only create a line with constant width. Even sophisticated pens with line thickness varying under pressure would not allow for varying lines as the pressure can not be changed during the drawing process. Furthermore, a hobby plotter is not able to draw filled forms. Instead, a collection of narrowly spaced lines has to be created in order to approximate filled regions. As a hobby plotter can only use one or in some cases two pens, the plotter is not able to use a palette of colors and only monochrome images are possible. The software used by conventional plotters is targeted at designs with limited complexity. While this is not a fundamental restriction of the hardware, for practical reasons our system can optimize the appearance of the drawings with an upper limit on the complexity. In Table D.1, we compare existing approaches against the plotter restrictions and our requirements. In addition to the restrictions in the table, the high price and limited availability of the hardware of existing custom solutions are the primary reasons for targeting conventional home-use plotters.

Rel. work	Visual feedback	Complexity	Line width	Similarity opt.	Scalability
E-David	Optional	No constrain as each stroke is computed separately	Varying width possible	Via visual feedback after a set of strokes	Limited
Paul the Robot	Required	No constrain the pen movement is updated on the fly	Constant line width	No tonal optimization	Limited
Conventional plotter solutions	Optional	No constrain due to successive feed	Varying width possible	Varies with solution	Limited

Table D.1: A brief comparison of existing algorithm requirements and plotter restrictions.

D.3.3 Optimization

The goal of our system is the generation of line art or cutouts from any image input and we provide a flexible plugin-based architecture that allows for the combination of different types of positional and shape encodings. However, finding good parameters for these encodings can be a challenging task that typically results in a trial-and-error approach. While greedy algorithms for tonal reproduction can generally achieve good results for the generation of drawings based on a given set of fine-tuned input parameters, their results are heavily affected by these inputs (e.g., thresholds or parameter ranges). This task becomes even harder when the artistic style has to scale with the medium, as the parameters do not necessarily scale linearly with the medium size. Often, artists create drawings with an intended viewing distance, and following this idea we also want to be able to globally indicate a desired distance instead of individually controlling the various parameters of the individual encodings. We therefore set up our approach as a meta-optimization procedure, where the process of creating an individual drawing is steered by a second optimization loop which seeks to optimize its parameters.

Our general optimization process consists of a greedy algorithm that iteratively samples the parameter space, using Latin Hypercube sampling, and simulates the drawing for each parameter setting. In each iteration, the optimizer finds the parameter setting that generate a drawing closest to the input image and then refines the sampling in a local neighborhood reduced by the common factor of $\sqrt{2}$ around the current best sample. Per default, we stop the refinement after $N = 8$ iterations or if

$$\frac{\delta_{n+1}}{\delta_n} > 0.99, \tag{D.1}$$

where δ_n is the minimal difference between the simulated image and the input image in the n-th iteration. This means that we stop the optimization either after the search space was reduced to ca. 6% of the initial parameter space or the optimization improves by less than 1%. While these values can of course be adjusted, we found that they provide robust results and all the images in this paper were generated with these settings. We compute the differences between the simulated drawing and the input image by rasterizing both and computing the normalized mean square difference between them. We choose this rather simple metric because it less sensitive to the structural differences between the input image and the generated primitives. Furthermore, our system does not create the typical failure cases for the mean square error metric, such as transformed images or color shift. If the means square error falls under a user-specified threshold (we use the value of 0.05 as a default and for all the results presented in this paper) or the maximum number of primitives is reached, the drawing is considered to be finished and the drawing loop ends.

As a drawing can only be an approximation of the input image, it is infeasible to compute the differences directly. Instead, both images are scaled to be of the same size and filtered with a Gaussian filter. By adapting the kernel size of the Gaussian filter, we can optimize the similarity of the drawing and the input image based on the desired viewing distance. The visual acuity of a normally-sighted person is defined through a visual angle of 1 arc minute. This means that a person with normal sight would be able to identify an isolated object of the size of approximately 1 mm from a distance of 3 m. However, this is the smallest object an average human could recognize with the naked eye. A study performed by Crété-Roffet et al. [51] indicates that humans hardly perceive blur generated with a Gaussian kernel with the size of 1 mm from a 1 m distance. Using this information, we can optimize the similarity of the drawings dependent on the distance by using an appropriate kernel size. More precisely, we can directly compute the kernel size k of a Gaussian filter to optimize the visual similarity from a fixed distance with the following formula:

$$k = ceil\left(\frac{h_I(px)}{h_P(mm)} \cdot d_{P,V}(m)\right), \tag{D.2}$$

where h_I is the height of the scaled input image and simulated image in pixels, h_P is the height of the physical drawing in mm and $d_{P,V}$ is the distance between the viewer and the physical drawing in m. Using this formula, we compute the appropriate kernels for drawings of size A4 and A3, for an image scaled to 1500 px height during the computation, viewed from 1 m distance as 5 and 7, respectively. In Figure D.9, we compare the results of our simulation for a drawing of size A3 optimized for the viewing distance of 1, 2, and 4 m, respectively. In Figure D.10, we show closeups of the lower-right corner of these drawings. One can notice that high frequency details become less and less preserved as the distance increases.

D.3.4 Drawing Simulation

The process of simulating an individual drawing for a given set of parameters is realized as a pipeline consisting of two stages. Our approach first generates a set of primitive positions, which are subsequently assembled into shapes, based on the configuration of position and shape encoding modules. As already discussed in the previous section, each plugin module has a set of parameters that typically represent the range of properties. For instance, the shape module that generates circles exposes the minimum and maximum radius of a circle and internally maps the image intensity to a value within this range.

Position Generation: For the generation of primitive positions, our system offers two basic strategies: they can either be precomputed or iteratively generated. For *precomputed positions*, the user can either choose from a set of predefined functions or define a new function that covers the canvas with a set of points or curves defining the

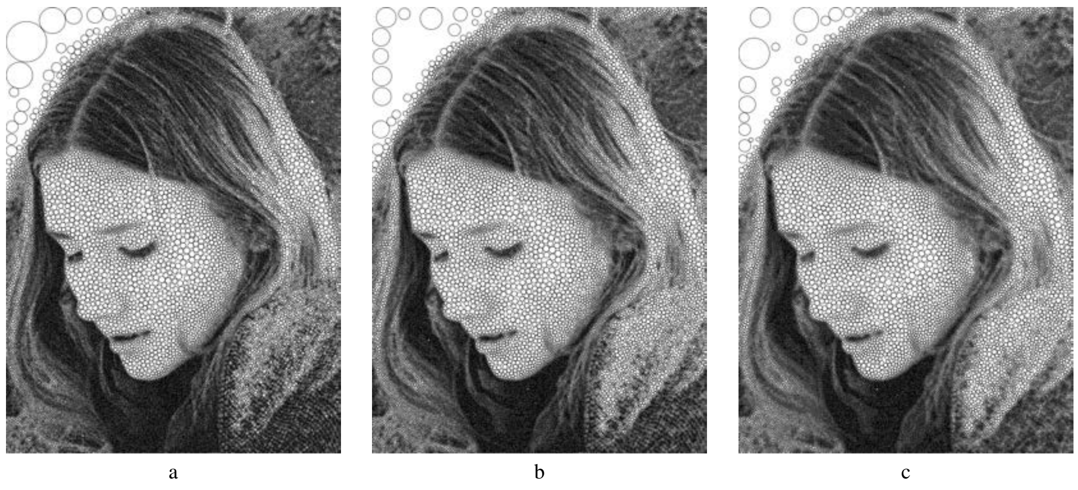


Figure D.9: Results of drawing simulations optimized for a sheet of A3 format viewed from a distance of (a) 1 m, (b) 2 m, and (c) 4 m.

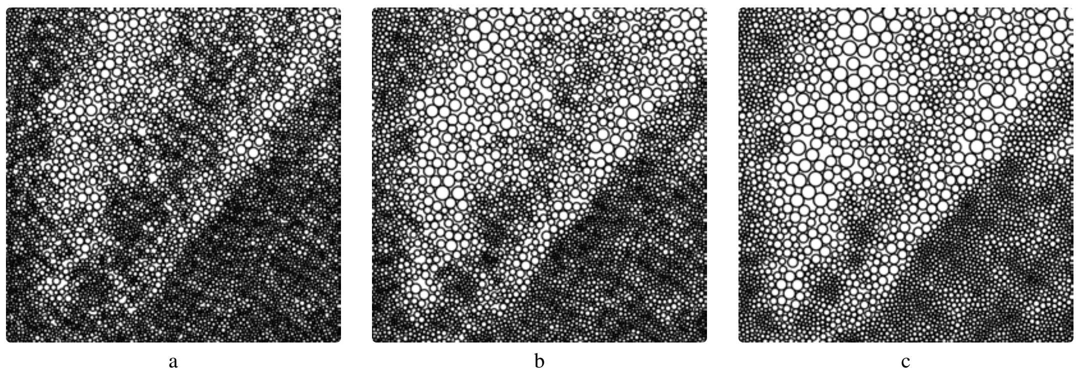


Figure D.10: Close-up on an image depicting drawing of a fabric. The drawing was optimized for a viewing distance of (a) 1 m, (b) 2 m, and (c) 4 m. One can see that the high frequency changes disappear with the distance.

backbones of the primitives. A positional encoding module can either output a set of positions directly, or generate curves that are then automatically sampled by our system. The final point positions are used in the shape generation either as center points of the primitives or in a user-defined way. For instance, a set of points on a regular grid used as center of the primitive would represent a typical halftone image, while as a set of lines can be used for a cutout as in Figure D.15(b). Our system evaluates the backbones in terms of intersection and aborts the computation if the backbones of the primitives already intersect each other. This ensures paper stability for cutouts and prevents overlapping elements during the drawing process, which lead to dark spots in the drawing. In the case of paper cutouts, our system additionally determines how concave the backbone of the visual primitive is. A highly concave primitive can lead to unstable results that can not be hanged freely. To measure the concavity, we compute the maximum distance between the primitive's backbone and its convex hull. If the system detects elements that exceed the degree of concavity that can be handled by the output device, they are discarded to prevent the generation of mechanically unstable results.

Many artistic techniques are best modeled with an *iterative computation of primi-*

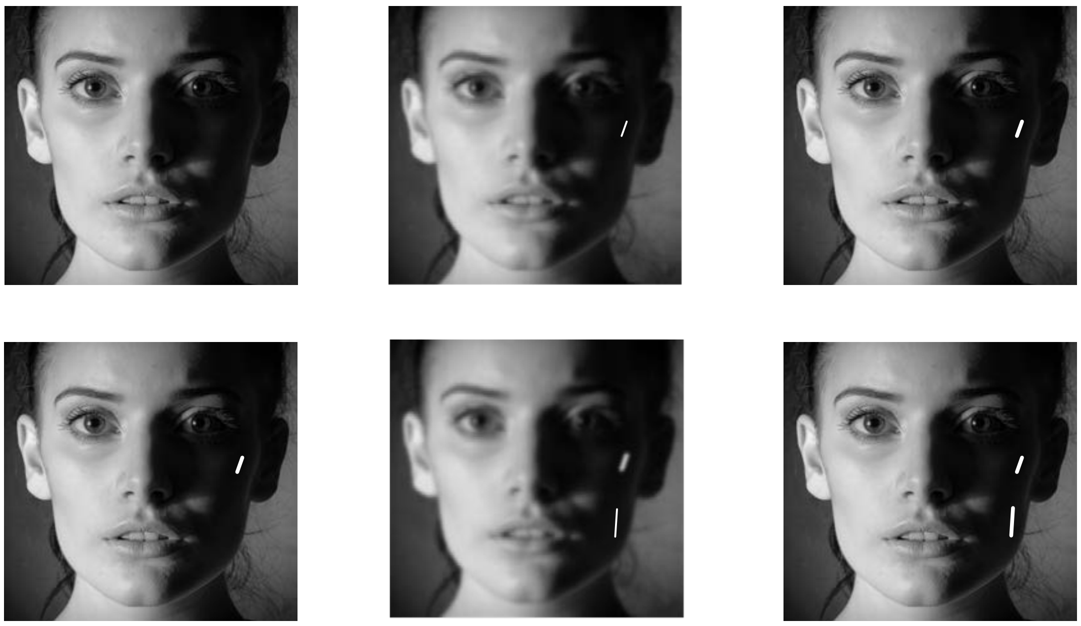


Figure D.11: An example of incremental position encoding. The system creates a copy of the input image (top left). The copy is blurred, a primitive is computed at the darkest pixel of the blurred image (top middle). A bigger primitive with the same orientation is added to the initial copy (top right). This process is iterated (lower row).

tive positions. This design is not only a natural choice, but also allows us to effectively simulate the drawing process on the computer and thus to optimize the drawing parameters. To allow for efficient operation while providing a simple interface, we utilize a simple strategy for modules that use iterative position computation, as illustrated in Figure D.11. Essentially, we create a copy of the input image that functions as canvas. In this copy, the system finds the position of the darkest pixel, computes a primitive at that position, and adds the primitive in white color to the input image copy. To account for the perceptual error, the image copy is blurred with the same Gaussian kernel as used for the outer optimization loop. Each computed primitive is evaluated for constraint violation, such as intersections for the paper cutouts. If the primitive violates any constraints, it is discarded and not added to the drawing simulation. However, the center position of the primitive is still marked in the image copy, to ensure that this position is not selected again.

Shape Generation: The shape of the primitives can either be encoded through a set of predefined functions or via a user-defined function. Typical encodings consist of a shape variation around the generated backbones, such as variation of circle size or amplitude variation, that directly encodes the image intensity, but it is equally possible to use derivative information, e.g., to steer the direction of strokes. In Figure D.12, we show a typical shape encoding for a cutout style. The cutout is represented by a polygon that is symmetric along the backbone axis with the width encoding the gray values. Our system automatically samples along the primitive backbones to compute the shape, following the restriction on maximum number of elements (if such as restriction has been specified). In the case of cutouts, the system also computes the distance between the visual primitives – if the distance falls below 1.4mm, the result is discarded.

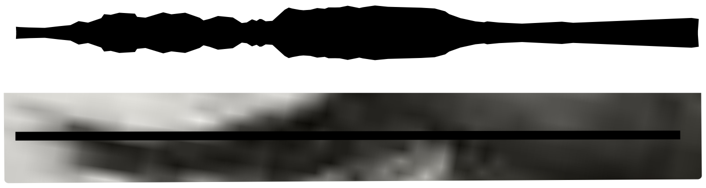


Figure D.12: The shape of the polygon encodes the darkness values along its centerline as thickness.

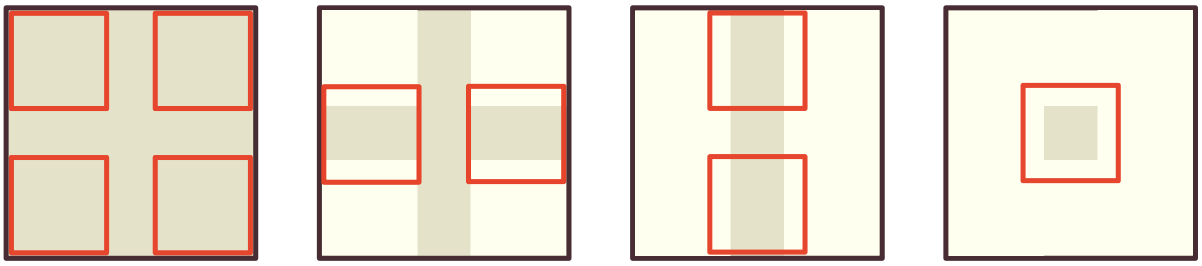


Figure D.13: We parallelize the primitive generation by subdividing the image into a set of tiles, where the primitives are computed.

D.4 Implementation and Performance

Our system was implemented in Python 3.6 using CUDA, OpenCV and DXFWrite. CUDA was used to parallelize the drawing simulation, OpenCV is a library mainly aimed at real-time computer vision that was used for efficient image operations, and DXFWrite was used to export the drawing paths as dxf files that can be read by conventional plotter software. We minimize the computation time by computing the primitives in parallel on the GPU. For shape encodings where the positions of all primitives are fixed, we compute the shape of all primitives in parallel. Hence, the computation time for purely shape-encoded styles lies within a few milliseconds. To reduce the synchronization overhead required for the parallelization of positional encodings, we subdivide the image into overlapping tiles (Figure D.13) which are then computed in parallel. This process is performed in the following manner: First, we subdivide the image into tiles that have at least a pairwise distance of twice the size of the largest primitive. This ensures that no primitives are in conflict (Figure D.13, left). Then the tiles are shifted as shown in the subsequent images in Figure D.13 and the computation is repeated in each step.

The subdivision into tiles not only allows for parallel computation, but also results

Style	Figure	Comp. time	Prod. time
Cutout	D.15(b)	0.09s	56min
Dashes	D.16	1.70s	92min
Spiral	D.5(b)	0.01s	47min
Stippling	D.4(a)	0.67s	96min
Circles	D.19(b)	4.76s	87min
Triangulation	D.4(b)	0.72s	84min

Table D.2: A brief comparison of existing algorithm requirements and the plotter restrictions

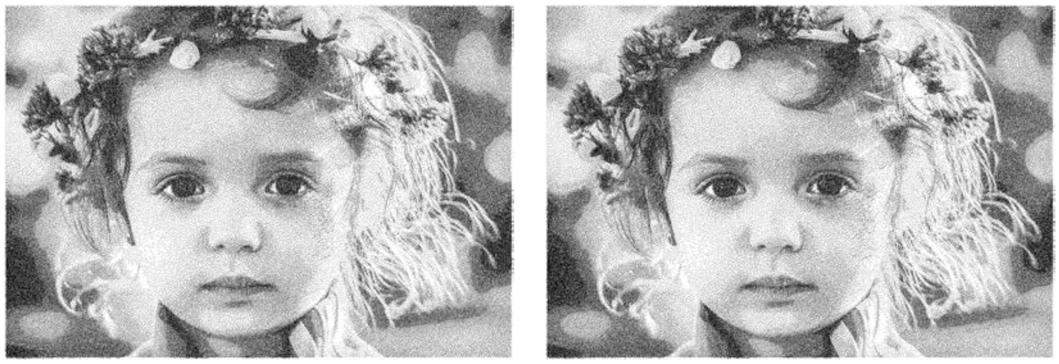


Figure D.14: A comparison of a drawing computed as whole (a) and with tiles (b). Illustration (a) took 741 seconds to compute while (b) took 1.43 seconds to compute. Qualitatively the results are almost undistinguished.

in an overall speedup as only a limited area has to be evaluated. This simple strategy leads to a performance increase by a factor of approximately 380 compared to the naive approach. In our experiments, only the circle and the single line style exceeded computation time of 2 seconds, due to their serial structure. On average the remaining styles take around 0.5 to 1 second to generate an image. Table D.2 lists the computation times for the results presented in the paper. For completeness, we also include the production times as measured on a Silhouette Studio plotter.

Because the primitive positions are computed by evaluating the image row by row, the primitives tend to accumulate at the right and the lower border of the tiles. This can potentially introduce undesired artifacts. To minimize these artifacts, the computation is performed for a small range of gray values, i.e., we subdivide the computation into n steps and in the k-th step compute the primitives up to $\frac{k}{n}$ image intensity. This leads to much more evenly distributed primitives as the primitives of the neighboring tiles cover parts of the right and lower tiles borders in each iteration. In practice, we found that $n = 5$ provides results with no visible artifacts. In Figure D.14, we show a comparison between an illustration computed as whole (a) and one computed with tile subdivision (b). The illustration in Figure D.14 (a) took 741 seconds to compute while the one in (b) took 1.43 seconds. The resulting images are indistinguishable to the naked eye.

D.5 Results

To generate a drawing, the user has to provide the function names of the positional and shape encoding functions and two lists of the function parameters and their range (if the upper and lower bound of the range are the same, then the parameter is not considered in the optimization). This information is then sent to the optimizer function, which samples the parameters and calls the drawing simulation function for each parameter setup. The drawing simulation function calls the positional encoding and shape encoding functions. After each primitive computation LinesLab, evaluates the result for constraint violation. In the following we present several pieces generated with our system. The pseudocode for the used encodings can be found in the Appendix.

Paper Cutouts: The Korean artist Yoo Hyun[16] creates breathtaking portraits of

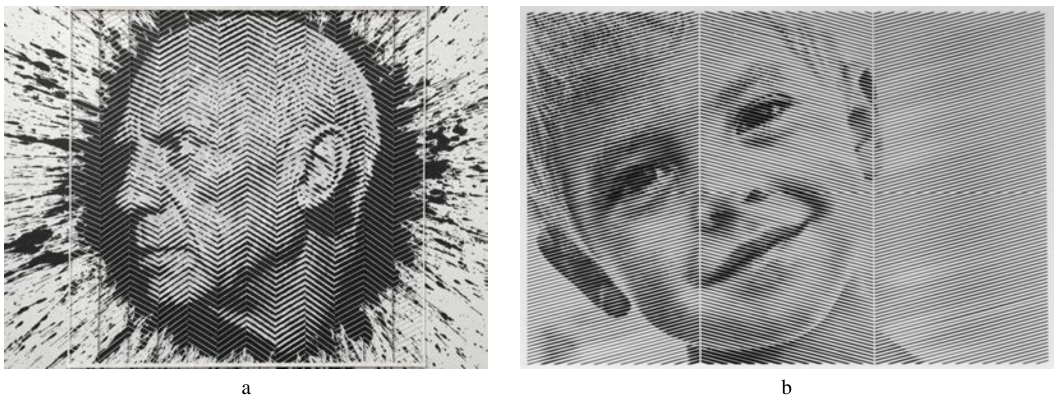


Figure D.15: (a) Cutout portrait by Yoo Hyun. (b) Cutout generated with our system.

celebrities by carefully slicing away thin slivers of paper. In this way, Hyun forms a woven zigzag pattern contained inside a solid frame as shown in Figure D.15(a). With our system, we are able to recreate the same style by choosing a similar positional encoding as Huyn, a regular tiled grid, and a shape encoding that translates the darkness values into line thickness. The pseudocode of the encodings is shown in Algorithms 1 and 2. The positional encoding essentially creates a herringbone pattern. LinesLab automatically optimizes the encoding parameters to create a result with the highest tonal similarity to the input image while ensuring paper stability by restricting the distance between the cutouts to at least 1.6 mm. The result of our cutout is shown in Figure D.15(b). In contrast to Yoo Huyn, our system leaves small strips of paper between the segments to ensure the stability of the paper while cutting. Hyun spends tens of hours creating such portraits. Using our system, we are able to create the cutout in just over 45 minutes (including the production time).

