3D Visualization of Multivariate Data

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

Vorgelegt von

Harald Sanftmann

aus Temeschburg

Hauptberichter:	Prof. Dr. D. Weiskopf
Mitberichter:	Prof. Dr. H. Hauser
Tag der mündlichen Prüfung:	23.05.2012

Visualisierungsinstitut
der Universität Stuttgart

2012

ACKNOWLEDGMENTS

During my years in Stuttgart working at the Visualization Research Center I had the privilege to meet many people who I would like to thank. They all made this time a very nice part of my life.

I am most grateful to my adviser Daniel Weiskopf, who always took the time to discuss and guide my work. I thank Thomas Ertl for offering me the position at the Visualization Research Center and giving me the opportunity to work on my thesis. Helwig Hauser from the University of Bergen gets my thanks for examining my thesis and his helpful expertise.

The German Research Foundation (DFG) and the state of Baden-Württemberg I thank for funding my work. I thank my roommate Mike Eißele for introducing me to Nexus. In the DFG Collaborative Research Center Nexus I thank Nazario Cipriani, Carlos Lübbe, Harald Weinschroth, and Julia Möhrmann for the collaboration on the visualization-pipeline demonstrator program. Special thanks go to André Blessing, Julia Möhrmann, and Andre Burkovski for providing interesting visualization tasks, data sets, and participating in expert studies. I also thank the students of our university for participating in the user study on 3D scatter plot navigation. I appreciate the work Mikael Vaaraniemi, Daniel Kauker, and Ilona Heurich have done in their diploma theses I supervised.

I thank all the members of the "Cafete" team for the various discussions, especially Benjamin Höferlin, Markus Höferlin, Michael Wörner, Martin Falk, Markus Üffinger, and Steffen Müthing who even tried to teach me table football, with marginal success. Many thanks go to Martin Falk, Markus Höferlin, and Corinna Vehlow for proof reading my thesis. I thank NVIDIA for producing such good GPUs and for raffling 3D cameras which made me start working on eliminating ghosting artifacts in anaglyph stereo images to enjoy my holiday pictures.

Finally, I thank my parents and grandparents for supporting my education as well as my wife and my son for giving me so much support.

Eching, June 2012

Harald Sanftmann

CONTENTS

List of Abbreviations and Acronyms

aka.	also known as
API	application programming interface
bit	binary digit
byte	eight bits
CIE	Commission internationale de l'éclairage
CPU	central processing unit
CRT	cathode ray tube
fMRI	functional magnetic resonance imaging
e.g.	exempli gratia
et al.	et alii
FIFO	first-in, first-out
fps	frames per second
FSAA	full-scene anti-aliasing
GB	gigabyte
GLSL	OpenGL Shading Language
GPU	graphics processing unit
GUI	graphical user interface
i.e.	id est
LCD	liquid crystal display
LIC	line integral convolution
LOC	lateral occipital complex
Ly	langley
MDA	Multidimensional Analyzer
ML	machine learning
mph	miles per hour
MPI	Message Passing Interface
MSAA	multisample anti-aliasing
*n*D	*n*-dimensional
NLP	natural language processing
n-space	*n*-dimensional space
OpenGL	Open Graphics Library
OSG	OpenSceneGraph
PC	personal computer
PDA	personal digital assistant
PET	positron emission tomography
PLC	proximity-luminance covariance
pixel	picture element
ppb	parts-per-billion
RGB	red, green, blue
SFM	structure from motion
SOM	self-organizing map
SVM	support vector machine
TCP/IP	Transmission Control Protocol/ Internet Protocol
TFT	thin-film transistor
V1	primary visual cortex
VA	visual analytics
VBO	vertex buffer object
VTK	Visualization Toolkit
VR	virtual reality
vs.	versus

ABSTRACT

Nowadays large amounts of data are organized in tables, especially in relational databases where the rows store the data items to which multiple attributes are stored in the columns. Information stored this way, having multiple (more than two or three) attributes, can be treated as multivariate data. Therefore, visualization methods for multivariate data have a large application area and high potential utility.

This thesis focuses on the application of 3D scatter plots for the visualization of multivariate data. When dealing with 3D, spatial perception needs to be exploited, by effectively using depth cues to convey spatial information to the user. To improve the presentation of individual 3D scatter plots, a technique is presented that applies illumination to them, thus using the shape-from-shading depth cue. To enable the analysis not only of 3D but of multivariate data, a novel technique is introduced that allows the navigation between 3D scatter plots. Inspecting the large number of 3D scatter plots that can be projected from a multivariate data set is very time consuming. The analysis of multivariate data can benefit from automatic machine learning approaches. A presented method uses decision trees to increase the speed a user can gain an understanding of the multivariate data at no extra cost. Stereopsis can also support the display of 3D scatter plots. Here an improved anaglyph rendering technique is presented, significantly reducing ghosting artifacts. The technique is not only applicable for information visualization, but for general rendering or to present stereoscopic image data. Some information visualization algorithms require high computation time. Many of these algorithms can be parallelized to run interactively. A framework that supports the parallelization on shared and distributed memory systems is presented.