Dashes Style: Inspired by Van Gogh, we created an image composed of short dashes in the direction of lowest variance. In this style, the positional encoding is performed iteratively as described in Section D.3.4 and can be formulated as shown in Algorithm 3. The shape encoding is done by sampling a line at each generated position and choosing the direction of the sample with the value variance along the line. We describe the shape computation with the pseudocode shown in Algorithm 4. For this style only the maximum length of the primitives was declared as an optimization parameter and is determined in the outer optimization loop. For instance, for an image of size A4 drawn with a pen of 0.5 mm thickness and a viewing distance of 1 m, the resulting value of the maximal line length is 4 mm. Three results generated in this style are shown in Figure D.16. The respective photographs were taken at the optimized distance for each image. The leftmost drawing has a size of 3.5×5 cm and is optimized for a viewing distance of 20 cm, the middle image is 21×29.7 cm big and is optimized for a viewing distance of 120 cm, and the rightmost image measures 29.7×42 cm and is optimized for a viewing distance of 170 cm.

Circle Style: Many artists use familiar shapes to create complex portraits. Màrton Jancsò, for example, created a portrait composed of only circles, shown in Figure D.19(a). Our system is able to recreate this style with the above mentioned iteratively-computed positional encoding in Algorithm 3 with circles as primitives. The marked circles are computed using the image brightness and a minimum and maximum ra-

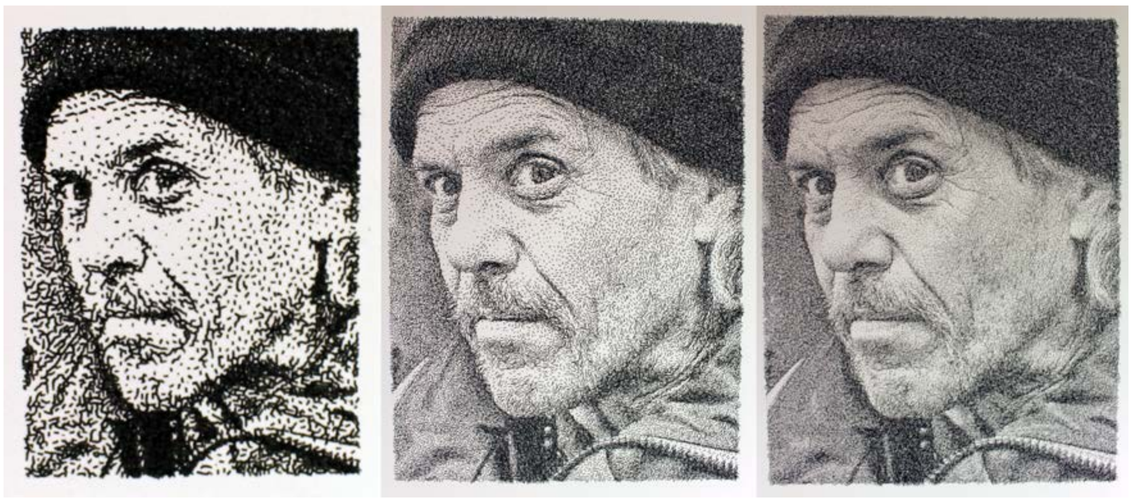


Figure D.16: A photograph of three drawings with the sizes 35×50 mm, 210×297 mm and 297×420 mm. The photograph is a blend of three photographs to bring each drawing in focus.

dius as input parameters. To draw the circles in a space filling manner, we compute a Voronoi diagram of the primitive positions. For each cell, we find the largest fitting circle. A drawing made by our system is depicted in Figure D.19(b). As can be seen, our approach tends to leave blank spaces between the circles in very bright areas and at areas close to the border. This is due to the elongated shape of the Voronoi cells that occurs more often at the image border.

Crosshatching Style: Creating drawings with crosshatching techniques originated in the middle ages and was perfected by Albrecht Dürer. It is still a popular technique for artists in modern days. The main concept of crosshatching is that the quantity, thickness, and spacing of lines depicts the brightness of the image. As mentioned in Section D.3.2 we cannot change the line thickness during the drawing process. Therefore, to recreate a convincing crosshatching style, only the line quantity and spacing can be adjusted in LinesLab. A crosshatching drawing made using our system can be seen in Figure D.17(a). The illustration consist only of straight lines, and as such, only the positional encoding is responsible for the drawing result. The pseudocode for the positional encoding is shown in Algorithm 5. The system first computes the prominent contours in the image with a Canny edge detection algorithm. Next, the image is quantized to five intensity levels. For each of the levels, the algorithm then fills in the corresponding areas with crosshatching lines. To create a natural appearance, we add noise to the start- and endpoints of each line. The system then optimizes the length of the lines, the distance between the lines for each tone, and the noise level. Our system can be directly applied to other media as well. In Figure D.17(b) we show the same illustration engraved with a laser cutter on a Medium-density fiberboard.

Single Line Style: Artwork created with a single line ranges from abstract minimalist drawings to highly detailed portraits. We can realize a single line style in our system by using the same positional encoding as for the stippling style (Algorithm 3). Then, in the shape encoding, we choose a random point and declare this point as the line ending. We compute the five nearest points to the endpoint and randomly choose one as the next point. This process is repeated until all points have been processed and the line can be drawn. We show a result of this style in Figure D.18. The only optimized

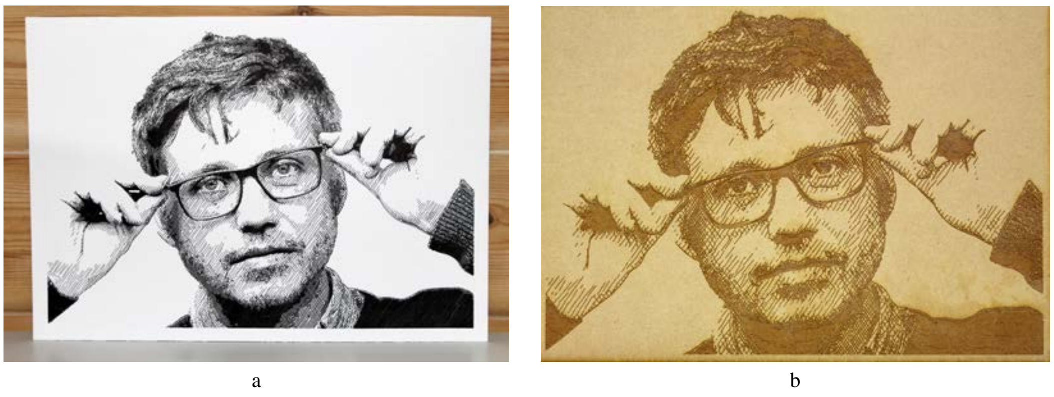


Figure D.17: Crosshatching drawing made with the LinesLab system. (a) The illustration is drawn on paper (b) The same illustration etched on MDF with a laser cutter

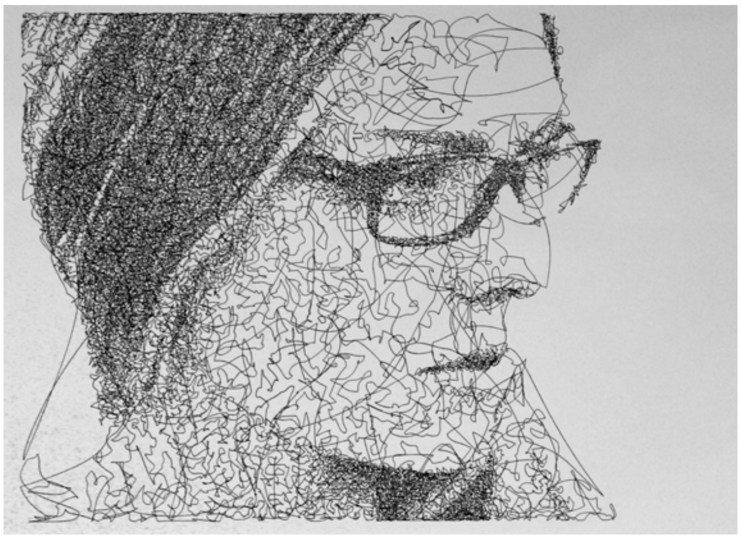


Figure D.18: Single line drawing, created by connecting the positions of a stippled image with a single line.

parameter for this style is the size of the marker in the positional encoding. We increase the performance of finding the closest primitives by considering only primitives in the adjacent cells. However, due to the iterative structure this style is rather slow to compute.

Mosaic Style: A popular task for art students is to create portraits on a regular grid. The portrait by Katherine Cahoon in Figure D.20(a) is an example of this style. With our method, we are able to create such a grid portrait by simply subdividing the input image into regular tiles and processing the tiles individually with a random style. The result of this approach can be seen in Figure D.20.

D.6 User Feedback

To gain feedback on the usability and utility of our approach, we presented LinesLab to two computational artists with different levels of programming experience. Both

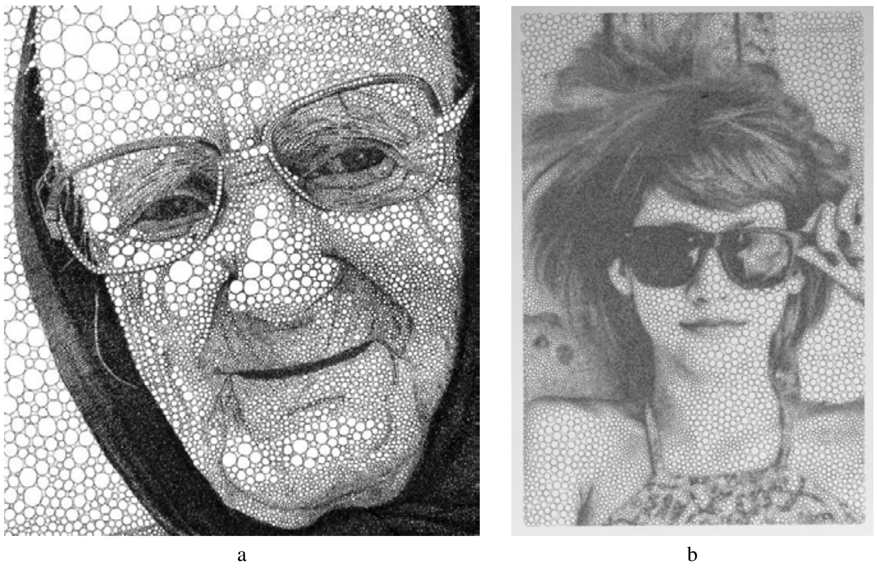


Figure D.19: (a) Circles drawing by Màrton Jancsò (b) drawing generated by our system.

artists use Processing to create their artworks, but the first artist has used Processing for several years, while the second user has only experimented with it for three months. We explained LinesLab to them and let them freely explore the system with the possibility to ask questions at any point. After the exploration phase, we asked both artists to add a new style to the system. Finally, we asked them to comment on LinesLab and its usability.

Experience with programming languages was benefitial for the first artist in terms of required learning time and in implementing new encodings. The first artist spent 34 minutes in the exploration phase, while the second artist needed 78 minutes to fully understand the entire functionality of the system. During the exploration phase, both artists simulated different styles by exchanging the positional and shape encodings, while investigating the code of the used encodings. Both artists were able to create a new style in our system. However, the first artist was able to implement a new style within just under 20 minutes, while the artist with less programming experience needed 37 minutes. We show the results of the two styles in Figure D.21. In Figure D.21(a), the artist added a new positional encoding in form of lines aligned in a hexagonal pattern, which were superimposed with the existing amplitude modulation function. LinesLab automatically optimizes the spacing inside the hexagon as well as the sampling rate and the amplitude of the modulation function. In Figure D.21(b), the artist used an existing positional encoding (regular grid), and then created a star pattern for each primitive. For this style, our system optimizes the spacing between the primitives and the length of the drawn line in the star pattern.

After the hands-on phase, we asked the two artists to comment on the usability of our system, features they liked and disliked, and if they would consider using our system in the future. With respect to usability, both artist said that the functionality of the system can be understood fairly easily through experimentation with the different encoding functions. When asked to comment on features they liked, the first artist replied: "I especially like the separation between spatial and positional encodings. This really makes it easy for me to experiment with different style combinations. Furthermore, it

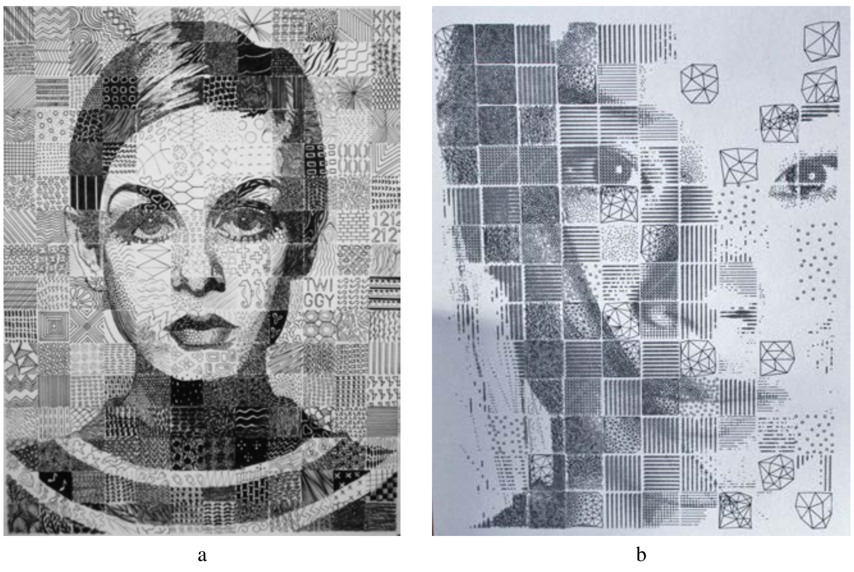


Figure D.20: (a) Grid portrait by Katherine Cahoon (b) Grid Portrait generated with our system.

forces me to mentally decompose my ideas for a style, which makes it easier to translate them into code later on." The second artist noted: "The automatic parameter tuning is really helpful. I would still change the parameters a bit to get the result I had in mind, but the automatic tuning drastically reduces the testing time." When asked about any annoyances or features of dislike the second artist replied: "It confused me that some styles were computed almost instantly while some took a bit longer. Maybe having a progress bar would be good.". Both artists mentioned that a drag and drop interface for the encodings would make the exploration of the system easier. Nonetheless, both artists said they would appreciate to use our tool in the future. The first artist said: "I would absolutely use this tool, but mostly to create a basis on which I can work in other programs such as Illustrator or Inkscape.". The second artist replied: "Yes, I would use this tool. I think it would save me a lot of time. For me it is much faster to create good results in this tool than in Processing, because I can easily reuse old styles and I get to see good results without tuning all the parameters."

D.7 Discussion

The creation of physical drawings or cutouts can have unexpected and unwanted artifacts that we normally do not encounter when only dealing with digital images. For instance, the pen may not distribute the ink evenly resulting in too light or too dark areas. Similarly, a blade can wear off and tear the paper instead of producing a clean cut. In our experience, such issues can be minimized by reducing the speed of the plotter, allowing for a more even distribution of ink and lower the forces on the paper. Another way to avoid excessive ink leakage of a pen is to use strongly absorbing paper, such as Aquarelle paper.

To ensure paper stability, we simulated the stress of a paper sheet during the cutting process and the deformation of the paper after the cutting and incorporated the results

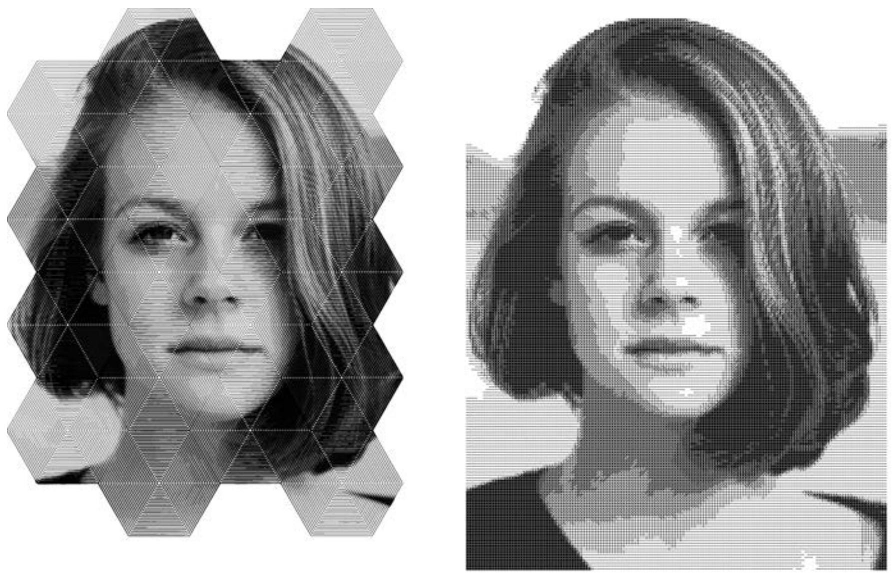


Figure D.21: (a) A new hexagonal style created by an artist with long strong programming skills (b) A halftone style create by an artist with limited programming experience.

as global parameters of our system. The simulation was performed in ANSYS, where the paper was approximated as a thin sheet of custom material based on wood with its tensile strength and density changed to the values of paper. The result of the simulation was that the stress during cutting is negligible, if the cutting paths have at least 1.4 mm distance between them.

Furthermore simulated the deformation of hanging paper cutouts fixed on the upper corners. The simulation showed that the deformation is minimized when the cutting pattern follows vertical lines or lies within an angle of 14 and 46 degrees. While such a simulation can only provide guidelines for the cutting process, we still see such results as valuable for the design of our system and hope to incorporate further results in the future. In fact, we could confirm that the paper often tore when the distance between cutting paths fell below 1.4 mm.

At present, we use a rather simple optimization process of refining around the best result of the current iteration. While this approach has proven to be effective, more complex optimizers could be easily integrated, as our system is completely modular in this respect. While we deliberately targeted off-the-shelf hobbyist plotting and cutting devices in the development of our system, there are no fundamental restrictions that would prevent its use for professional devices. In fact, we tested our system on industrial laser cutters, as can be seen in Figure D.17. In the future, we plan to experiment with professional vinyl cutters that can be used for large scale drawings with only moderate expenses. The current implementation of LineLab provides a function-based interface In the future, we plan to extend the usability by providing a simple drag and drop functionality to further simplify experimentation.

Despite its modularity and the simplicity of our optimization process, our system is still able to closely recreate the results of specialized approaches. For instance, we were able to create similar results to Chiu et al.[45], by computing the positions as stippling points first and constructing a backbone path through them with a TSP solver.

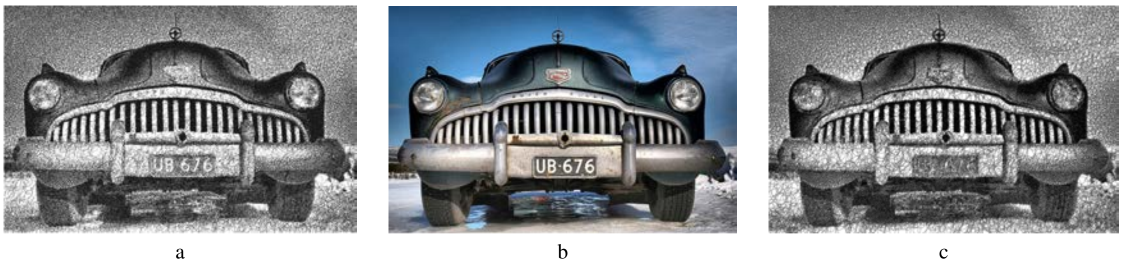


Figure D.22: (a) Scribble art generated by a method proposed by Chiu et al., (b) Input image, (c) Scribble art generated with our system.

The shape encoding was done through a trochoid function, which is a curve defined by a fixed point on a rotating circle traveling along a line. The optimized parameters were the mapping parameters that computed the rotation speed and circle radius from the local image intensity. The comparison of our result and the result by Chiu et al. can be seen in Figure D.22. Clearly, our formulation can not exactly reproduce the tonal quality of a highly specialized approach. However, this example shows the strength of our system in being able to produce high quality results with a much simpler setup.

D.8 Conclusion

We presented a novel system capable of creating line art and cutouts with conventional hobby plotters and cutting machines. Our approach is flexible and modular, supporting a wide range of artistic styles generated from a basic set of position and shape encodings. By employing a nested optimization process, we are able to automatically generate artistic results that conform to the physical and mechanical constraints of the output devices. To the best of our knowledge, our system is the first one that optimizes the generated drawings based on the distance of the viewer to the artwork itself, while maintaining scalable with respect to the medium size. We achieve results comparable to existing works of art that can be easily reproduced with off-the-shelf hardware. Furthermore, our architecture features a simple plugin mechanism, which allows the easy addition of new styles.

Appendix

In the following, we give the pseudocode for the positional and shape encodings used throughout the paper.

```
Cutout positional encoding
input : number of divisions: nx, number of cut lines per division: ny, angle of the
        cutout: $\alpha$
for cx in range (0, nx) do
    for cy in range (0, ny) do
        compute the start and endpoints pS, pE for alternating primitive backbones
    end
    primitives.add([pS, pE])
end
return primitives
```

Algorithm 1: Positional encoding for the cutout style.

```
Cutout shape encoding
input : thickness of the cutout: th, sampling distance: d, primitive backbone
        containing pS and pE: p
compute primitive length pL
ns = pL/d
for s in range (0, ns) do
    pos = pE * s/ns + pS * (ns-s)/ns
    br = (1-img(pos))
    primitiveThickness = br * th
end
```

Algorithm 2: Shape encoding for the cutout style.

```
Dashes positional encoding
input : input image with all existing primitives marked white: img
p = minLoc(img)
mark primitive in img
return p
```

Algorithm 3: Pseudocode of positional encoding for the dashes style.

```
Dashes shape encoding
input : input image: img, primitive pos: p, maximal primitve length: mL, number of
        radial samples: ns
lowVar = +inf
for s in range (0, ns) do
    a = s * pi/ns
    pS = [p.x + mL/2 * sin(a),p.y + mL/2 * cos(a)]
    pE = [p.x - mL/2 * sin(a),p.y - mL/2 * cos(a)]
    var = measure variance in img from pS to pE
    if var <lowVar then
        lowVar = var
        aa = a
    end
end
pL = (1-img[p]) * mL
pS = [p.x + pL/2 * sin(aa),p.y + pL/2 * cos(aa)]
pE = [p.x - pL/2 * sin(aa),p.y - pL/2 * cos(aa)]
return (pS,pE)
```

Algorithm 4: Shape encoding for the dashes style. For each primitive the system find the direction of the lowest variance around that point and draws a line in that direction encoding the image brightness with the line length.

```
Crosshatching shape encoding
input : input image: img, line angle: a, distance between lines: d, maximal line
        length: l
edges = findCannyEdges(img)
postImages = posterize(img,5) // returns 5 binary images encoding the posterized are
  as 1 and background as 0
canStartLine = True
canEndLine = False
for pImg in postImages: do
    for x in range (0,img.w) do
        for y in range (0,img.h) do
            if canStartLine and pImg == 1 then
                begin line
                canStartLine = False
                canEndLine = True
            end
            if canEndLine and (pImg == 1 or lineLenght > l) then
                end line
                draw line
                canStartLine = True
                canEndLine = False
            end
        end
    end
end
```

Algorithm 5: Positional Encoding for the crosshatching style

D

Paper E

Firefly: Virtual Illumination Drones for Interactive Visualization

Sergej Stoppel, Magnus Paulson Erga and Stefan Bruckner

University of Bergen, Norway

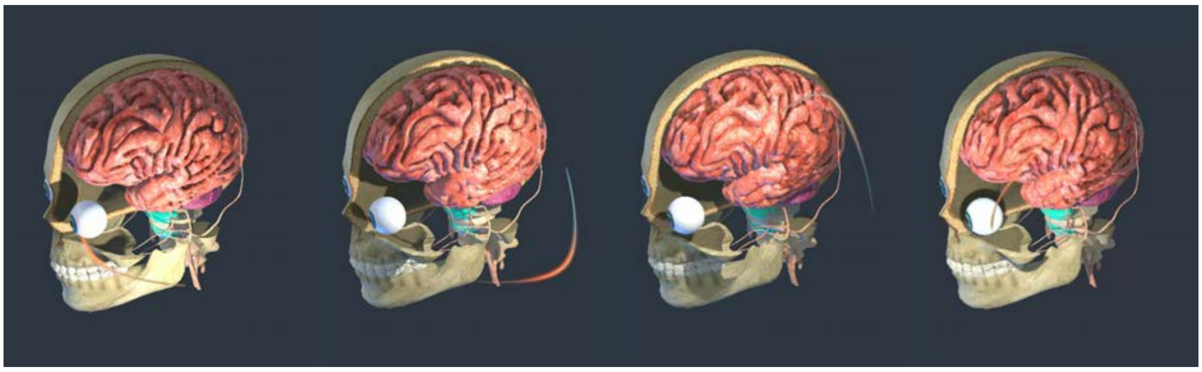


Figure E.1: Light specification in three dimensional scenes is a complex problem and several approaches have been presented that aim to automate this process. However, there are many scenarios where a static light setup is insufficient, as the scene content and camera position may change. Simultaneous manual control over the camera and light position imposes a high cognitive load on the user. To address this challenge, we introduce a novel approach for automatic scene illumination with Fireflies. Fireflies are intelligent virtual light drones that illuminate the scene by traveling on a closed path. The Firefly path automatically adapts to changes in the scene based on an outcome-oriented energy function. To achieve interactive performance, we employ a parallel rendering pipeline for the light path evaluations. We provide a catalog of energy functions for various application scenarios and discuss the applicability of our method on several examples.

Abstract

Automatic light placement in three dimensional scenes is a very complex and highly researched problem. While a dynamic light can increase the structural comprehension,

This article was accepted for IEEE VIS 2018

the automatic generation of dynamic light paths is not sufficiently explored. In this paper, we introduce a novel approach for dynamic scene illumination with Fireflies. Fireflies are intelligent light drones that illuminate the scene by traveling on a closed path. We optimize the Firefly path on the fly based on various criteria, such as increase of shape perception or a balanced overall illumination.

E.1 Introduction

Illumination has a crucial impact on the appearance of 3D objects and shape perception in computer-generated scenes. Once the geometry, textures, and material properties of the scene have been defined, its appearance is greatly affected by the illumination setup. Shape perception, for example, is highly dependent on light placement. Uncommon positioning of light sources can distort the perceived geometry as it is known from crater or dome illusions. The traditional approach for lighting specification is an iterative process of trial and error, where the user continuously adjusts the light parameters and evaluates the rendered image. This makes lighting design a challenging task even for static lights.

Moreover, in interactive scenarios static lights alone may not be sufficient. When asked to visually assess the geometry of objects, observers most commonly rotate them back and forth. Studies show that the motion of an object relative to the light source helps to evaluate its geometry and material properties [54, 208]. When the object of interest cannot be moved easily, observers tend to obtain geometry and material cues by moving the light source. It has been shown that participants have a clear intuition of how the light source distance and position affect the shading patterns for a variety of different surfaces [184]. This indicates that animated lights could be used to enhance shape perception in 3D scenes.

Modern animation approaches use moving light sources to emphasize temporal changes in the scene or to adapt the light conditions to a dynamic setup. While static lighting specification is already a challenging task with some existing solutions for automated support, the simultaneous control of the camera and moving light sources represents an even greater hurdle for most users. Furthermore, when considering the rapid advances in immersive virtual reality technology, which limit the available degrees of freedom for interacting with parameters like light source configurations, the requirement for "intelligent" light sources that automatically adjust to the movement of the camera becomes more and more pressing.

To address this challenge, we propose a novel automated approach for scene illumination with dynamic light sources that offers additional perceptual cues compared to static lights. In addition to the static light sources in the scene, we generate a Firefly – a moving light drone that illuminates the scene while flying on an automatically generated path. Our system continuously optimizes the Firefly path based on a flexible energy function, tailored for various visualization tasks. We propose several energy function constructions for different visualization scenarios, that are designed to enhance different aspects of the rendered objects, while following established lighting design rules from photography and art.

E.2 Related Work

Automation and optimization techniques have a long tradition in the field of visualization. A common problem is the choice of suitable parameters for a particular visualization technique and/or dataset. In volume rendering, for instance, the specification of transfer functions is a challenging task. While presets can provide some additional support to the user, the presence of additional variables such as differences in the data acquisition necessitates more advanced approaches. Examples include the work of Ruiz et al. [179], who presented a framework to define transfer functions based on a target distribution provided by the user. Similarly, Borga et al. [34] proposed an optimization based algorithm that shifts preset transfer functions, to account for general deviations and local variations in the data. To deal with occlusion in flow visualization, Günther et al. [79] translate the occlusion problem into a view-dependent global optimization problem that is solved with the least squares method. Modern visualization solutions for 3D scenes often employ large and complex scene geometries. Early on it was found that the increased scene complexity hindered the orientation of the user and impaired the ability to navigate effectively. Freitag et al. [71] automatically adjusted the camera speed based on viewpoint quality to reduce the cognitive effort for camera control in indoor scenes. Xie et al. [238] proposed an automatic camera path planing method, that aims to improve the user's sense of direction in VR setups. The roles of static, animated, and interactive presentations of 3D scenes have been investigated by Froes et al. [72]. Coffey et al. [46] extended the study to virtual reality. Our approach can be seen as a natural development of expanding animations to light setups.

As our approach aims to generate virtual drones to automatically support the user in their task, we draw inspiration from physical autonomous vehicles. Already in 1969, Keiser and Peebles [116] discussed a concept for automatic drone control. Nikolos et al. [156] used an evolutionary approach to design a Bezier path for unmanned aerial vehicles. While their setup significantly differs from ours, Srikanth et al. [196] used physical light drones for rim illumination of dynamic objects in indoor photography. Joubert et al.[111] presented an interactive path planing tool for drone cinematography, particularly focusing on the importance of the trajectory smoothness and the spatial awareness of the user.

Illumination has a significant impact on the perception of a scene, but can necessitate a tedious trial and error process in order to arrive at a desired result. Cost et al. [50] discussed an automated approach for lighting design that employed optimization strategies based on object geometry, material properties, and design goals. To support non-experts, Shacked et al. [187] developed a method for fully automatic lighting design based on a perceptual quality metric. Gumhold [78] used entropy to place a light source that maximizes the information added by illumination. Halle et al. [82] presented LightKit, a lighting system for 3D scenes inspired by light designs of artists and photographers. Lee et al. [131] introduced Light Collages, an illumination system that enhances local features using a globally inconsistent lighting setup. Wang et al. [225] proposed a lighting system that enhances visual cues for local and global features. Zhang and Ma [249] extended automatic three-point lighting setup to volume rendering employing global illumination. Recent work by Wambecke et al. [222] introduced a lighting design approach based on photographic rules, taking into account the shapes and materials of the objects. A coherent lighting setup is especially important in augmented reality, as the rendered object must be integrated into an already illuminated scene. Haller et al. [83] used real-time shadow maps to add realism to the augmented scene. Okumura et al. [160] and Klein et al. [122] incorporated blurring filters on the rendered image to match the depth of field of the captured video stream. Aittala [22] used real-world observations from a diffuse sphere to adjust virtual lighting parameters for augmented reality setups.

To produce an effective lighting setup, it is important to account for various aspects of human perception. Studies by Ramachandran [172], Kleffner and Ramachandran [121], and Mamassian et al. [143] investigated the assumptions made by the human visual system and how they are affected by the light position. Doerschner et al. [54] identified three motion cues the visual system relies on to distinguish between matte and shiny surfaces. Kersten et al. [118] performed a study on information provided by cast shadow motion. Their research suggests that the visual system assumes a stationary light source even if a moving light source is present. However, this may be due to the fact that no visual cues for the light position where presented to the participants. A recent study by Schütt et al. [184], where the participants could control the light position to some degree, suggests that participants do have a clear intuition of how light positions affect the appearance of the illuminated object. Guided by these findings, we inform the user of the Firefly position through visual cues and avoid sudden and unexpected trajectory changes by imposing constraints on the path shape.

In a cinematic context, lighting is often used to convey emotions, and the effects of lighting on the perceived scene atmosphere have been investigated in several studies. An overview of different lighting setups can be found in the book Advanced RenderMan by Apodaca and Gritz [27]. De Melo et al. [52] focused on the emotions induced by different lighting setups and proposes a model for the expression of emotions in virtual humans with a composition of lights, shadows, and chromatic filters. Wisessing et al. [232] investigated how animated characters are perceived when viewed under different lighting conditions. Nasr et al. [60] presented a lighting system that automatically adjusts to accommodate variations of the dramatic scene characteristics. We use the findings of these studies as guidelines in the construction of the Firefly paths.

E.3 Firefly

The goal of our approach is to support users by providing an animated light source – a Firefly – that moves along an adaptive path continuously adjusting its trajectory if necessary. The Firefly complements additional static light sources and in particular aims to enhance dynamic aspects of the exploration process. As the control and planning of a moving light source is even more complex than the design of a static lighting environment, automatic generation and adaptation of the Firefly path is a key component of our system.

Whereas previous approaches for the placement of static lights could partially rely on precomputation, our aim is to provide a fully dynamic solution that adapts interactively to changes in the camera and scene setup, and does not impose any constraints on the content of the scene. A key aspect of our approach is that the FireFly acts in a view-dependent manner, i.e., it aims to adjust its trajectory according to what the user sees. As such, our approach is an online optimization process. We designed Firefly

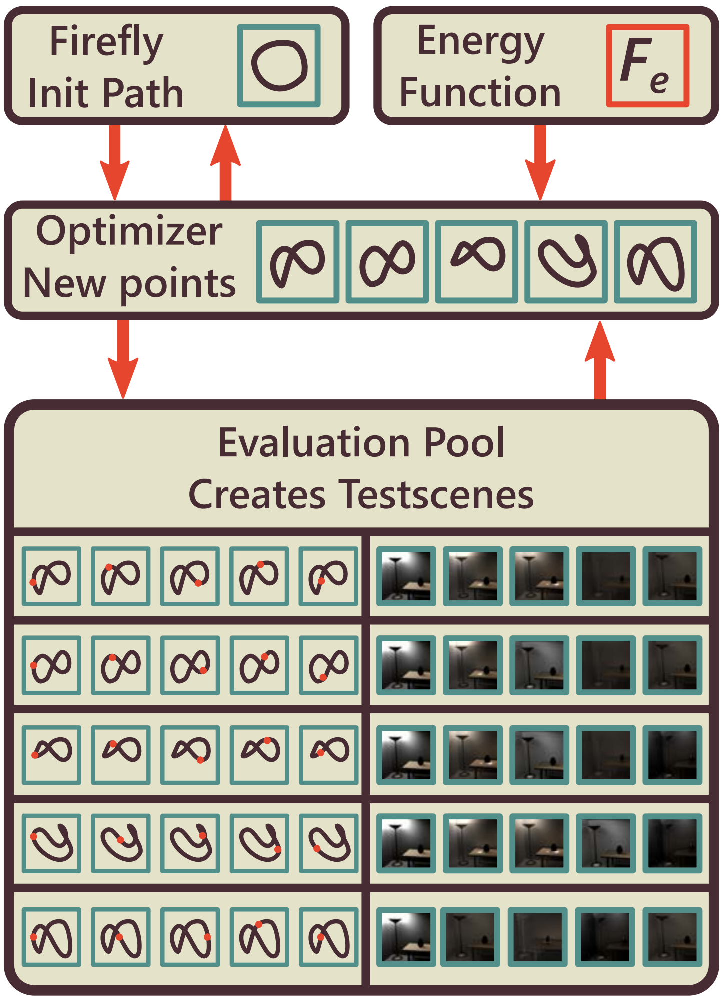


Figure E.2: A conceptual overview of the Firefly system. The Firefly works as a parallel multi-threaded process. The only input required by the user is a selection of the energy function from a provided library. The system iteratively updates the Firefly path by creating a set of new paths and evaluating them in the background (Evaluation Pool). The Firefly transitions to a better path when one is found.

as an independent component that can be easily integrated into existing systems. To achieve a high degree of flexibility for versatile illumination tasks, we created the Firefly system as a plugin-based detached optimization process. We explain the general Firefly system in the remainder of this section, discussing the individual components in detail in the following subsections.

A general overview of our approach can be seen in Figure E.2. The Firefly generation and optimization works as a parallel process to the rendering pipeline. The only input required from the user is the selection of an energy function from a catalog. When a Firefly is triggered during the interaction process, an initial Firefly path is constructed and placed in similar positions as lights in common photography setups. Next, the optimizer suggests a new set of light paths. The light paths are sent to an evaluation pool, which handles every test path in a separate thread. In each thread a new scene with the Firefly path is generated and the path is evaluated according to the energy function. After all paths have been processed, the evaluation pool returns a fitness value for each path to the optimizer. When the optimizer has found a better path than the one that is

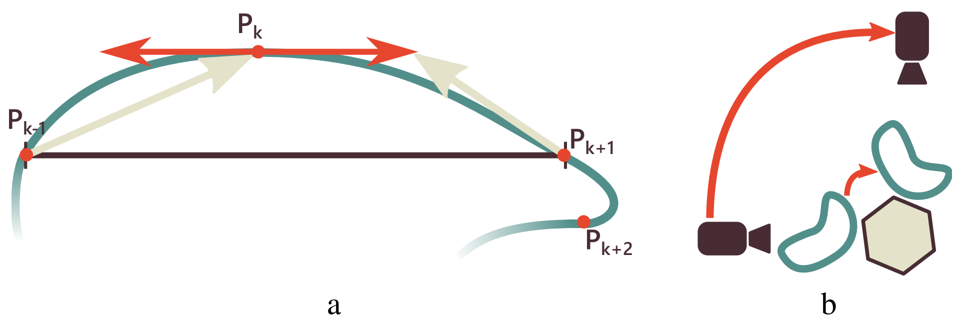


Figure E.3: (a) To reduce the complexity during the optimization process we compute the tangents at point P_k automatically from the neighboring points P_{k-1} and P_{k+1}. (b) Assuming that small changes of the camera position introduce only small changes to the scene, we transform the Firefly path with the camera around the object of interest to provide an initial path.

currently displayed, the Firefly transitions to the new best light path, without interrupting the optimization process. If the user moves the camera during the optimization, the existing Firefly path is transformed accordingly. To provide the user with visual ques on the position change of the Firefly, we display a tail behind the moving light source. The user is able to remove or adjust the tail through a simple slider. During the development of the Firefly, we found that users preferred a Firefly with a tail as reference.

E.3.1 Path Generation and Adaptation

Before discussing the optimization process of the Firefly path, we briefly describe the generation of the initial Firefly path and the adaptation to the camera movement. The Firefly path is defined by a set of control points as a smooth, closed, three dimensional curve. While our system is not limited to a specific path formulation, we implement the Firefly path as collection of cubic Bezier curves, with a C^1 transition at the endpoints. To reduce the complexity during the optimization, we automatically compute the tangent vectors for the Bezier curve and use the segment endpoints as optimization parameters.

Because we designed the Firefly system with the goal of high flexibility, we generate the initial Firefly path as a simple shape that is able to adapt very fast, i.e., as a circular path between the camera and the object of interest. The simple circular shape has several advantages, as it has the lowest curvature of all possible shapes in the same space, thus avoiding sudden trajectory changes at the initialization. Furthermore, the circular shape naturally provides well distributed initial control points for the optimization process, as the control points are evenly spaced and cover a relatively large portion of the design space. Thus, the circular shape provides a fast convergence rate at the beginning of the optimization process. A common approximation for a circle with n Bezier curves is to construct the tangents at a point p_k with the length l as:

$$l = \frac{4}{3} \cdot \tan\left(\frac{\pi}{2n}\right), \tag{E.1}$$

and the same direction as the circle tangent. To keep the Firefly path consistent, we keep the initial ratio of the tangent length and the distance between neighboring points fixed. Having saved the initial tangent length l and the initial distance d between two points p_{k-1} and p_{k+1}, we can compute the new tangent t_k at the point p_k as:

$$t_k = (p_{k+1} - p_{k-1}) \cdot \frac{l}{d} \tag{E.2}$$

We illustrate the automatic tangent computation in Figure E.3(a). The initial placement of the path is inspired by photography rules discussed in Section E.3.5 as an elevated key light.

As the control points are updated during the optimization, the Firefly transitions from the old to the updated path on a linearly interpolated trajectory. The intermediate position of the Firefly is then computed as:

$$P_{trans} = P_{new} \cdot \frac{t}{w} + P_{old} \cdot \frac{w-t}{w}, \tag{E.3}$$

where w is a time window for the interpolation and t is the time that has passed since the interpolation start. In our implementation, we use a window size w of three seconds. As the sum of two smooth functions is smooth as well, the transition occurs without undesired jumps of the Firefly. When the user changes the camera position, the Firefly path needs to adapt to the changes in the visible scene. A reasonable assumption is that small changes to the camera transformation result in small changes to the visible scene and thus the path energy. Therefore, to create a well suited initial condition for the path, we rotate the Firefly path together with the camera position around the object of interest. We illustrate this transformation in Figure E.3(b).

E.3.2 Energy Function

We aim for a flexible and easily-adjustable lighting system that automatically creates a well-suited light path for various application scenarios. To achieve this goal, we draw inspiration from approaches as active contour models [115], originally developed in the context of image segmentation, where the outline of an object is modeled as an energy-minimizing deformable curve using a combination of energy terms for the contour shape and image properties. This provides a high degree of flexibility in modeling specific application requirements using different variations of the individual energy terms. Inspired by this concept, we define the Firefly path through a plugin-based energy function. The optimization of the Firefly path can be formulated as a minimization problem of the used energy function. As such, the choice of the energy function directly determines the shape and the evolution of the Firefly path. In this section, we discuss the components of an energy function for light path optimization on a conceptual level, while we introduce detailed formulations of energy functions for specific application scenarios in Section E.5.

Previous approaches for lighting design are guided by rules taken from photography or image properties such as entropy. In addition, our approach needs to take the path shape into consideration as well. For instance, drastic and unexpected changes of the light trajectory can be confusing for the user [118], as they might be misinterpreted as object movement in the scene. Hence, a general energy function consists of two components, one component for the rendered scene and one component for the path

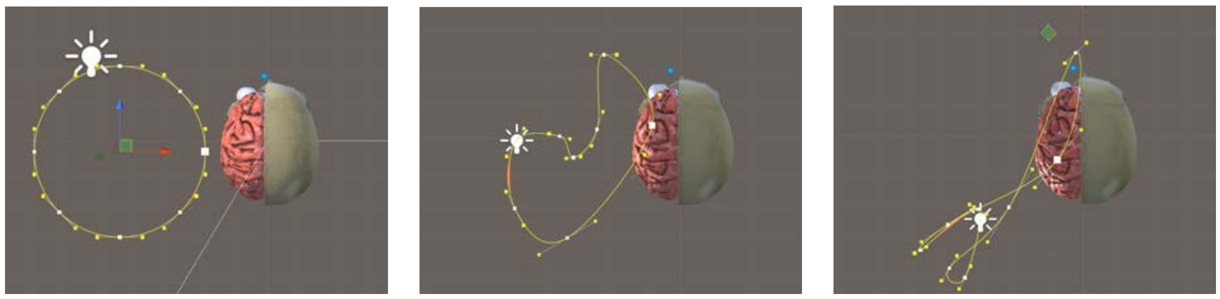


Figure E.4: The influence of α on the light trajectory shape. The weight is set to 0, 0.95 and 0.999 in (a), (b), and (c), respectively. Using $\alpha = 0$ results in a circular trajectory (a), whereas neglecting the path shape too much can result in sharp trajectory turns (c). An α of 0.95 (b) results in a light path that is able to accommodate the image energy without excessively sharp turns.

properties. A general energy function for light path optimization can be constructed as follows:

$$E = \alpha \cdot E_I + (1-\alpha) \cdot E_P \tag{E.4}$$

where E_I is the image energy, E_P the path energy, and α is a weight to control the influence of the energy components. The weight α essentially controls the degree of directional change of the Firefly path. A value of $\alpha = 1$ discards the path shape completely and can produce very sharp trajectory changes. On the other hand a value of $\alpha = 0$ results in a closed curve with the least trajectory change, i.e., a large circle. Because we ensure C^1 continuity of the path, we can choose a relatively high α without causing excessively sharp trajectory changes. While this weight can differ for various application scenarios, we used $\alpha = 0.95$ for all examples in this paper. In Figure E.4, we illustrate the light trajectories resulting from different values for the weight α. In the remainder of this section, we discuss the image and path energy in detail.

Image Energy: The image energy is the most versatile and important component of the energy function. With image energy, we denote any energy formulation that can be derived directly from the rendered scene. The formulation of this component indirectly determines how the light path will illuminate the scene. As there are countless possibilities to define a meaningful image energy, we illustrate only a handful formulations and their effects on the resulting light path.

As we are interested in how the light path affects the scene as seen from the current camera position, we only consider illumination contributions that are visible from the current camera's point of view. As mentioned before, we formulate the path computation as a minimization problem of the energy function, and therefore the light path must be aggregated into a single energy value. To achieve this, the illumination evaluation must be aggregated over the rendered image as well as over the path. Therefore, we define the image energy as two nested functions. Using the notation I for image and P for the path domain, we define the energy function as:

$$E_I = F_P(F_I(sc)) \text{ or:} \tag{E.5}$$

$$E_I = F_I(F_P(sc)) \tag{E.6}$$

As the respective functions are not necessarily commutative, the order of the evaluation can have critical impact on the resulting energy function and should be chosen according to the application scenario.

The purpose of a light is the illumination of the scene. Therefore the image energy must account for the scene illumination explicitly or implicitly. A straight forward measure of the scene illumination is the measurement of the brightness of the rendered object. A useful target for a Firefly would be the illumination of the scene with a desired intensity. We can formulate such behavior with the following energy function:

$$E_I = max_P\left(avg_I(br(sc) - \gamma)\right) \tag{E.7}$$

where γ is the target brightness. Such a formulation forces all light positions on the path to be close to the desired brightness on average. However, the energy does not penalize an image with too dark and too bright regions, as long it does not change the average brightness of the scene. Changing the order of the functions yields the following energy function:

$$E_I = avg_I\left(max_P(br_{x,y} - \gamma)\right). \tag{E.8}$$

This energy function first creates a new image with the maximal difference along the path for each pixel and then computes the average of the image. This means that we first aggregate over time and evaluate the influence of the light path locally. Therefore, this formulation penalizes paths that produce surface illuminations differing strongly from the desired brightness anywhere on the illuminated object, hence enforcing a uniformly illuminated object for the whole Firefly path.

In Figure E.5, we illustrate the effects of these two energy functions on the Firefly path shape and the illuminated scene. For this example we have chosen γ as 0.3. The first row of Figure E.5 shows the paths of the Firefly. In Figure E.5(a) we show the initial Firefly path, in (b) and (c) we depict a Firefly path after 30 iterations with the image energy defined in Equation E.7 and Equation E.8, respectively. The second row shows a captured scene that illustrates the differences in the energy functions. The initial path comes close to the object, creating an overly bright appearance in Figure E.5(d). Using the $max_P(avg_I)$ energy function enforces a greater distance between the object and the light, but it can create strong shadows, as shown in Figure E.5(e). Using $avg_I(max_P)$ results in a scene as in Figure E.5(f). Clearly, the surface is much more evenly illuminated compared to the one in Figure E.5(e). We want to point out that we have chosen these energy functions to demonstrate the importance of the function order and not necessarily as best suited for certain illumination scenarios. We introduce energy functions tailored to specific application scenarios in Section E.5.

Lighting setups are commonly centered around an object of interest. However, the user may be interested in multiple objects in the scene or might want to change the object of interest. To address this, we allow the user to select objects of interest in the scene though a simple click on the object. When an object is selected, the system generates an object mask M_{obj} to evaluate only the pixels covered by the mask. We illustrate such a mask in Figure E.6(b), where the ink container in Figure E.6(a) is selected as the object of interest resulting in a mask as shown in Figure E.6(b), which discards the background completely. If the user is only interested in parts of the image, they can add an additional input mask M_{in} to emphasize these regions. The input mask

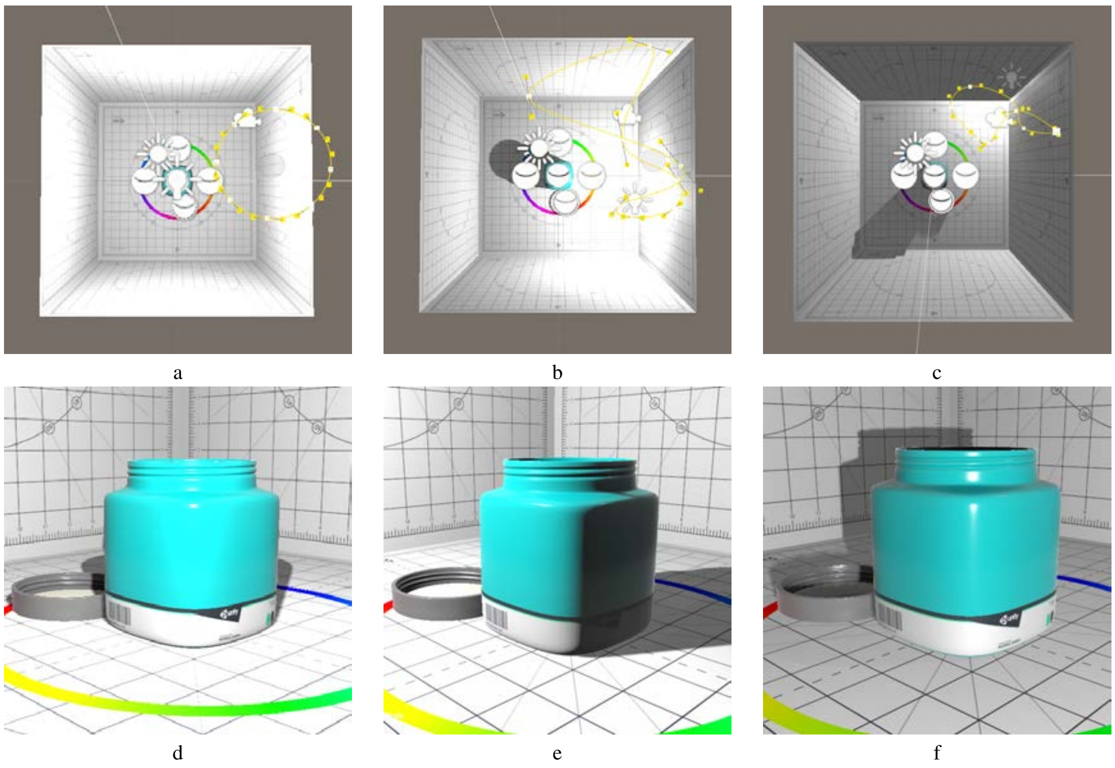


Figure E.5: An illustration of path shapes with varying energy functions. The first row shows the paths seen from above while the second row presents a scene state typical for the energy function. (a) and (d) show the path and a scene for the initial path. The object is too bright for some positions of the Firefly. (b) and (e) show the resulting path for the energy function described in Equation E.7. On this path the Firefly can take positions that create strong shadows in the scene. In (c) and (f) the path was changed according to the energy function in Equation E.8. The scene is much more evenly illuminated compared to (e).

is defined as a smooth function between 1 and 0, that decreases with the distance to the mouse position. A combination of the object mask and input mask can be seen in Figure E.6(c).

Path Energy: The visual system is very sensitive to changes in the lighting conditions. Sudden and unexpected changes of the light trajectory can lead to misinterpretation of the scene dynamics. As indicated by Kersten et al. [118], users can misinterpret unexpected changes of light conditions as movements in the scene. The likelihood of such misinterpretations can be reduced using two strategies: by employing a smooth light trajectory without sharp turns and by giving the user explicit feedback on the light positions. The path energy ensures the former. The degree of how drastically a path is changing can be measured through the curvature of the path. To avoid sharp turns, the path energy is defined as the maximal curvature $\kappa(s)$ over the path P:

$$E_P = \max(\kappa(s)), \;\; s \in P. \tag{E.9}$$

This simple restriction in addition to the C^1 continuous path formulation ensures a smooth trajectory of the Firefly without excessively sharp turns.

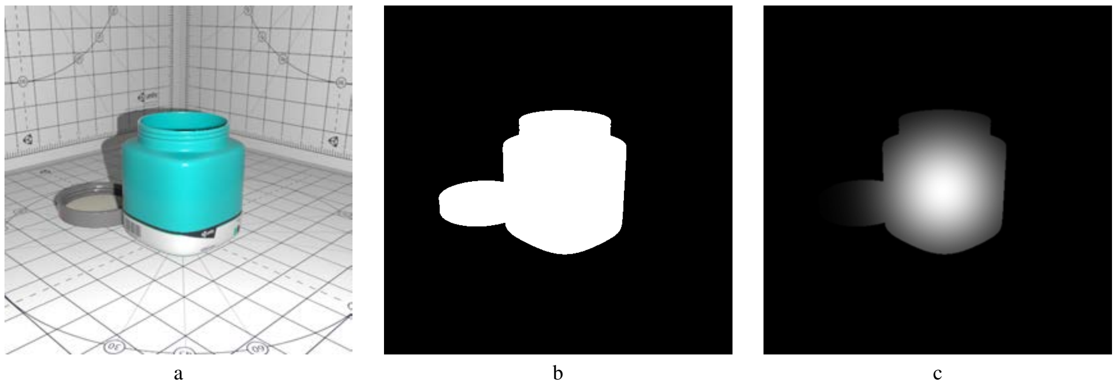


Figure E.6: We allow the user to select objects of interest. Here, the user selected the paint jar in (a) with a object mask shown in (b). If the user wants to further specify a local region of interest, they can use an input mask. The combined object and input mask are shown in (c).

E.3.3 Optimization

Having specified the desired energy function, a well-suited path for the Firefly is one that minimizes the total energy. Choosing a suitable optimization approach is crucial for the quality of the Firefly path. As mentioned in Section E.3.1, the Firefly path is defined by a set of control points. Thus, the dimensionality of the parameter space is equivalent to the number of control points. Hence, there is a trade-off between the problem complexity and the granularity of the direct path shape specification. Because the Firefly is created as a supportive tool during user interaction, the optimization must provide adequate results on the fly. In our examples, we construct the Firefly path with eight control points, which provides enough control for the Firefly path definition, while constraining the optimization to a reasonable degree of complexity that allows us to maintain interactivity.

As the energy functions are dependent on the scene composition, they can be highly non-convex, resulting in many local minima. This imposes a challenge for many gradient descent algorithms. Even modern methods such as Adagrad, AdaDelta, RMSprop, and NADAM [178] can get stuck in local minima. To overcome local minima, we implemented an adapted version of the Simulated Annealing (SA) algorithm [217]. Traditional SA considers one neighboring state $\tilde{S}$ of the current state S, and decides whether to update the system state to $\tilde{S}$, based on a temperature-dependent probability function. In each iteration, the system temperature is decreased, stabilizing the energy state, until the computational budget has been used up. In our approach, we always evaluate an ensemble of neighbors at once, thus increasing the convergence speed of the algorithm.

A straight-forward adoption of SA to ensemble sampling is a Monte Carlo sampling of the ensemble over the parameter subspace. However, several publications [117, 189, 241] show an improved convergence rate of Latin hypercube sampling over Monte Carlo, due to its improved space filling properties. Therefore, in each iteration, we perform a Latin hypercube sampling, computed using the method of Stein [197]. We assume a normal parameter distribution and no parameter correlation for the sampling process. In each iteration, we use the best state of the ensemble for the update decision.

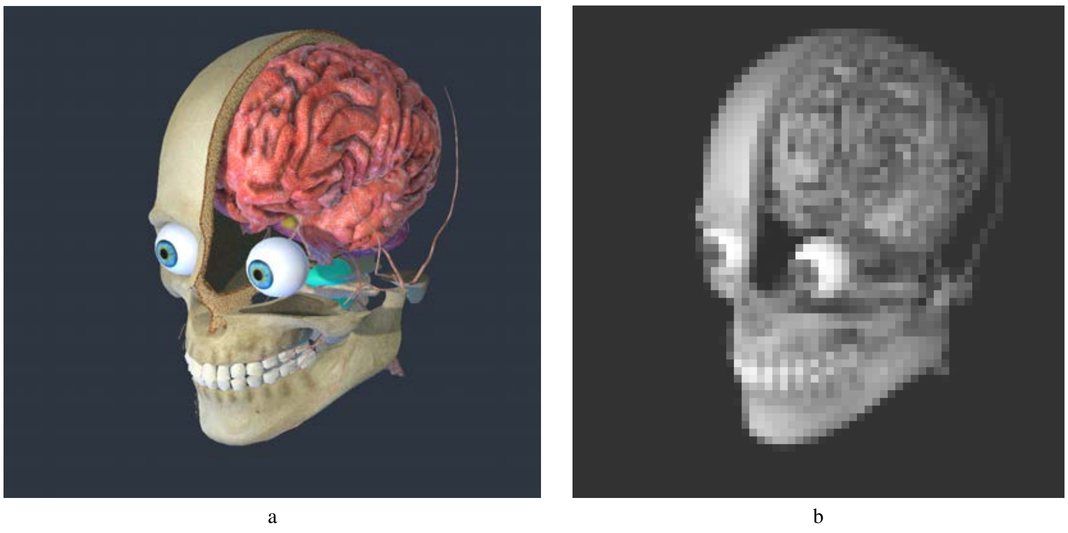


Figure E.7: The brightness values of the rendered image in (a) are partitioned into 64×64 segments resulting in a down-sampling shown in (b). This partitioning provides a good trade-off between the preserved level of detail and data reduction.

E.3.4 Sample Evaluation

As mentioned previously, computation speed is crucial for interactive scenarios. A common bottleneck for light design approaches is the evaluation of the scene. This is even more true for our approach, as the scene needs to be evaluated not just for different light positions but for different light paths. To approach this challenge, we can essentially employ two strategies: increase the computational performance of our approach and reduce the computational complexity of the problem.

To increase the performance of our approach, sampling and evaluation of the scene are performed in a detached parallel rendering pipeline. We employ a worker thread pool to generate the scenes and collect their information for each a test path. Optimization techniques require an aggregation of the fitness to a numeric value. For the Firefly path, the fitness is computed through the energy function. As mentioned in Section E.3.2, the order of the energy function components affects the order of the aggregation over the image domain and the path domain. If the aggregation is performed over the image domain first, then the worker can return a single numerical value for each scene. However, if the aggregation is performed over the path domain first, then the worker would need to store the scene information for every light sample on the path before performing the aggregation.

Clearly, this creates a substantial overhead for the worker performance. To address this challenge, we need to find a suitable trade-off between computational efficiency and the preservation of features of the energy function. Usually, light affects the scene over neighborhoods instead of isolated points. Therefore it is reasonable to assume that neighboring pixels will have similar energy characteristics. We use this fact to abstract the localized energy states in the scene through partitions that store the average energy of their pixels. We illustrate the partition of the illumination energy in Figure E.7. In our current implementation, we divide the scene image into 64 by 64 partitions, which still captures enough details. For each path, the workers evaluate the partitions first

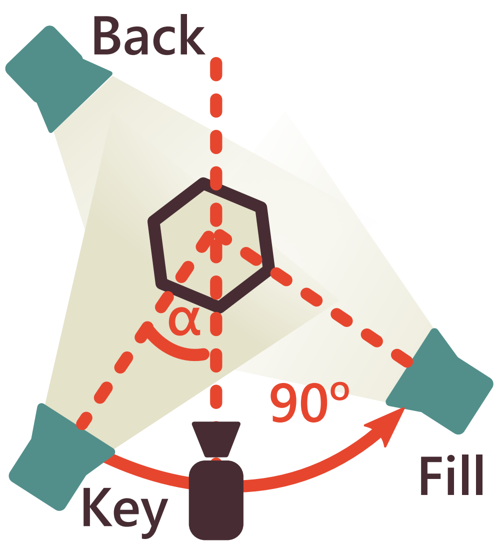


Figure E.8: Firefly can be easily integrated into existing lighting setups. In this Figure, a Firefly is integrated into a three-point lighting setup as the key light at an angle α of 30°. A fill light is positioned at 90° to the key light. The back light is placed behind the illuminated object with respect to the camera position.

over the path domain and then over the image domain according to the energy function, and then report the resulting image energy as a single value.

E.3.5 Lighting Design

The three-point lighting setup is a common lighting method used in photography, cinematography, and computer-generated imagery. It is a relatively simple but versatile approach which forms the basis for most lighting setups. In the following, we briefly discuss the integration of Firefly into a basic version of three-point lighting as described in several photography text books [95, 175]. As the name suggests, three-point lighting uses three light sources, a *key light*, a *fill light*, and a *back light*, as shown in Figure E.8.

The key light is the main and usually the strongest light of the setup. The goal of the key light is to produce tonal variations in the image. Therefore, it is usually placed to one side of the illuminated object in order to produce shadows visible to the camera, as illustrated in Figure E.8. A strong key light can create very strong shadows and thus hide geometric details. To overcome this problem, a secondary light is used to "fill" out the shadows with a soft light. The fill light is often placed at 90° to the key light (see Figure E.8). The fill light is usually less bright and softer than the key light, playing only a secondary role for the illumination. The back light is placed behind the illuminated object. Instead of providing direct illumination of the object, the back light emphasizes and provides subtle highlights of the object's silhouette. This helps to separate the object from the background and provides additional depth cues.

We integrate our Firefly into a three-point light setup by considering it as the key light. A common placement of the key light is at an angle α of 30° to 45° at the side of the camera and the illuminated object, as shown in Figure E.8. Furthermore, the key light is commonly placed slightly above the camera. We initialize the Firefly path

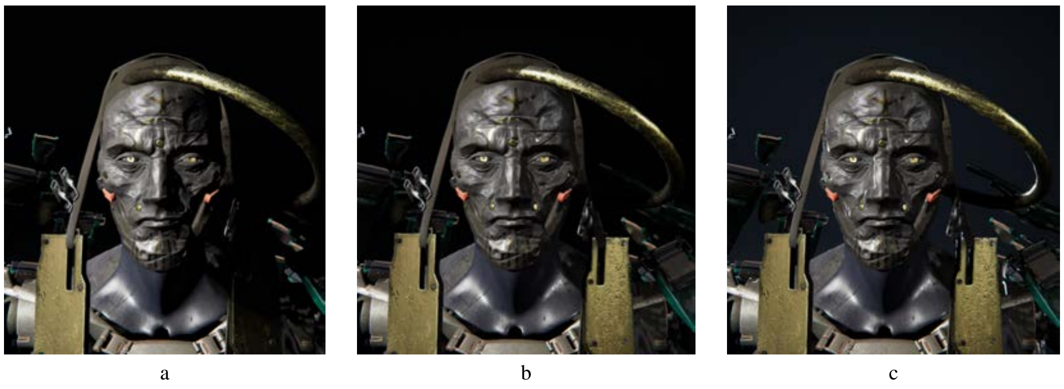


Figure E.9: A comparison between (a) a single key light, (b) a combination of key and fill light and (c) a three-point lighting setup.

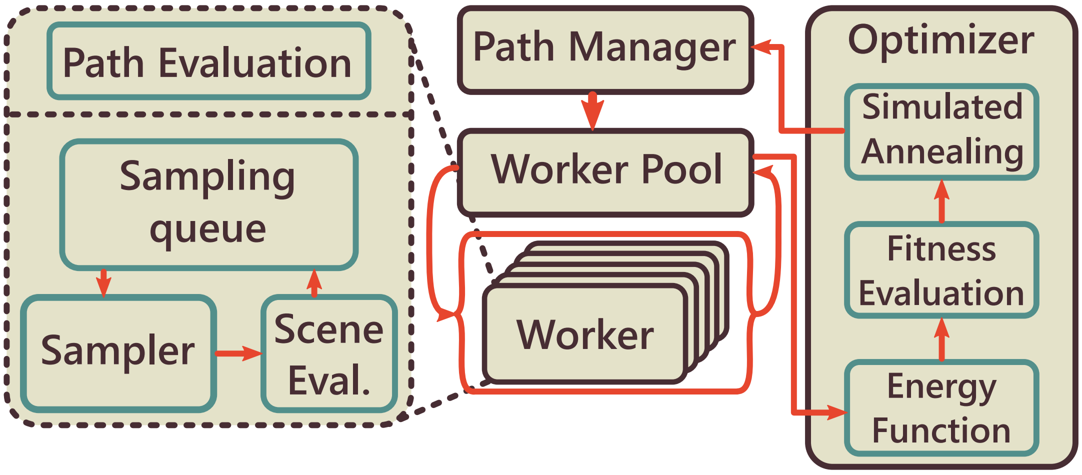


Figure E.10: The Firefly system is implemented as a background process running parallel to the main thread. To achieve on-the-fly adaptation of the Firefly path, we evaluate 40 paths at once in a worker pool. When the paths are evaluated, the information is sent to the optimizer that computes the path energy and requests further samples.

position at an α value of 30° and a relative elevation above the camera equal to half of the path radius. The fill light moves with the key light and is placed at an angle of 90° to the key light rotated on a plane defined by the right and front camera vectors. The back light is placed behind the object in the view space. However, this setup should be considered as guideline rather than a rule, as the Firefly path converges quickly even when poorly initialized.

In Figure E.9, we illustrate the effect of the number of lights on a model from the animated short film Adam [2]. In Figure E.9(a), only the key light is used and in (b) a fill light was added to the scene creating a more even illumination. In Figure E.9(c), the addition of a back light provides additional subtle highlights of the geometry.

E.4 Implementation

We implemented the Firefly system as a camera component in Unity [15]. Unity is a popular multi-purpose engine that supports 2D and 3D graphics. The drag-and-drop functionality of Unity and the C# scripting interface provide fast prototyping possibilities. Furthermore, the efficient plugin system allows for easy sharing of the results across multiple platforms. To use the system, the user simply adds Firefly as a camera component to the scene setup. For selecting the object of interest, Firefly only requires that the 3D objects have a collision geometry and a unique name.

Firefly is implemented as a background process, running in parallel to the main rendering thread, as illustrated in Figure E.10. When an object of interest is selected, a new Firefly path is initialized and the optimizer creates a list with control point offsets that is sent to the path manager. The path manager creates the light paths that are evaluated in a worker thread pool. The workers create a sampling queue of scenes that are rendered and evaluated in a process that is fully transparent to the user. The primary benefit of a worker thread pool over creating a new thread for each path evaluation is that the thread creation and destruction overhead is restricted to the pool creation. The size of the worker thread pool depends on the used hardware. We found that on an Intel Core i7 3.00 GHz CPU, a worker thread pool of size five delivered the best performance.

Each worker evaluates one path at a time by computing its image and path energy. The image energy is computed on the GPU by rendering the scene for uniformly-sampled Firefly positions along the path. Thus, the number of scene evaluations directly corresponds to the evaluation time for the image energy. During the development of the Firefly system, we found that evaluating 14 samples for each path provides a robust estimate of the path quality while still allowing for a fast computation. When a worker has finished the evaluation of a path, the information of this path is stored in the worker pool. When all paths have been processed, the worker pool sends the results of all paths to the optimizer.

In the optimizer, the collected information is used to compute the final energy function. Next, the optimizer evaluates the fitness of the paths and possibly updates the Firefly path to a better one. The last step of the iteration is the computation of a set of new sampling parameters that are sent to the path manager. On an Nvidia GeForce GTX 780 GPU and an Intel Core i7 3.00 GHz CPU and a screen resolution of 1920×1200, one such iteration requires on average 973 ms for the evaluation of 40 paths with 14 rendered images for each path. The main thread is virtually unaffected by this computation and we did not detect a noticeable drop of the frame rate for the rendered scene.

E.5 Results

In the following, we demonstrate the use of Firefly for four different scenarios. To show the versatility of our approach, we selected examples covering scientific visualization applications as well as scenarios inspired by applications in the entertainment industry. All Firefly paths described in this paper are initialized according to the basic three-point lighting setup. In each iteration, the sampling is performed with 40 test paths and 14 samples for each path. As it is difficult to fully capture the dynamic behavior of our approach in text and still images, we encourage the reader to also refer to our supplemental video.

E.5.1 Molecular Structures

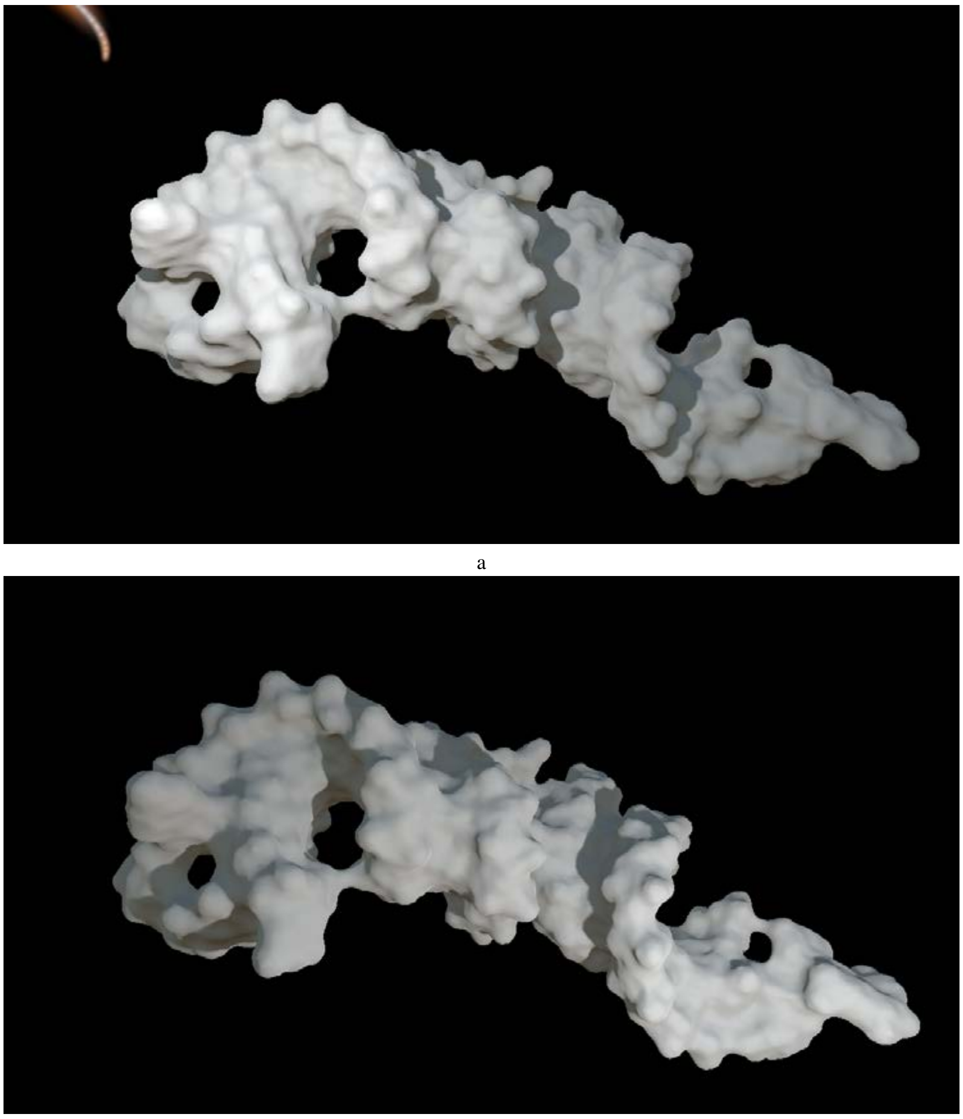


Figure E.11: Input mask guided illumination. In (a) the mask was held over the left side of the molecule. In (b) the right molecule side was masked. The Firefly path adapts automatically to the input mask. This way the user is able to adjust the illumination indirectly by moving the input mask.

The exploration and analysis of molecular data is a prominent topic in scientific visualization. The purpose of molecular visualization is to provide an understanding of the rich and highly complex world of atomic structures, by mapping molecular structures, their functionality, properties, and interactions to visual characteristics. Molecular data is commonly very crowded, and the visualization of molecules features high visual complexity. The illumination of such complex geometry is a challenging task. In this example we use a 3D model of a tRNA structure 1GAX, consisting of 17210 atoms. This molecule has several deep cavities and tunnels, that pose a challenge to an illumination setup. A static setup will often result in deeply shadowed cavities and tunnels, hindering the geometry assessment. We can address this challenge by guiding the

Firefly to the regions we want to illuminate using an input mask. We define a simple energy function that illuminates the surface with a desired brightness value of $\gamma = 0.3$. As strong shadows can be beneficial for shape perception, we use the energy function given in Equation E.7. In Figure E.11, we show two images of the same scene illuminated with two different Firefly paths. In Figure E.11(a) ,the mask was set to cover the left side of the molecule, while in (b) the right part was the focus. The Firefly path automatically deforms to create a better energy for the masked part only. One can clearly see that tunnels on the left side of the molecule are much better illuminated in the top image.

E.5.2 Human Anatomy

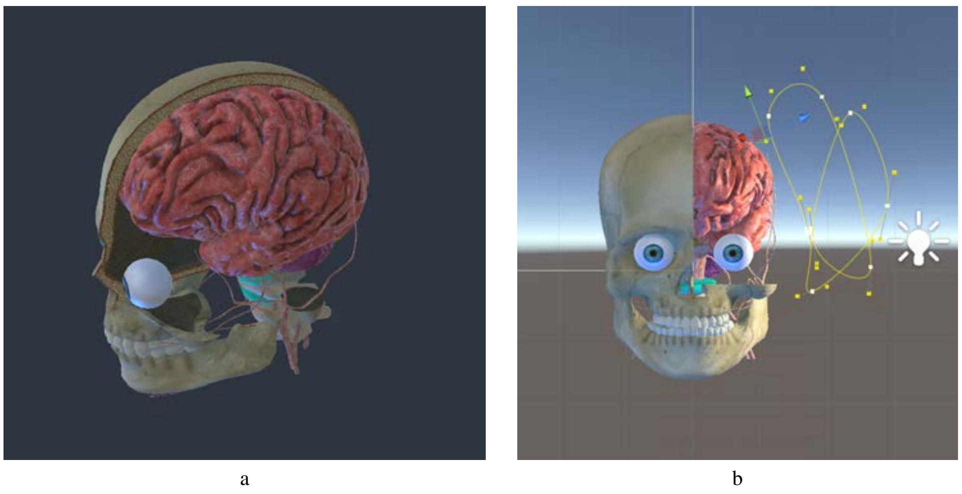


Figure E.12: (a) A model of the human head with a static light setup. (b) A firefly path after 34 iterations.

Advances in computer technology have profoundly affected the domain of medical education. It has been shown that using 3D computer models as a teaching medium of human anatomy significantly improves the recollection of anatomical structures [155]. Often such models are slightly exaggerated to emphasize structures and textures of the anatomical objects. The visualization of such structures highly benefits from a lighting setup that highlights the variations of the model surfaces. However, as the models tend do be highly complex, it is difficult to find a static light source that provides a good illumination of all relevant structures. We illustrate such a scenario on a model of the human head [14]. We show the model illuminated with the preset static lights in Figure E.12(a). The complex spatial relationships between the individual structures can significantly benefit from a moving light source. Intuitively, a well-suited light path should not illuminate the object near to the camera position as this would decrease perceived shape variation [143]. Instead, the light should illuminate the object from several sides to account for all aspects of the model surface.

The shape of the brain gyri and sulci is most prominent with a high contrast between the valleys and ridges of the surface. We can indirectly measure the local contrast of the model through a local variance measure. However, using the variance alone does not account for the brightness of the image. Therefore, the image brightness must be

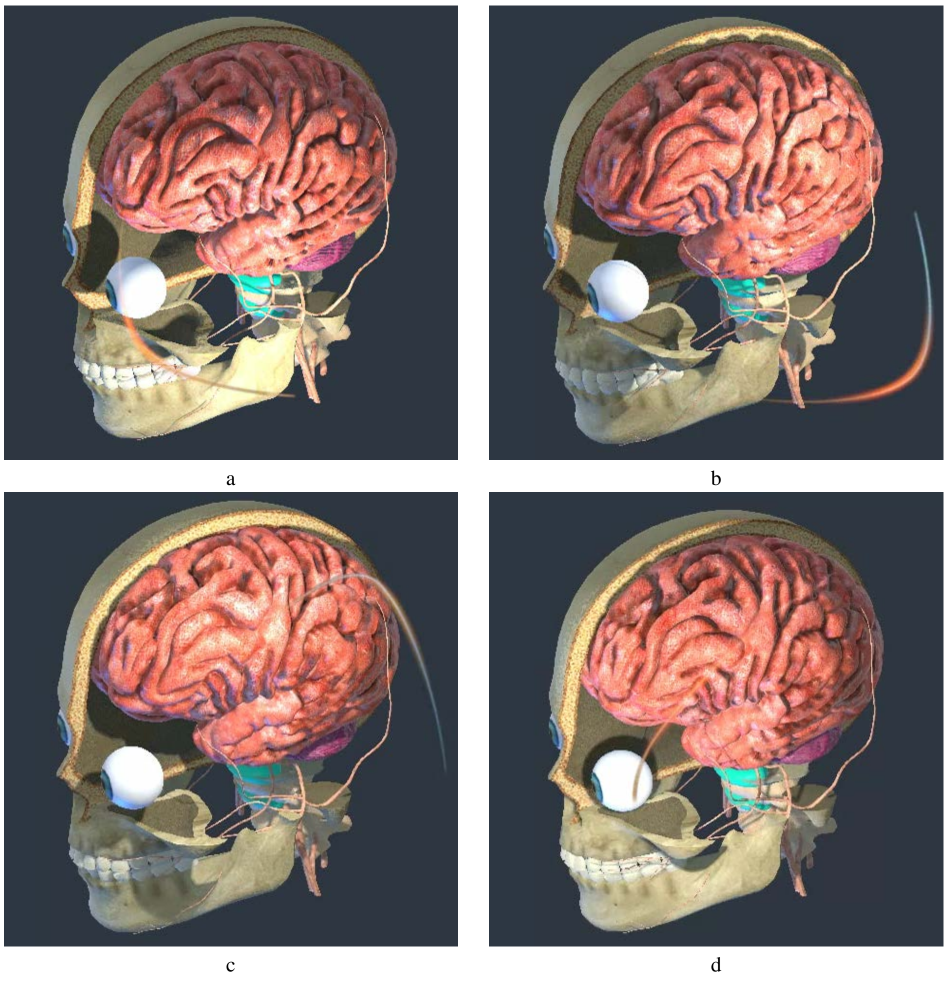


Figure E.13: Four consecutive positions of the Firefly. The Firefly travels on a loop illuminating the head from (a) below, (b) right, (c) above and (d) left.

included in the energy as well. Because the image brightness value is much higher than the variance, we multiply the variance with an importance factor ξ. To allow for stronger shadows, we measure the average instead of the maximum brightness for this example. Incorporating the difference in brightness and variance results in the following energy function:

$$E_I = \sum_{x,y} \xi \cdot (min_P|(var(sc,s) - \theta|) + avg_P(|br_{x,y} - \gamma|)) \tag{E.10}$$

where θ is the desired variance. Here, we used $\theta = 0.02$ and $\xi = 5$. Using this energy function favors paths that emphasize high variance in the scene while maintaining uniform illumination. In Figure E.12(b), we show the resulting Firefly path after 34 iterations. One can see that the Firefly path resembles two loops with a relatively uniform distance to the model. We show four consecutive snapshots for this Firefly path in Figure E.13. One can see that the Firefly indeed travels on a path that illuminates the scene from different directions (up, down, left, right). Interestingly, the Firefly path forms a loop, as this light trajectory is better suited to emphasize the features at the edge of the brain as well as the ones in the center of the model.

E.5.3 Animation

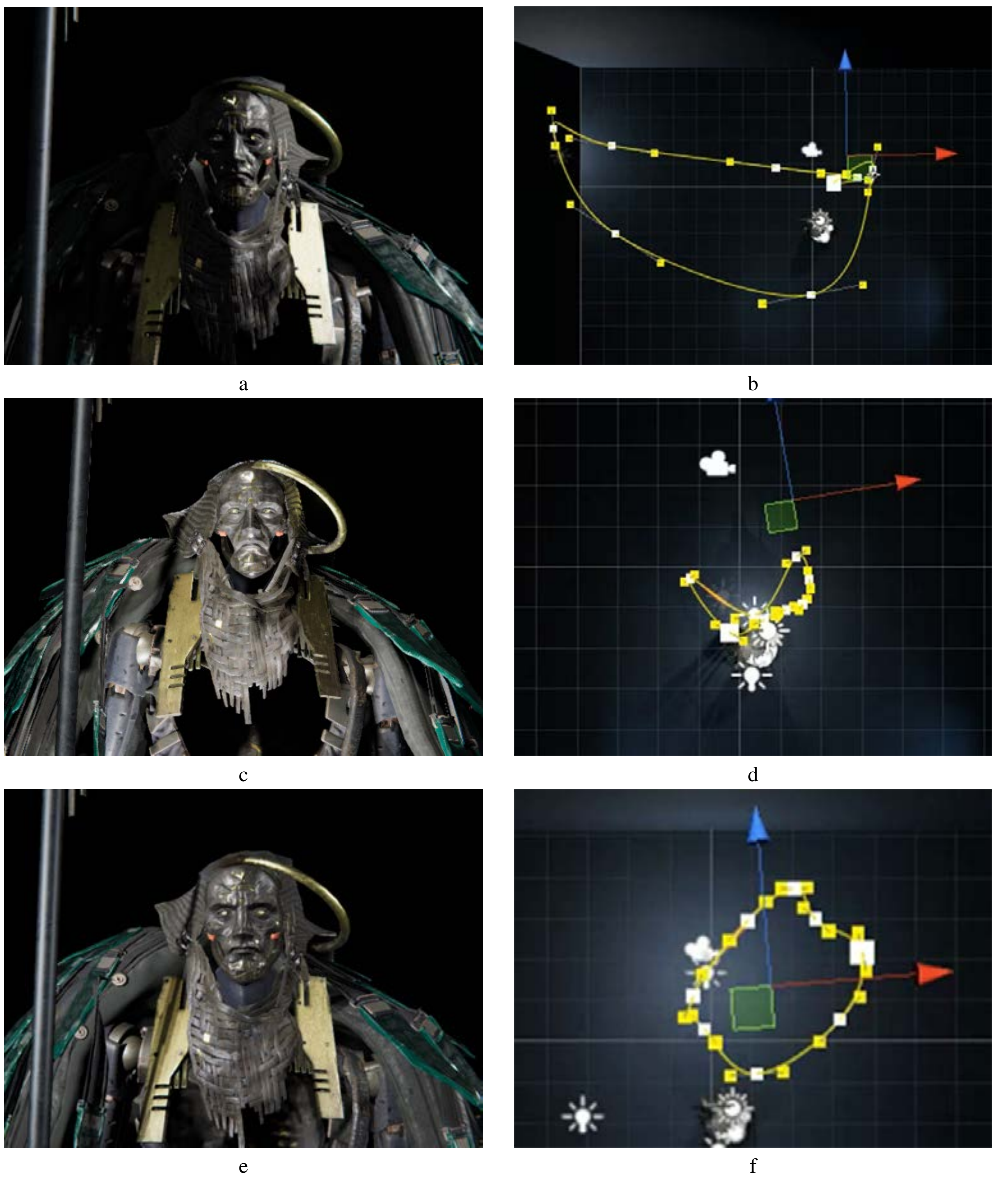


Figure E.14: Difference between scenes with different moods encoded through the energy function. (a) shows a dramatic scene, (b) a threatening scene and (e) a calm scene. Images (b),(d) and (f) show the corresponding paths, respectively.

Lighting setup is a crucial part of cinematic animations. In addition to static lights, many animation techniques employ dynamic lights as they can be used to convey the passing of time, emphasize a dramatic change of a character, or change the focus of attention. When an illuminated character moves in the scene, the dynamic light needs to adjust to this motion, which poses a challenge on the animator to carefully construct a lighting path that still creates the desired result while in motion. With our approach this challenge can be addressed through the automatic adjustment of the Firefly path. As the Firefly path is defined relative to the object of interest, it moves with the object automatically adapting to the changing conditions. In this example, we use a character from the animated short film Adam [2]. We illuminate the same scene with three different illumination setups that create different effects for the user. The first setup creates a dramatic atmosphere, the second a threatening one, and the third setup aims to produce a calm atmosphere.

To create a dramatic effect, cinematography often employs rim lights. Rim lights are used to create dramatic scenes with a Chiaroscuro lighting. This artistic technique, developed in the Renaissance, uses strong lighting on one side of the object to create distinct one-sided shadows. We can formulate this by maximizing the variance and requiring a certain brightness in the image. To restrict the effect to one side of the object, we compute the dot product between the normalized vector from the Firefly to the object l_o and the right vector r in camera space. Minimizing the dot product favors light positions on the right side of the object.

To create a threatening scene effect, we follow the guidelines of digital cinematography [27]. Disregarding the light color, a threatening effect can be generated by using a low key light, that has a light intensity ratio of 8:1 between the key and fill light, and a strong back light. Furthermore, the light should illuminate the object from underneath creating a high variance in the scene.

In contrast, a calming atmosphere can be achieved with a high key light, i.e., a similar intensity of key and fill light. This setup is reduces the overall variance in the image. In addition, the light elevation should be close to the elevation of the camera. Following these guidelines we can define the three energy functions as:

$$\text{dramatic:} \quad E_I = \sum_P -var_I(s) + |br_s - \gamma| + (l_0 \cdot r) \tag{E.11}$$

$$\text{threatening:} \quad E_I = \sum_P -var_I(s) + |br_s - \gamma| + |el + \frac{\pi}{3}|\, ds \tag{E.12}$$

$$\text{calm:} \quad E_I = \sum_P var_I(s) + |br_s - \gamma| + |el|\, ds. \tag{E.13}$$

Where el is the elevation angle in radians. In Figure E.14, we show a representative result for the energy functions and the corresponding paths. One can clearly see the different moods present in the images. Figure E.14(a) shows a dramatic scene, (c) a threatening scene, and (e) shows a rather calm atmosphere. The corresponding paths are shown in (b),(d), and (f).

E.5.4 Still Life

The assumption of light position is stronger for scenes with a familiar setup. Illuminating such a scene with a Firefly constructed with the previously described energy functions might create an odd scene not matching the user's expectations. Therefore, when designing an animated light for a realistic scene, for example a still life, we need to closely follow photographic lighting setup rules.

In this example, we illuminate a still life and formulate the energy function guided by the approach of Wambecke et al. [222]. This method, based on photographic rules, optimizes the azimuth and elevation of the light to emphasize the surface variation of the object while constraining the light position to the upper front hemisphere. Illuminating the scene with this method results in a rendering as shown in Figure E.15(a).

A Firefly constructed in accordance with this method should be arranged above the camera and the object, with an azimuth range that accounts for the majority of the

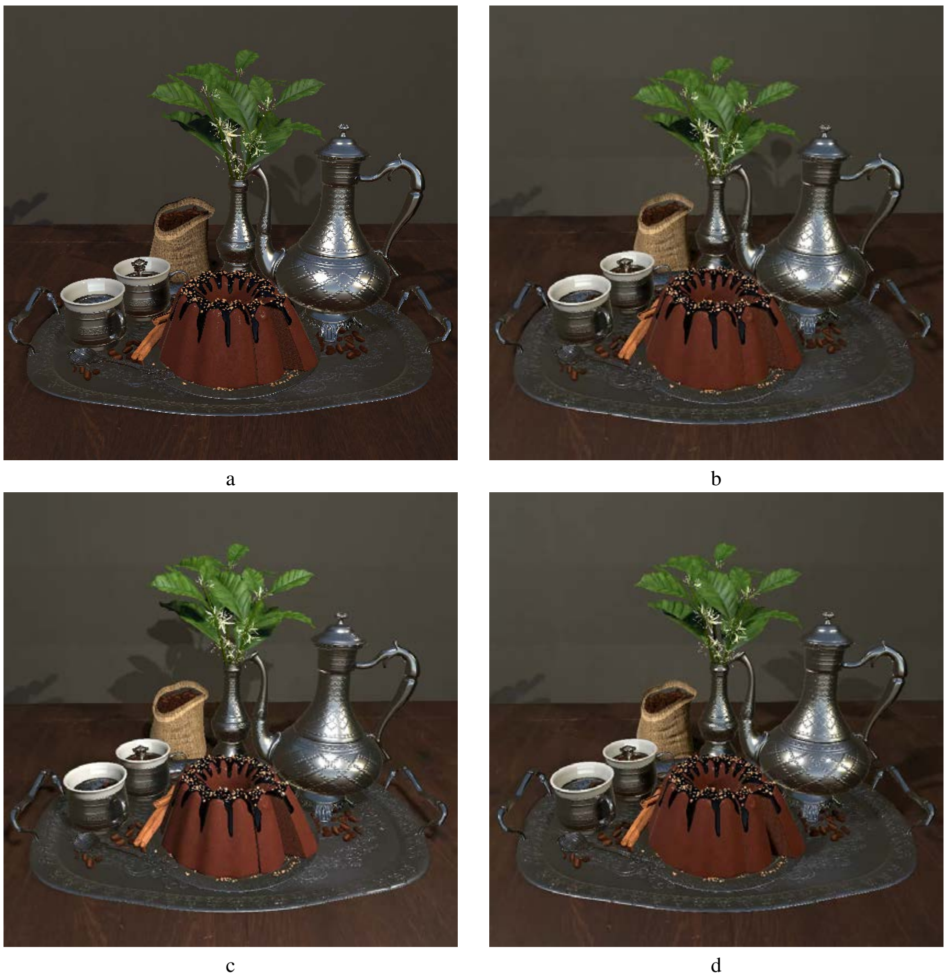


Figure E.15: Still life rendered with the method of Wambecke et al. (a). The same model illuminated with the Firefly method (b,c,d). The lighting in (a) and (b) is almost identical, but (c) and (d) reveal the surface variation of the remaining model parts.

surface variation. Following the approach of Wambecke et al., we break down the light position into the light azimuth and elevation. The azimuth is computed using the structure tensor of the geometry measuring the direction of surface variance, and the elevation is used to produce a grazing light. For more details, we refer to the paper of Wambecke et al.[222].

For the azimuth, we first obtain the gradient of the surface depth ∇d for each pixel, and compute the local structure tensor $S_{x,y}$ for each image partition $P_{x,y}$:

$$\nabla d = \left(\frac{-n_x}{n_z}, \frac{-n_y}{n_z} \right) \tag{E.14}$$

$$S_{x,y} = \sum_{p \in P_{x,y}} \nabla d \nabla d^T . \tag{E.15}$$

The eigenvector e_1 of S associated to the largest eigenvalue λ_1 represents the direction in which most surface variations appear for each partition. If e_1 is pointing downwards, then the vector is simply flipped. Wambecke et al. define the light azimuth at an angle of e_1 in view space. In our approach, we measure the fitness of the Firefly azimuth by computing the dot product between e_1 and the normalized vector from the Firefly to the center of gravity of the illuminated object l_o.

To favor light positions close to e_1 for the whole path, we define the azimuth energy as:

$$E_{az} = max_s\,(1 - (e_1 \cdot l_o))\,. \tag{E.16}$$

The elevation of the light is chosen such that l_o is orthogonal to hidden surface normals n_0 defined as:

$$n_0 = \begin{cases} n & \text{if } l_{e_0,o} \cdot n > 0 \\ \vec{0} & \text{if } l_{e_0,o} \cdot n \leq 0 \end{cases} \tag{E.17}$$

where $l_{e_0,o}$ is the vector between the light position projected on an elevation of 0 and the object of interest. Therefore, the elevation energy is defined as:

$$E_{el} = max_s(l_{e_0,o} \cdot n_0)\,. \tag{E.18}$$

Incorporating the information of the optimal light direction can lead to light positions too far away or too near to the object, resulting in too weak or to strong illumination. A straightforward solution would be to define a minimum distance between the object and the Firefly, but such a condition could produce rather unnatural illumination. Using the energy function, we can instead easily control the distance between the object and the Firefly indirectly by minimizing the maximal brightness difference for each image partition.

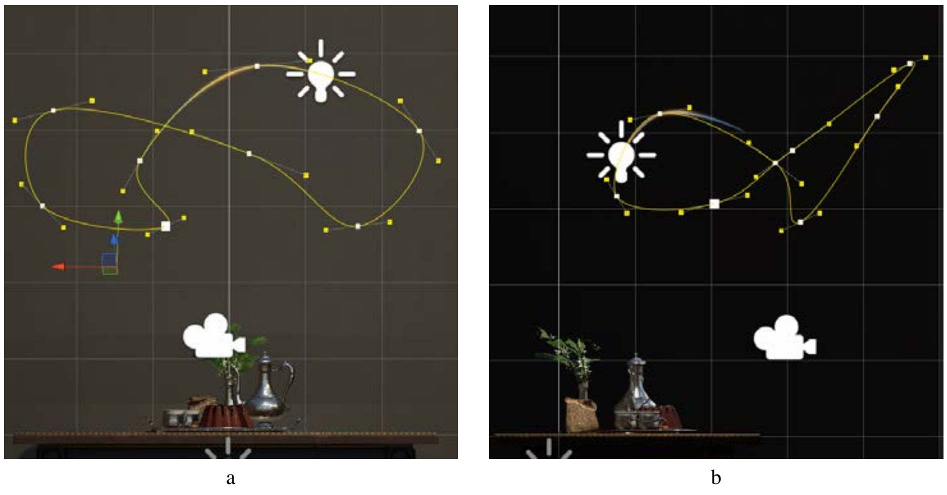


Figure E.16: The Firefly path for the still life scene seen from the front (a) and from the side (b). In (b), the Firefly moves away from the object briefly to generate illumination from the front that does not produce shadows on the object.

Putting all parts of the energy function together, the image energy is evaluated as a sum of the individual energy states over the image:

$$E_I = \sum_{x,y} (E_{az} + E_{el} + max_P(|br_{x,y} - \gamma|))\,. \tag{E.19}$$

Where x and y are the partition indices. The desired brightness γ was set to 0.3. Using this energy function results in a scene illumination shown in Figure E.15(b), (c), and (d). We show the static lighting setup computed with the method of Wambecke in Figure E.15(a). Naturally, the Firelfy produces very similar results as the static state-of-the-art method – in fact the light positions in images (a) and (b) are almost identical. However, for the static setup we had to adjust the distance between the light source and the object to achieve the desired brightness. Our method adjusted the distance automatically based on the desired brightness value. Furthermore, our method emphasizes all of the geometry throughout the path. In Figure E.15(c), the Firefly creates shadows that emphasize the shape of the cake and in Figure E.15(d) the shape of the coffee bag is much more prominent. We show the corresponding Firefly path in Figure E.16. As expected, the path is organized above and in front of the object, forming an arc on the azimuth plane. In Figure E.16(b), one can see that the path moves away from the object over a short path section in front of the object. This movement of the light away from the object generates a light position that produces almost no shadows on the object, thus creating an overall stronger illumination of the scene. To account for this, the light moves farther away in order to achieve an illumination close to the desired values.

E.6 Discussion

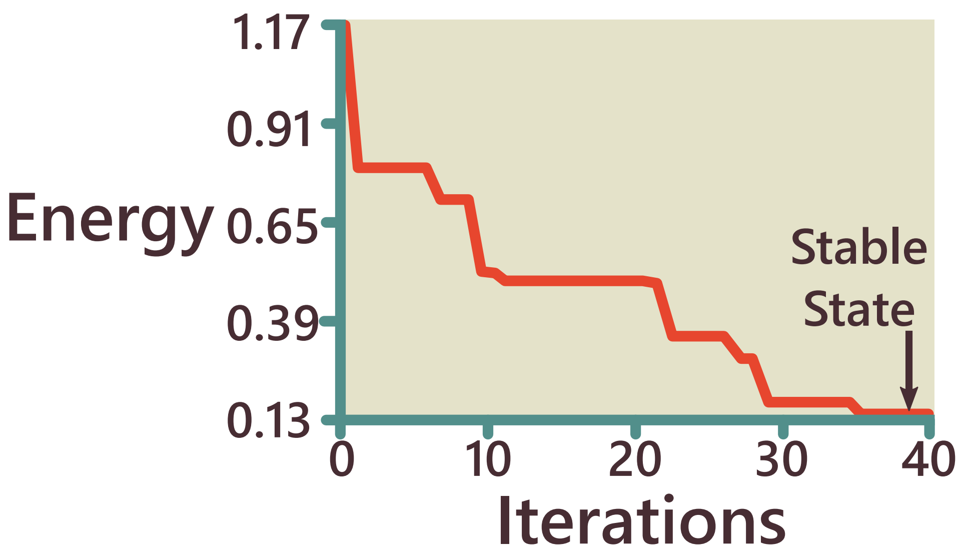


Figure E.17: A representative convergence curve for an energy function described in Equation E.8. The Firefly path significantly improves already after a single iteration. After 10 iterations (approximately 9 seconds) the energy was more than halved. After 30 iterations, the energy state was almost optimal. The stable state was reached after 36 iterations requiring just under 33 seconds.

In our experiments we found that the animated light of the Firefly provides a powerful and versatile addition to a static lighting setup. We deliberately chose relatively simple energy functions for most of our examples, but nonetheless showed that our approach is capable of incorporating state-of-the-art solutions for static lighting design in a dynamic setup. The Firefly approach can be easily integrated into other renderers, but we believe that our implementation as a Unity plugin makes it already accessible to a large number of applications. Our method is independent of the scene content and rendering method, and hence can be used with a wide variety of different types of data.

While the evaluation of the test paths requires around 500 milliseconds on average, the fact that this process is executed in the background makes it transparent to the user. After the initialization, the optimization requires 35 to 40 iterations to reach a stable state. We show a representative convergence rate in Figure E.17. The whole optimization process takes around 34 seconds for full convergence, but since the Firefly continuously updates to the best available solution, a good result is typically already achieved after about 9 seconds. When the user changes the view, the optimization needs 3 to 6 iterations to update the Firefly path to a new stable state, requiring 2.5 to 5 seconds. Because these background computations do not affect the performance of the main rendering thread and the Firefly transitions smoothly into a new path, we did not experience any negative effects on the overall usability. The initialization of the worker pool requires around 9 seconds during the program start, since all the workers need to be initialized with the corresponding Unity scene and settings.

When the user employs a selection mask, regions outside the selection are not accounted for in the illumination optimization. This can cause unwanted effects outside the selection mask, such as strong shadows or change of light intensity. However, we believe that these results are tolerable as the user deliberately shifts the attention to selected substructures. While dynamic light can emphasize and enhance certain aspects of the scene, it can also be misleading in some situations. Dynamic lights might be unsuitable for scenarios with highly dynamic scenes or dynamic textures as the change of light position might be interpreted as a change in the scene. Furthermore, dynamic lights might deflect the user's attention and thus may not be not suitable for applications where high concentration is needed to carefully study the data.

In this paper, we define a limited number of energy functions. To allow for an easy energy function formulation, we construct the energy function as an assembly of building blocks, measuring different properties in the scene and the rendered image. New energy functions can be constructed by using different combinations of these building blocks. However, the current version of the Firefly requires the user to define the energy function by writing a short code segment combining the building blocks. We are working on a visual editor that will allow the user to easily construct custom energy functions by combining the building blocks in a drag-and-drop manner. Finally, the effects of animated lights on the perception need to be formally evaluated, and we plan to conduct a user study investigating their perceptual consequences.

E.7 Conclusion and Future Challanges

We presented a novel approach for the automated generation of dynamic illumination paths in interactive scenes. Our approach shows that animated light provides a powerful and versatile addition to static lighting setups. We designed the Firefly tool as a flexible plugin that can be easily added to various visualization scenarios. The applicability of Firefly was demonstrated on various examples ranging from scientific visualization to applications for the entertainment industry. At present, the Firefly travels with a constant speed over the curve. In the future, we would like to investigate how the speed of the Firefly can be adapted for a better illumination without appearing unnatural to the user. While in this paper we only used point lights, Firefly is not restricted to a specific light type. In the future, we will investigate how Firefly can be integrated into complex lighting setups with box lights, strip lights, ring lights, and reflector probes. Moreover, we will investigate how a Firefly with dynamic chromatic light could emphasize changing moods in the scene.

Acknowldegments

The research presented in this paper was supported by the MetaVis project (#250133) funded by the Research Council of Norway.

«How lucky I am to have something
that makes saying goodbye so hard.»
Edward Bear

Bibliography

[1] A. Paulina-Carabajal, Y-N. Lee, and L.L. Jacobs. Pawpawsaurus campbelli. http://digimorph.org/specimens/Pawpawsaurus_campbelli/. Accessed: 2017-09-21. C.5.3

[2] Adam. https://unity3d.com/pages/adam. Accessed: 2018-03-17. 4.5, E.3.5, E.5.3

[3] Caravaggio. http://caravaggio-a-drawing-machine.webnode.it/. Accessed: 2017-07-27. 2.4, D.2

[4] Cricut. https://home.cricut.com/. Accessed: 2017-05-22. D.1

[5] Datalizer Slide Charts, Inc. http://www.datalizer.com/. Accessed: 2016-03-19. B.1

[6] Frieder Nake. https://en.wikipedia.org/wiki/Frieder_Nake. Accessed: 2017-07-22. 2.4, 3.3.2, D.1, D.2

[7] Georg Nees. https://en.wikipedia.org/wiki/Georg_Nees. Accessed: 2017-07-22. 2.4, 3.3.2, D.1, D.2

[8] Harold Cohen Aaron. http://www.aaronshome.com/aaron/aaron/index.html. Accessed: 2017-07-22. D.2

[9] Jean Tinguely. https://en.wikipedia.org/wiki/Jean_Tinguely. Accessed: 2017-07-22. D.2

[10] The next Rembrandt, can the great master be brought back to create one more painting? https://www.nextrembrandt.com/. Accessed: 2017-05-22. D.1

[11] Pindar Van Arman. http://www.cloudpainter.com/. Accessed: 2017-07-22. 2.4, D.2

[12] Processing. https://processing.org/. Accessed: 2018-02-20. 3.3.2, D.3

[13] Silhouette America. https://www.silhouetteamerica.com/. Accessed: 2017-05-22. D.1

[14] U-anatomy; ufulio anatomy realistic. https://ufulio.wixsite.com/ufulioanatomy. Accessed: 2018-03-23. E.5.2

[15] Unity game engine. https://unity3d.com. Accessed: 2018-03-23. E.4

[16] Yoo Hyun. https://www.instagram.com/yoo.hyun/. Accessed: 2017-08-22. 4.4, D.5

[17] AHLBERG, C., WILLIAMSON, C., AND SHNEIDERMAN, B. Dynamic queries for information exploration: An implementation and evaluation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (1992), pp. 619–626. 2.1

[18] AHMED, A. G. M. Modular line-based halftoning via recursive division. In *Proceedings of Workshop on Non-Photorealistic Animation and Rendering* (2014), pp. 41–48. 2.4, D.2

[19] AHMED, A. G. M. From stippling to scribbling. In *Proceedings of Bridges: Mathematics, Music, Art, Architecture, Culture* (2015), pp. 267–274. D.2

[20] AHMED, A. G. M., AND DEUSSEN, O. Proceedings of the conferece on computer graphics & visual computing. In *Computer Graphics and Visual Computing* (2016), pp. 41–43. 2.4, D.2

[21] AIGNER, W., MIKSCH, S., SCHUMANN, H., AND TOMINSKI, C. *Visualization of Time-Oriented Data*, sec. ed. Springer London, 2011. A.2

[22] AITTALA, M. Inverse lighting and photorealistic rendering for augmented reality. *The Visual Computer 26*, 6 (2010), 669–678. 2.3, E.2

[23] AKIBA, H., FOUT, N., AND MA, K.-L. Simultaneous classification of time-varying volume data based on the time histogram. In *Proceedings of EuroVis* (2006), pp. 171–178. A.2

[24] AKIBA, H., AND MA, K.-L. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *Proceedings of EuroVis* (2007), pp. 115–122. A.2

[25] AMENT, M., ZIRR, T., AND DACHSBACHER, C. Extinction-optimized volume illumination. *IEEE Transactions on Visualization and Computer Graphics 23*, 7 (2017), 1767–1781. 2.4, 2.3

[26] ANGELELLI, P., NYLUND, K., ODD HILJA, H., AND HAUSER, H. Interactive visual analysis of contrast-enhanced ultrasound data based on small neighborhood statistics. *Computers and Graphics 35*, 2 (2011), 218–226. 2.1, A.2

[27] APODACA, A. A., AND GRITZ, L. *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann, 1999. 2.4, E.2, E.5.3

[28] ATACHIANTS, R., GREGG, D., JARVIS, K., AND DOHERTY, G. Design considerations for parallel performance tools. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2014), pp. 2501–2510. 2.2, B.2

[29] BALABANIAN, J.-P., VIOLA, I., MÖLLER, T., AND GRÖLLER, M. E. Temporal styles for time-varying volume data. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission* (2008), pp. 81–89. A.3.1

[30] Bartesaghi, A., Sapiro, G., Malzbender, T., and Gelb, D. Three-dimensional shape rendering from multiple images. *Graphical Models 67*, 4 (2005), 332 – 346. 2.4, D.2

[31] Bergner, S., Sedlmair, M., Moller, T., Abdolyousefi, S. N., and Saad, A. Paraglide: Interactive parameter space partitioning for computer simulations. *IEEE Transactions on Visualization and Computer Graphics 19*, 9 (2013), 1499–1512. 2.1

[32] Birkeland, Å., Bruckner, S., Brambilla, A., and Viola, I. Illustrative membrane clipping. *Computer Graphics Forum 31*, 3 (2012), 905–914. 2.4, 2.3, C.2

[33] Birkeland, Å., and Viola, I. View-dependent peel-away visualization for volumetric data. In *Proceedings of Spring Conference on Computer Graphics* (2009), pp. 133–139. C.2

[34] Borga, M., Persson, A., Lenz, R., Lindholm, S., and Lathen, G. Automatic tuning of spatially varying transfer functions for blood vessel visualization. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2345–2354. 2.3, E.2

[35] Brodbeck, D., Chalmers, M., Lunzer, A., and Cotture, P. Domesticating bead: adapting an information visualization system to a financial institution. In *Proceedings of IEEE Symposium on Information Visualization* (1997), pp. 73–80. 2.1

[36] Bruckner, S., Grimm, S., Kanitsar, A., and Gröller, M. E. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1559–1569. B.2, C.2

[37] Bruckner, S., and Gröller, M. E. VolumeShop: An interactive system for direct volume illustration. In *Proceedings of IEEE Visualization* (2005), pp. 671–678. B.5, C.4

[38] Bruckner, S., and Gröller, M. E. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1077–1084. B.2, C.2

[39] Bruckner, S., and Gröller, M. E. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum 26*, 3 (2007), 715–724. 4.2, B.6.4

[40] Bruckner, S., and Moller, T. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1468–1476. 2.2

[41] Calinon, S., Epiney, J., and Billard, A. A humanoid robot drawing human portraits. In *Proceedings of International Conference on Humanoid Robots* (2005), pp. 161–166. 2.4, D.2

[42] Cang, R., Ghoniem, M., Kosara, R., Ribarsky, W., and Yang, J. Wirevis: Visualization of categorical, time-varying data from financial transactions. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology* (2007), pp. 155–162. 2.1, A.2

[43] Chang, J., Alain, B., and Ostromoukhov, V. Structure-aware error diffusion. In *Proceedings of ACM SIGGRAPH Asia* (2009), pp. 162:1–162:8. 2.4, D.2

[44] Cherubini, M., Venolia, G., and DeLine, R. Building an ecologically valid, large-scale diagram to help developers stay oriented in their code. In *Proceedings of Visual Languages and Human-Centric Computing* (2007), pp. 157–162. B.2

[45] Chiu, C.-C., Lo, Y.-H., Lee, R.-R., and Chu, H.-K. Tone-and feature-aware circular scribble art. In *Proceedings of Pacific Graphics* (2015), vol. 34, pp. 225–234. 2.4, D.2, D.7

[46] Coffey, D., Korsakov, F., Hagh-Shenas, H., Thorson, L., Ellingson, A., Nuckley, D., and Keefe, D. F. Visualizing motion data in virtual reality: Understanding the roles of animation, interaction, and static presentation. *Computer Graphics Forum 31*, 3pt3 (2012), 1215–1224. E.2

[47] Coffey, D., Lin, C. L., Erdman, A. G., and Keefe, D. F. Design by dragging: An interface for creative forward and inverse design with simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2783–2791. 2.2

[48] Colton, S., Valstar, M. F., and Pantic, M. Emotionally aware automated portrait painting. In *Proceedings of Digital Interactive Media in Entertainment and Arts* (2008), pp. 304–311. 2.4, D.2

[49] Correa, C. D., Silver, D., and Chen, M. Constrained illustrative volume deformation. *Computers & Graphics 34*, 4 (2010), 370–377. B.2

[50] Costa, A. C., de Sousa, A. A., and Ferreira, F. N. Lighting design: A goal based approach using optimisation. In *Rendering Techniques* (1999), pp. 317–328. 2.3, E.2

[51] Crété-Roffet, F., Dolmiere, T., Ladret, P., and Nicolas, M. The Blur Effect: Perception and Estimation with a New No-Reference Perceptual Blur Metric. In *Proceedings of Electronic Imaging Symposium* (2007), pp. 6492–6503. D.3.3

[52] de Melo, C., and Paiva, A. Expression of emotions in virtual humans using lights, shadows, composition and filters. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction* (2007), pp. 546–557. 2.4, E.2

[53] DEUSSEN, O., LINDEMEIER, T., PIRK, S., AND TAUTZENBERGER, M. Feedback-guided stroke placement for a painting machine. In *Proceedings of the First Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging* (2012), pp. 25–33. 2.5, 2.4, D.2

[54] DOERSCHNER, K., FLEMING, R. W., YILMAZ, O., SCHRATER, P. R., HARTUNG, B., AND KERSTEN, D. Visual motion and the perception of surface material. *Current Biology 21*, 23 (2011), 2010–2016. E.1, E.2

[55] DOLEISCH, H. Simvis: Interactive visual analysis of large and time-dependent 3d simulation data. *Winter Simulation Conference* (2007), 712–720. 2.1

[56] DOLEISCH, H., MUIGG, P., AND HAUSER, H. Interactive visual analysis of hurricane isabel with simvis, 2004. 2.1

[57] DRAGICEVIC, P., AND JANSEN, Y. List of physical visualizations. , 2012. Last accessed Feb 2015. 2.2, B.2

[58] DURAND, F., OSTROMOUKHOV, V., MILLER, M., DURANLEAU, F., AND DORSEY, J. Decoupling strokes and high-level attributes for interactive traditional drawing. In *Proceedings of the 12th Eurographics Conference on Rendering* (2001), pp. 71–82. 2.4, D.2

[59] DVOROŽŇÁK, M., LI, W., KIM, V. G., AND SÝKORA, D. ToonSynth: Example-based synthesis of hand-colored cartoon animations. *ACM Transactions on Graphics 37*, 4 (2018). 2.4

[60] EL-NASR, M. S., AND HORSWILL, I. Real-time lighting design for interactive narrative. In *Proceedings of the International Conference on Virtual Storytelling* (2003), pp. 12–20. 2.4, E.2

[61] ELLIS, G., AND DIX, A. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1216–1223. 2.1

[62] ELMQVIST, N., DO, T. N., GOODELL, H., HENRY, N., AND FEKETE, J. D. Zame: Interactive large-scale graph visualization. In *Proceedings of IEEE Pacific Visualization* (2008), pp. 215–222. 2.1, A.2

[63] ELMQVIST, N., AND TSIGAS, P. A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 5 (2008), 1095–1109. C.2

[64] FANG, Z., MÖLLER, T., HAMARNEH, G., AND CELLER, A. Visualization and exploration of time-varying medical image data sets. In *Proceedings of Graphics Interface* (2007), pp. 281–288. 2.1, A.2

[65] FEKETE, J. D., AND PLAISANT, C. Interactive information visualization of a million items. In *Proceedings of IEEE Symposium on Information Visualization* (2002), pp. 117–124. 2.1

[66] FISCHLER, M. A., AND BOLLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (1981), 381–395. 3.2.4, C.3.1

[67] FIŠER, J., JAMRIŠKA, O., LUKÁČ, M., SHECHTMAN, E., ASENTE, P., LU, J., AND SÝKORA, D. StyLit: Illumination-guided example-based stylization of 3d renderings. *ACM Transactions on Graphics 35*, 4 (2016). 2.4

[68] FIŠER, J., JAMRIŠKA, O., SIMONS, D., SHECHTMAN, E., LU, J., ASENTE, P., LUKÁČ, M., AND SÝKORA, D. Example-based synthesis of stylized facial animations. *ACM Transactions on Graphics 36*, 4 (2017). 2.5, 2.4

[69] FORLINES, C., AND WITTENBURG, K. Wakame: Sense making of multi-dimensional spatial- temporal data. In *Proceedings of the International Conference on Advanced Visual Interfaces* (2010), pp. 33–40. A.2

[70] FRASER, N. Rearrangeable wooden model of brain scan. www.dataphys.org/list, 2008. Last accessed Jan 2013. B.2

[71] FREITAG, S., WEYERS, B., AND KUHLEN, T. W. Automatic speed adjustment for travel through immersive virtual environments based on viewpoint quality. In *Proceedings of IEEE Symposium on 3D User Interfaces* (2016), pp. 67–70. 2.3, E.2

[72] FROESE, M. E., TORY, M., EVANS, G. W., AND SHRIKHANDE, K. Evaluation of static and dynamic visualization training approaches for users with different spatial abilities. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2810–2817. E.2

[73] FUA, Y.-H., WARD, M. O., AND RUNDENSTEINER, E. A. Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of IEEE Visualization* (1999), pp. 43–50. 2.1

[74] GAGVANI, N., KENCHAMMANA-HOSEKOTE, D., AND SILVER, D. Volume animation using the skeleton tree. In *Proceedings of IEEE Symposium on Volume Visualization* (1998), pp. 47–53. C.2

[75] GALEA, B., KIA, E., AIRD, N., AND KRY, P. G. Stippling with aerial robots. In *Proceedings of the Joint Symposium on Computational Aesthetics, Sketch Based Interfaces, Modeling, Non-Photorealistic Animation and Rendering* (2016), pp. 125–134. 2.4, D.2

[76] GERL, M., RAUTEK, P., ISENBERG, T., AND GRÖLLER, E. Semantics by analogy for illustrative volume visualization. *Computers & Graphics 36*, 3 (2012), 201 – 213. 2.2, C.2

[77] GUARNIERI, M. The roots of automation before mechatronics [historical]. *IEEE Industrial Electronics Magazine 4*, 2 (2010), 42–43. 2.3

[78] GUMHOLD, S. Maximum entropy light source placement. In *Visualization* (2002), pp. 275–282. 2.3, E.2

[79] GÜNTHER, T., THEISEL, H., AND GROSS, M. Decoupled opacity optimization for points, lines and surfaces. *Computer Graphics Forum 36*, 2 (2017), 153–162. 2.3, E.2

[80] GUO, H., MAO, N., AND YUAN, X. WYSIWYG (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2106–2114. C.2

[81] HÄGERSTRAND, T. What about people in regional science? *Papers of the Regional Science Association 24*, 1 (1970), 7–21. A.2

[82] HALLE, M., AND MENG, J. Lightkit: A lighting system for effective visualization. In *Visualization* (2003), pp. 48–57. 2.3, 3.3.3, E.2

[83] HALLER, M., DRAB, S., AND HARTMANN, W. A real-time shadow approach for an augmented reality application using shadow volumes. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (2003), pp. 56–65. 2.3, E.2

[84] HAUSER, H., KOSARA, R., AND MIKSCH, S. Focus+context taken literally. *IEEE Computer Graphics and Applications 22* (2002), 22–29. 2.1

[85] HAUSER, H., LEDERMANN, F., AND DOLEISCH, H. Angular brushing of extended parallel coordinates. In *Proceedings of IEEE Symposium on Information Visualization* (2002), pp. 127–130. 2.1

[86] HAUSER, H., MROZ, L., BISCHI, G. I., AND GRÖLLER, M. E. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (2001), 242–252. C.2

[87] HELFAND, J. *Reinventing the Wheel*. Princeton Architectural Press, 2006. 3.2.1, B.1

[88] HILLER, S., HELLWIG, H., AND DEUSSEN, O. Beyond Stippling - Methods for Distributing Objects on the Plane. *Computer Graphics Forum* (2003), 515–522. 2.4, D.2

[89] HINCKLEY, K., PAUSCH, R., GOBLE, J. C., AND KASSELL, N. F. Passive real-world interface props for neurosurgical visualization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (1994), pp. 452–458. 2.2

[90] HOGAN, B. A., AND JUAN WELLMAN, B. Visualizing personal networks: Working with participant-aided sociograms. *Field Methods 19*, 2 (2007), 116–144. B.2

[91] HOLMAN, D., GIROUARD, A., BENKO, H., AND VERTEGAAL, R. The design of organic user interfaces: Shape, sketching and hypercontext. *Interacting with Computers 25* (2013), 133–142. 2.2

[92] HOLMAN, D., HOLLATZ, A., BANERJEE, A., AND VERTEGAAL, R. Unifone: Designing for auxiliary finger input in one-handed mobile interactions. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction* (2013), pp. 177–184. 2.2

[93] HOLMAN, D., VERTEGAAL, R., ALTOSAAR, M., TROJE, N. F., AND JOHNS, D. Paper windows: Interaction techniques for digital paper. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2005), pp. 591–599. 2.2, B.2

[94] HSU, W.-H., MEI, J., CORREA, C., AND MA, K.-L. Depicting time evolving flow with illustrative visualization techniques. In *Proceedings of Arts and Technology* (2010), pp. 136–147. A.2

[95] HUNTER, F., BIVER, S., AND FUQUA, P. *Light Science & Magic: An Introduction to Photographic Lighting*. Focal Press, 2015. E.3.5

[96] HURON, S., JANSEN, Y., AND CARPENDALE, S. Constructing visual representations: Investigating the use of tangible tokens. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2102–2111. 2.2, B.2

[97] ISSARTEL, P., GUÉNIAT, F., AND AMMI, M. A portable interface for tangible exploration of volumetric data. In *Proceedings of Virtual Reality Software and Technology* (2014), pp. 209–210. B.2

[98] JACKSON, B., LAU, T. Y., SCHROEDER, D., TOUSSAINT, K. C., AND KEEFE, D. F. A lightweight tangible 3D interface for interactive visualization of thin fiber structures. *Proceedings of IEEE Visualization 19*, 12 (2013), 2802–2809. 2.3, 2.2, B.2

[99] JAIN, S., GUPTA, P., KUMAR, V., AND SHARMA, K. A force-controlled portrait drawing robot. In *Proceedings of Industrial Technology* (2015), pp. 3160–3165. 2.4, D.2

[100] JÄNICKE, H., BÖTTINGER, M., AND SCHEUERMANN, G. Brushing of attribute clouds for the visualization of multivariate data. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1459–1466. 2.1

[101] JANKUN-KELLY, T. J., AND MA, K.-L. A study of transfer functions generation for time-varying volume data. In *Proceedings of Volume Graphics* (2001), pp. 51–68. A.2

[102] JANKUN-KELLY, T. J., MA, K.-L., AND GERTZ, M. A model and framework for visualization exploration. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 357–369. B.2

[103] JANSEN, Y., AND DRAGICEVIC, P. An interaction model for visualizations beyond the desktop. *Proceedings of IEEE Visualization 19*, 12 (2013), 2396–2405. 2.2, B.2

[104] JANSEN, Y., DRAGICEVIC, P., AND FEKETE, J.-D. Evaluating the efficiency of physical visualizations. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2013), pp. 2593–2602. 2.2, B.2

[105] JAVED, W., AND ELMQVIST, N. Exploring the design space of composite visualization. In *Proceedings of IEEE Pacific Visualization* (2012), pp. 1–8. 2.1, A.2

[106] JESCHKE, S. Generalized diffusion curves: An improved vector representation for smooth-shaded images. *Comput. Graph. Forum 35*, 2 (2016), 71–79. D.2

[107] JOHANNES, K., PETER, F., AND HELWIG, H. Brushing moments in interactive visual analysis. *Computer Graphics Forum 29*, 3 (2010), 813–822. 2.1

[108] JOHANSSON, J., LJUNG, P., JERN, M., AND COOPER, M. Revealing structure within clustered parallel coordinates displays. In *Proceedings of IEEE Symposium on Information Visualization* (2005), pp. 125–132. 2.1

[109] JOHANSSON, J., LJUNG, P., JERN, M., AND COOPER, M. Revealing structure in visualizations of dense 2d and 3d parallel coordinates. *Information Visualization 5*, 2 (2006), 125–136. 2.1

[110] JÖNSSON, D., FALK, M., AND YNNERMAN, A. Intuitive exploration of volumetric data using dynamic galleries. *Proceedings of IEEE Visualization 22*, 1 (2016), 896–905. B.1

[111] JOUBERT, N., ROBERTS, M., TRUONG, A., BERTHOUZOZ, F., AND HANRAHAN, P. An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics (SIGGRAPH Asia 2015) 34*, 6 (2015), 238:1–238:11. E.2

[112] JUN, Y., JANG, G., CHO, B., TRUBATCH, J., KIM, I., SEO, S., AND OH, P. Y. A humanoid doing an artistic work - graffiti on the wall. In *Proceedings of Intelligent Robots and Systems* (2016), pp. 1538–1543. 2.4, D.2

[113] KALLWEIT, S., MÜLLER, T., MCWILLIAMS, B., GROSS, M., AND NOVÁK, J. Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 231:1–231:11. 2.4

[114] KAPLAN, C. S., AND BOSCH, R. Tsp art. In *Proceedings of Bridges Conference Renaissance Banff: Mathematics, Music, Art, Culture* (2005), pp. 301–308. 2.4, D.2

[115] KASS, M., WITKIN, A., AND TERZOPOULOS, D. Snakes: Active contour models. *International Journal of Computer Vision 1*, 4 (1988), 321–331. E.3.2

[116] KEISER, B. E., AND PEEBLES, P. Z. An automatic system for the control of multiple drone aircraft. *IEEE Transactions on Aerospace and Electronic Systems AES-5*, 3 (1969), 515–524. E.2

[117] KERAMAT, M., AND KIELBASA, R. Latin hypercube sampling monte carlo estimation of average quality index for integrated circuits. *Analog Integrated Circuits and Signal Processing 14*, 1 (1997), 131–142. 3.3.1, E.3.3

[118] KERSTEN, D., MAMASSIAN, P., AND KNILL, D. C. Moving cast shadows induce apparent motion in depth. *Perception 26*, 2 (1997), 171–192. E.2, E.3.2, E.3.2

[119] KHALIFA, F., SOLIMAN, A., EL-BAZ, A., ABOU EL-GHAR, M., EL-DIASTY, T., AND GIMEL'FARB, G. Models and methods for analyzing DCE-MRI: a review. *Medical Physics 41*, 12 (2014). A.1

[120] KHALILBEIGI, M., LISSERMANN, R., MÜHLHÄUSER, M., AND STEIMLE, J. Xpaaand: Interaction techniques for rollable displays. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2011), pp. 2729–2732. 2.2

[121] KLEFFNER, D. A., AND RAMACHANDRAN, V. S. On the perception of shape from shading. *Perception & Psychophysics 52*, 1 (1992), 18–36. E.2

[122] KLEIN, G., AND MURRAY, D. Compositing for small cameras. In *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality* (2008), pp. 57–60. 2.3, E.2

[123] KNISS, J., KINDLMANN, G., AND HANSEN, C. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of IEEE Visualization* (2001), pp. 255–262. C.2

[124] KÖNIG, A., AND GRÖLLER, M. E. Mastering transfer function specification by using volumepro technology. Tech. Rep. TR-186-2-00-07, Vienna University of Technology, 2000. 2.2, B.2

[125] KONYHA, Z., MATKOVIC, K., GRACANIN, D., JELOVIC, M., AND HAUSER, H. Interactive visual analysis of families of function graphs. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1373–1385. 2.1

[126] KRAAK, M. J. The space-time cube revisited from a geovisualization perspective. In *Proceedings of the International Cartographic Conference* (2003), pp. 1988–1995. A.2

[127] KWON, B. C., JAVED, W., ELMQVIST, N., AND YI, J. S. Direct manipulation through surrogate objects. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2011), pp. 627–636. 2.2, 3.2.3, C.1, C.2

[128] KYPRIANIDIS, J. E., COLLOMOSSE, J., WANG, T., AND ISENBERG, T. State of the art: A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics 19*, 5 (2013), 866–885. 2.4, D.2

[129] LAWONN, K., GLASSER, S., VILANOVA, A., PREIM, B., AND ISENBERG, T. Occlusion-free blood flow animation with wall thickness visualization. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 728–737. C.2

[130] LE GOC, M., DRAGICEVIC, P., HURON, S., BOY, J., AND FEKETE, J.-D. SmartTokens: Embedding motion and grip sensing in small tangible objects. In *Proceedings of User Interface Software and Technology* (2015), pp. 357–362. 2.2, B.2

[131] LEE, C. H., HAO, X., AND VARSHNEY, A. Light collages: lighting design for effective visualization. In *Visualization* (2004), Proceedings of IEEE Visualization, pp. 281–288. 2.3, 3.3.3, E.2

[132] LEE, T.-Y., CHAUDHURI, A., POIKLI, F., AND SHEN, H.-W. Cyclestack: Inferring periodic behavior via temporal sequence visualization in ultrasound video. In *Proceedings of IEEE Pacific Visualization* (2010), pp. 89–96. A.2

[133] LEE, T.-Y., AND SHEN, H.-W. Visualization and exploration of temporal trend relationships in multivariate time-varying data. *IEEE TVCG 15*, 6 (2009), 1359–1366. A.2

[134] LEE, T.-Y., AND SHEN, H.-W. Visualizing time-varying features with tac-based distance fields. In *Proceedings of IEEE Pacific Visualization* (2009), pp. 1–8. A.2

[135] LI, W., RITTER, L., AGRAWALA, M., CURLESS, B., AND SALESIN, D. Interactive cutaway illustrations of complex 3D models. *ACM Transactions on Graphics 26*, 3 (2007), 31. C.2

[136] LI, X.-Y., SHEN, C.-H., HUANG, S.-S., JU, T., AND HU, S.-M. Popup: automatic paper architectures from 3D models. *Transactions on Graphics 29*, 4 (2010), 111:1–9. 2.4, B.2

[137] LINDEMEIER, T., PIRK, S., AND DEUSSEN, O. Image stylization with a painting machine using semantic hints. *Comput. Graph. 37*, 5 (2013), 293–301. 2.5, 2.4, D.1, D.2

[138] LINDOW, N., BAUM, D., AND HEGE, H.-C. Perceptually linear parameter variations. *Computer Graphics Forum 31*, 2 (2012), 535–544. B.4.3, B.4.3

[139] LU, A., AND SHEN, H.-W. Interactive storyboard for overall time-varying data visualization. In *Proceedings of IEEE Pacific Visualization* (2008), pp. 143–150. A.2

[140] MA, K.-L. Image graphs - a novel approach to visual data exploration. *Proceedings of IEEE Visualization* (1999), 81–88. 2.2, B.2

[141] MACIEJEWSKI, R., ISENBERG, T., ANDREWS, W., EBERT, D., SOUSA, M., AND CHEN, W. Measuring stipple aesthetics in hand-drawn and computer-generated images. *IEEE Computer Graphics and Applications 28*, 2 (2008), 62–74. D.2

[142] MACKINLAY, J. D., CARD, S. K., AND ROBERTSON, G. G. Rapid controlled movement through a virtual 3d workspace. In *Proceedings of ACM SIGGRAPH* (1990), pp. 171–176. 2.3

[143] MAMASSIAN, P., AND GOUTCHER, R. Prior knowledge on the illumination position. *Cognition 81*, 1 (2001), 1–9. E.2, E.5.2

[144] MARKS, J., ANDALMAN, B., BEARDSLEY, P. A., FREEMAN, W., GIBSON, S., HODGINS, J., KANG, T., MIRTICH, B., PFISTER, H., RUML, W., RYALL, K., SEIMS, J., AND SHIEBER, S. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of ACM SIGGRAPH* (1997), pp. 389–400. 2.2, B.2

[145] MARSHALL, M. W. 'automation' today and in 1662. *American Speech 32*, 2 (1957), 149–151. 2.3

[146] MARTÍN, D., ARROYO, G., RODRÍGUEZ, A., AND ISENBERG, T. A survey of digital stippling. *Computers & Graphics 67* (2017), 24 – 44. D.2

[147] MCGUFFIN, M. J., TANCAU, L., AND BALAKRISHNAN, R. Using deformations for browsing volumetric data. In *Proceedings of IEEE Visualization* (2003), pp. 53–61. 2.2, C.2

[148] MINDEK, P., BRUCKNER, S., RAUTEK, P., AND E.GRÖLLER, M. Visual parameter exploration in gpu shader space. *Journal of WSCG 21*, 3 (2013), 225–234. 2.1

[149] MINDEK, P., MISTELBAUER, G., GRÖLLER, M. E., AND BRUCKNER, S. Data-sensitive visual navigatio. *Computers & Graphics 67*, C (2017), 77–85. 2.2

[150] MLEJNEK, M., ERNEST, P., VILANOVA, A., VAN DEN BOSCH, H., GERRITSEN, F., AND GROLLER, M. E. Profile flags: a novel metaphor for probing of T2 maps. In *Proceedings of IEEE Visualization* (2005), pp. 599–606. 2.1, A.2

[151] MOERE, A. V. Time-varying data visualization using information flocking boids. In *Proceedings of IEEE Symposium on Information Visualization* (2004), pp. 97–104. A.2

[152] MUZIC, M. L., MINDEK, P., SORGER, J., AUTIN, L., GOODSELL, D., AND VIOLA, I. Visibility equalizer: Cutaway visualization of mesoscopic biological models. *Computer Graphics Forum 35*, 3 (2016), 161—-170. 2.3, C.2

[153] NESRINE AKKARI, HENRIK ENGHOFF, B. D. M. A new dimension in documenting new species: High-detail imaging for myriapod taxonomy and first 3D cybertype of a new millipede species (diplopoda, julida, julidae). *PLoS ONE 10*, 8 (2015), e0135243. 4.2, B.6.3

[154] NGUYEN, B. P., TAY, W.-L., CHUI, C.-K., AND ONG, S.-H. A clustering-based system to automate transfer function design for medical image visualization. *The Visual Computer 28*, 2 (2012), 181–191. 2.3

[155] NICHOLSON, D. T., CHALK, C., FUNNELL, W. R. J., AND DANIEL, S. J. Can virtual reality improve anatomy education? a randomised controlled study of a computer-generated three-dimensional anatomical ear model. *Medical Education 40*, 11 (2006), 1081–1087. E.5.2

[156] NIKOLOS, I. K., VALAVANIS, K. P., TSOURVELOUDIS, N. C., AND KOSTARAS, A. N. Evolutionary algorithm based offline/online path planner for uav navigation. *IEEE Transactions on Systems, Man, and Cybernetics) 33*, 6 (2003), 898–912. E.2

[157] NORIS, G., SÝKORA, D., COROS, S., WHITED, B., SIMMONS, M., HORNUNG, A., GROSS, M., AND SUMNER, R. Temporal noise control for sketchy animation. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering* (2011), pp. 93–98. 2.4

[158] NORTON, M. I., MOCHON, D., AND ARIELY, D. The IKEA effect: When labor leads to love. *Journal of Consumer Psychology 22* (2012), 453–460. 3.2.1, B.1

[159] NOVOTNY, M., AND HAUSER, H. Similarity brushing for exploring multidimensional relations. *Journal of WSCG 14* (2006), 105–112. 2.1

[160] OKUMURA, B., KANBARA, M., AND YOKOYA, N. Augmented reality based on estimation of defocusing and motion blurring from captured images. In *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality* (2006), pp. 219–225. 2.3, E.2

[161] OSTROMOUKHOV, V., AND HERSCH, R. D. Multi-color and artistic dithering. In *Proceedings of ACM SIGGRAPH* (1999), pp. 425–432. D.2

[162] OWADA, S., NIELSEN, F., AND IGARASHI, T. Volume catcher. In *Proceedings of Symposium on Interactive 3D Graphics and Games* (2005), pp. 111–116. 2.3, C.2

[163] OWADA, S., NIELSEN, F., IGARASHI, T., HARAGUCHI, R., AND NAKAZAWA, K. Projection plane processing for sketch-based volume segmentation. In *Proceedings of Symposium on Biomedical Imaging* (2008), pp. 117–120. 2.3, C.2

[164] PANG, W.-M., QU, Y., WONG, T.-T., COHEN-OR, D., AND HENG, P.-A. Structure-aware halftoning. *ACM Trans. Graph. 27*, 3 (2008), 1–8. 2.4, D.2

[165] PATEL, D., ŠOLTÉSZOVÁ, V., NORDBOTTEN, J. M., AND BRUCKNER, S. Instant convolution shadows for volumetric detail mapping. *ACM Transactions on Graphics 32*, 5 (2013), 154:1–154:18. C.4

[166] PEDERSEN, H., AND SINGH, K. Organic labyrinths and mazes. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering* (2006), pp. 79–86. 2.4, D.2

[167] PERIN, C., LE GOC, M., DI VOZZO, R., FEKETE, J.-D., AND DRAGICEVIC, P. DIY Bertin Matrix. In *Proceedings of the CHI Workshop on Exploring the Challenges of Making Data Physical* (2015). 2.2

[168] Piringer, H., Berger, W., and Krasser, J. Hypermoval: Interactive visual validation of regression models for real-time simulation. *Comput. Graph. Forum 29* (2010), 983–992. 2.1

[169] Plaisant, C., Milash, B., Rose, A., Widoff, S., and Shneiderman, B. Lifelines: Visualizing personal histories. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (1996), pp. 221–227. 2.1

[170] Pnueli, Y., and Bruckstein, A. M. Gridless halftoning: A reincarnation of the old method. *Graphical Models and Image Processing 58*, 1 (1996), 38 – 64. 2.4, D.2

[171] Prévost, R., Jacobson, A., Jarosz, W., and Sorkine-Hornung, O. Large-scale painting of photographs by interactive optimization. *Computers & Graphics 55*, 0097-8493 (2016), 108 – 117. 2.4, D.2

[172] Ramachandran, V. S. Perception of shape from shading. *Nature 331*, 6152 (1988), 163–166. E.2

[173] Rautek, P., Bruckner, S., Gröller, E., and Viola, I. Illustrative visualization: New technology or useless tautology? *SIGGRAPH Computer Graphics 42*, 3 (2008), 4:1–4:8. B.2

[174] Rezk-Salama, C., Keller, M., and Kohlmann, P. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1021–1028. B.2

[175] Richards, A. *How to Set Up Photography Lighting for a Home Studio*. CreateSpace, 2014. E.3.5

[176] Roberts, J. C. State of the art: Coordinated and multiple views in exploratory visualization. In *Proceedings of International Conference on Coordinated and Multiple Views in Exploratory Visualization* (2007), pp. 61–71. 2.1, A.2

[177] Ropinski, T., Prassni, J., Steinicke, F., and Hinrichs, K. Stroke-based transfer function design. In *Proceedings of Eurographics/IEEE VGTC Conference on Point-Based Graphics* (2008), pp. 41–48. C.2

[178] Ruder, S. An overview of gradient descent optimization algorithms. *CoRR abs/1609.04747* (2016). E.3.3

[179] Ruiz, M., Bardera, A., Boada, I., Viola, I., Feixas, M., and Sbert, M. Automatic transfer functions based on informational divergence. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1932–1941. 2.3, E.2

[180] Russell, J., and Cohn, R. *Volvelle*. Book on Demand, 2015. 3.2.1, B.1

[181] Rusu, R. B., and Cousins, S. 3D is here: Point Cloud Library (PCL). In *Proceedings of IEEE International Conference on Robotics and Automation* (2011), pp. 1–4. C.4

[182] S., R., R., S., K., M., M., D., E., G., AND H., H. Towards quantitative visual analytics with structured brushing and linked statistics. *Computer Graphics Forum 35*, 3 (2016), 251–260. 2.1, 2.1

[183] SCHEIDEGGER, C., VO, H., KOOP, D., FREIRE, J., AND SILVA, C. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1560–1567. B.2

[184] SCHÜTT, H. H., BAIER, F., AND FLEMING, R. W. Perception of light source distance from shading patterns. *Journal of Vision 16*, 3 (2016), 9. E.1, E.2

[185] SEDLMAIR, M., HEINZL, C., BRUCKNER, S., PIRINGER, H., AND MÖLLER, T. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2161–2170. 2.1

[186] SELIM, A., ELGHARIB, M., AND DOYLE, L. Painting style transfer for head portraits using convolutional neural networks. *ACM Trans. Graph. 35*, 4 (2016), 1–18. 2.4

[187] SHACKED, R., AND LISCHINSKI, D. Automatic Lighting Design using a Perceptual Quality Metric. *Computer Graphics Forum 20*, 3 (2001), 215–227. 2.3, E.2

[188] SHEARER, J., OGAWA, M., MA, K.-L., AND KOHLENBERG, T. Pixelplexing: Gaining display resolution through time. In *Proceedings of IEEE Pacific Visualization* (2008), pp. 159–266. 2.1, A.2

[189] SHIELDS, M. D., AND ZHANG, J. The generalization of latin hypercube sampling. *Reliability Engineering and System Safety 148*, 1 (2016), 96–108. 3.3.1, E.3.3

[190] SHNEIDERMAN, B. Direct manipulation: A step beyond programming languages. *Computer 16*, 8 (1983), 57–69. 2.2, 3.2.3, C.1, C.2

[191] SHUGRINA, M., BETKE, M., AND COLLOMOSSE, J. Empathic painting: Interactive stylization through observed emotional state. In *Proceedings of Symposium on Non-photorealistic Animation and Rendering* (2006), pp. 87–96. 2.4, D.2

[192] SON, M., LEE, Y., KANG, H., AND LEE, S. Structure grid for directional stippling. *Graphical Models 73*, 3 (2011), 74 – 87. 2.4, D.2

[193] SPICKER, M., HAHN, F., LINDEMEIER, T., SAUPE, D., AND DEUSSEN, O. Quantifying visual abstraction for stipple drawings. In *Proceedings of Symposium on Computational Aesthetics, Sketch-Based Interfaces, Modeling, Non-Photorealistic Animation and Rendering* (2017), pp. 1–10. D.2

[194] SPINDLER, M., STELLMACH, S., AND DACHSELT, R. PaperLens: Advanced magic lens interaction above the tabletop. In *Proceedings of Intelligent Tutoring Systems* (2009), pp. 69–76. 2.2, B.2

[195] SPINDLER, M., TOMINSKI, C., SCHUMANN, H., AND DACHSELT, R. Tangible views for information visualization. In *Proceedings of Intelligent Tutoring Systems* (2010), pp. 157–166. 2.2, B.2

[196] SRIKANTH, M., BALA, K., AND DURAND, F. Computational rim illumination with aerial robots. In *Proceedings of the Workshop on Computational Aesthetics* (2014), pp. 57–66. E.2

[197] STEIN, M. Large sample properties of simulations using latin hypercube sampling. *Technometrics 29*, 2 (1987), 143–151. E.3.3

[198] STONE, M. C., FISHKIN, K., AND BIER, E. A. The movable filter as a user interface tool. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (1994), pp. 306–312. 2.1

[199] STOPPEL, S., HEGE, H.-C., AND WIEBEL, A. Visibility-Driven Depth Determination of Surface Patches in Direct Volume Rendering. In *Proceedings of EuroVis Short Papers* (2014), pp. 97–101. 2.3, C.2

[200] STUSAK, S., AND ASLAN, A. Beyond physical bar charts: An exploration of designing physical visualizations. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2014), pp. 1381–1386. 2.2, B.2

[201] STUSAK, S., AND BUTZ, A. Can physical visualizations support analytical tasks? In *Proceedings of IEEE Visualization, Posters* (2013). 2.2, B.2

[202] STUSAK, S., HOBE, M., AND BUTZ, A. If your mind can grasp it, your hands will help. In *Proceedings of Tangible, Embedded and Emodied Interactions* (2016), pp. 92–99. 3.2.1, B.1

[203] STUSAK, S., SCHWARZ, J., AND BUTZ, A. Evaluating the memorability of physical visualizations. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2015), pp. 3247–3250. 2.2, 3.2.1, B.1, B.2

[204] STUSAK, S., TABARD, A., SAUKA, F., KHOT, R. A., AND BUTZ, A. Activity sculptures: Exploring the impact of physical visualizations on running activity. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2201–2210. 2.3, 2.2, B.2

[205] SULLIVAN, W. The new york times archives, page 53, June 7, 1964. 3.1

[206] SWAMINATHAN, S., SHI, C., JANSEN, Y., DRAGICEVIC, P., OEHLBERG, L., AND FEKETE, J.-D. Creating physical visualizations with MakerVis. In *Proceedings of the ACM Conference on Human Factors in Computing Systems, Extended Abstracts* (2014), pp. 543–546. 2.2, B.2

[207] TAHER, F., HARDY, J., KARNIK, A., WEICHEL, C., JANSEN, Y., HORNBÆK, K., AND ALEXANDER, J. Exploring interactions with physically dynamic bar charts. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2015), pp. 3237–3246. 2.2, B.2

[208] TANI, Y., ARAKI, K., NAGAI, T., KOIDA, K., NAKAUCHI, S., AND KITAZAKI, M. Enhancement of glossiness perception by retinal-image motion: Additional effect of head-yoked motion parallax. *PLOS ONE 8*, 1 (2013), 1–8. E.1

[209] TATZGERN, M., KALKOFEN, D., GRASSET, R., AND SCHMALSTIEG, D. Hedgehog labeling: View management techniques for external labels in 3d space. *2014 IEEE Virtual Reality (VR)* (2014), 27–32. 2.1

[210] TAYLOR, S., KIM, T., YUE, Y., MAHLER, M., KRAHE, J., RODRIGUEZ, A. G., HODGINS, J., AND MATTHEWS, I. A deep learning approach for generalized speech animation. *ACM Trans. Graph. 36*, 4 (2017), 93:1–93:11. 2.4

[211] TIKHONOVA, A., CORREA, C., AND MA, K.-L. Explorable images for visualizing volume data. In *Proceedings of IEEE Pacific Visualization* (2010), pp. 177–184. B.2, B.4.1

[212] TIKHONOVA, A., CORREA, C., AND MA, K.-L. An exploratory technique for coherent visualization of time-varying volume data. *CGF 29*, 3 (2010), 783–792. A.2

[213] TOMINSKI, C., GLADISCH, S., KISTER, U., DACHSELT, R., AND SCHUMANN, H. Interactive lenses for visualization: An extended survey. *Computer Graphics Forum* (2016), 173–200. C.2

[214] TRESSET, P., AND FOL LEYMARIE, F. Portrait drawing by paul the robot. *Comput. Graph. 37*, 5 (2013), 348–363. 2.4, D.2, D.3.2

[215] TUFTE, E. R. *The Visual Display of Quantitative Information*, second ed. Graphics Press Cheshire, 2001. 3.1, A.3, A.6

[216] VAN DER MAATEN, L., AND HINTON, G. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research 9* (2008), 2579–2605. 1

[217] VAN LAARHOVEN, P. J. M., AND AARTS, E. H. L. *Simulated annealing*. Springer Netherlands, 1987, pp. 7–15. E.3.3

[218] VIOLA, I., FEIXAS, M., SBERT, M., AND GRÖLLER, M. E. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 933–940. C.2

[219] VIOLA, I., AND GRÖLLER, M. E. Smart visibility in visualization. In *Proceedings of the First Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging* (2005), pp. 209–216. 2.3, C.2

[220] VIOLA, I., KANITSAR, A., AND GRÖLLER, M. E. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics 11*, 4 (2005), 408–418. B.2, C.2

[221] ŠEREDA, P., VILANOVA, A., AND GERRITSEN, F. A. Automating transfer function design for volume rendering using hierarchical clustering of material boundaries. In *Proceedings of EuroVis* (2006), pp. 243–250. C.2

[222] WAMBECKE, J., VERGNE, R., BONNEAU, G.-P., AND THOLLOT, J. Automatic lighting design from photographic rules. In *WICED: Eurographics Workshop on Intelligent Cinematography and Editing* (2016), pp. 1–8. 2.3, 3.3.3, 4.5, E.2, E.5.4

[223] WANG, C., YU, H., AND MA, K.-L. Importance-driven time-varying data visualization. *IEEE TVCG 14*, 6 (2008), 1547–1554. A.2

[224] WANG, C., YU, H., AND MA, K.-L. Application-driven compression for visualizing large-scale time-varying data. *IEEE Computer Graphics and Applications 30*, 1 (2010), 59–69. A.2

[225] WANG, L., AND KAUFMAN, A. E. Lighting system for visual perception enhancement in volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 19 (2013), 67–80. 2.3, 3.3.3, E.2

[226] WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (2004), 600–612. B.4.3

[227] WASER, J., FUCHS, R., RIBICIC, H., SCHINDLER, B., BLOSCHL, G., AND GROLLER, E. World lines. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1458–1467. 2.2

[228] WEISKOPF, D., ENGEL, K., AND ERTL, T. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (2003), 298–312. C.2

[229] WEN, F., LUAN, Q., LIANG, L., XU, Y.-Q., AND SHUM, H.-Y. Color sketch generation. In *Proceedings of Symposium on Non-photorealistic Animation and Rendering* (2006), pp. 47–54. 2.4, D.2

[230] WIEBEL, A., PREIS, P., VOS, F. M., AND HEGE, H.-C. 3D strokes on visible structures in direct volume rendering. In *Proceedings of EuroVis Short Papers* (2013), pp. 91–95. 2.3, C.2

[231] WIEBEL, A., VOS, F. M., FOERSTER, D., AND HEGE, H. C. WYSIWYP: What you see is what you pick. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2236–2244. 2.3, C.2, C.3.1

[232] WISESSING, P., DINGLIANA, J., AND MCDONNELL, R. Perception of lighting and shading for animated virtual characters. In *Proceedings of the ACM Symposium on Applied Perception* (2016), pp. 25–29. 2.4, E.2

[233] WOODRING, J., AND SHEN, H.-W. Chronovolumes: a direct rendering technique for visualizing time-varying data. In *Proceedings of Volume Graphics* (2003), pp. 27–34. A.2

[234] WOODRING, J., AND SHEN, H.-W. Multi-variate, time varying, and comparative visualization with contextual cues. *IEEE TVCG 12*, 5 (2006), 909–916. A.2

[235] WOODRING, J., AND SHEN, H.-W. Multiscale time activity data exploration via temporal clustering visualization spreadsheet. *IEEE TVCG 15*, 2 (2009), 123–137. A.2

[236] WOODRING, J., AND SHEN, H.-W. Semi-automatic time-series transfer functions via temporal clustering and sequencing. *CGF 28*, 3 (2009), 791–198. A.2

[237] WOODRING, J., WANG, S., AND SHEN, H.-W. High dimensional direct rendering of time-varying volumetric data. In *Proceedings of IEEE Visualization* (2003), pp. 417–424. A.2

[238] XIE, J., ZHOU, Y., WU, W., AND ZHOU, Z. Automatic path planning for augmented virtual environment. In *Proceedings of the International Conference on Virtual Reality and Visualization* (2016), pp. 372–379. 2.3, E.2

[239] XU, J., AND KAPLAN, C. S. Image-guided maze construction. In *Proceedings of ACM SIGGRAPH* (2007). 2.4, D.2

[240] XU, J., KAPLAN, C. S., AND MI, X. Computer-generated papercutting. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (2007), pp. 343–350. 2.4, D.2

[241] YANG, M., LIU, Z., AND LI, W. A fast general extension algorithm of latin hypercube sampling. *Journal of Statistical Computation and Simulation 87*, 17 (2017), 3398–3411. 3.3.1, E.3.3

[242] YEH, R. B., BRANDT, J., BOLI, J., AND KLEMMER, S. R. Interactive gigapixel prints: Large paper interfaces for visual context, mobility, and collaboration. Tech. rep., Stanford University, 2006. B.2

[243] YNNERMAN, A., RYDELL, T., PERSSON, A., ERNVIK, A., FORSELL, C., LJUNG, P., AND LUNDSTRÖM, C. Multi-Touch Table System for Medical Visualization. In *Proceedings of EUROGRAPHICS* (2015), pp. 1775–1784. C.1

[244] YU, L., EFSTATHIOU, K., ISENBERG, P., AND ISENBERG, T. CAST: Effective and Efficient User Interaction for Context-Aware Selection in 3D Particle Clouds. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 886–895. 2.3, C.2

[245] YU, L., SVETACHOV, P., ISENBERG, P., EVERTS, M. H., AND ISENBERG, T. FI3D: Direct-touch interaction for the exploration of 3d scientific visualization spaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1613–1622. 2.2, C.2

[246] ZACARIAS, M., AND OLIVEIRA, J. *Exploratory Data Analysis*, vol. 1. 1977. 2.1

[247] ZACARIAS, M., AND OLIVEIRA, J. *Human-Computer Interaction: The Agency Perspective*, vol. 396. 01 2012. 1

[248] ZHANG, L., TANG, C., SONG, Y., ZHANG, A., AND RAMANATHAN, M. Viz-cluster and its application on classifying gene expression data. *Distributed and Parallel Databases 13*, 1 (2003), 73–97. 2.1

[249] ZHANG, Y., AND MA, K.-L. Lighting design for globally illuminated volume rendering. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2946–2955. 2.3, E.2