KURZBESCHREIBUNG

In der heutigen Zeit werden große Datenmengen in Tabellen gespeichert, vor allem in relationalen Datenbanken. Dort speichern die Zeilen die Datensätze und die Spalten unterschiedliche Datenattribute. Informationen, die so gespeichert sind, können als multivariate Daten behandelt werden. Daher haben Visualisierungsmethoden für multivariate Daten ein breites Anwendungsgebiet und hohen potentiellen Nutzen.

Diese Arbeit hat die Anwendung von 3D Streudiagrammen zur Visualisierung multivariater Daten zum Fokus. Wenn dreidimensionale Objekte verwendet werden, muss die dreidimensionale Struktur durch den effektiven Einsatz von Hinweisreizen für das Tiefensehen (engl. depth cues) dem Betrachter vermittelt werden. Um die Darstellung einzelner 3D Streudiagramme zu verbessern, wird ein Verfahren vorgestellt, welches Beleuchtung auf diese anwendet und dadurch den gleichnamigen Hinweisreiz verwendet. Um die Analyse von nicht nur dreidimensionalen, sondern auch multivariaten Daten zu ermöglichen, wird ein neues Verfahren vorgestellt, welches die Navigation zwischen 3D Streudiagrammen ermöglicht. Eine große Anzahl von 3D Streudiagrammen zu untersuchen ist sehr zeitraubend. Die Analyse multivariater Daten kann jedoch von automatischen maschinellen Lernverfahren profitieren. Ein vorgestelltes Verfahren verwendet Entscheidungsbäume, um die Geschwindigkeit zu erhöhen, mit der ein Benutzer multivariate Daten verstehen kann, ohne einen zeitlichen Mehraufwand auf der Seite des Benutzers zu verursachen. Stereopsis kann auch die Anzeige von 3D Streudiagrammen unterstützen. Hier wird ein verbessertes Verfahren zur Erzeugung von Anaglyph-Stereo-Darstellungen vorgestellt, welches Geisterbildartefakte signifikant reduziert. Das Verfahren ist nicht nur in der Informationsvisualisierung, sondern auch generell für Rendering einsetzbar und eignet sich sogar zur Darstellung von stereoskopischen Fotoaufnahmen. Manche Informationsvisualisierungsalgorithmen benötigen viel Rechenzeit. Viele dieser Algorithmen lassen sich mit Hilfe von Parallellisierung interaktiv ausführen. Es wird ein System vorgestellt, welches die Parallelisierung sowohl auf Mehrkernprozessoren als auch verteilt über einem Netwerk unterstützt.

CHAPTER 1

INTRODUCTION

Visualization uses the visual channel to convey information to the user. This information can be for example a temperature on a surface. Data can also be described by a function. Functions may map a vector from one domain to a vector in another domain. Data can be characterized based on the dimensionality of the dependent and independent variables [WB97], see Table 1.1.

Table 1.1: This taxonomy uses the dimensionality of independent and dependent variables to organize data. Some fields contain diverse data types; typical examples are set in italic; e.g. the 2D/2D field not only contains 2D flow fields but also general 2-tuples like pressure and temperature given over a 2D domain.

dependent variables						
	mD	multivariate	MV	MV/*multi-field*	MV/*tensor*	MV
	3D	3-tuple(s)			*3D flow field*	
	2D	2-tuple(s)		*2D flow field*		
	1D	scalar value(s)	1D scalar field	2D scalar field	3D scalar field	
	0D		1D category	2D category	3D category	
		0D	1D	2D	3D	*n*D
		independent variables				

Each cell in the table can be associated with several visualization methods. After classification of the data visualization techniques falling in the corresponding category can be used to visualize the data. It should be mentioned that other taxonomies exist. The visualization taxonomy by Tory and Möller [TM04], consisting of a "high-level" and a "low-level" part, is claimed to be based on a design model and not on data. The taxonomy can be used to classify visualization techniques. However, their low-level taxonomy still is based on data much like the one presented above. Therefore, in this thesis the data-based model is used. Visualization methods can be classified in two main categories.

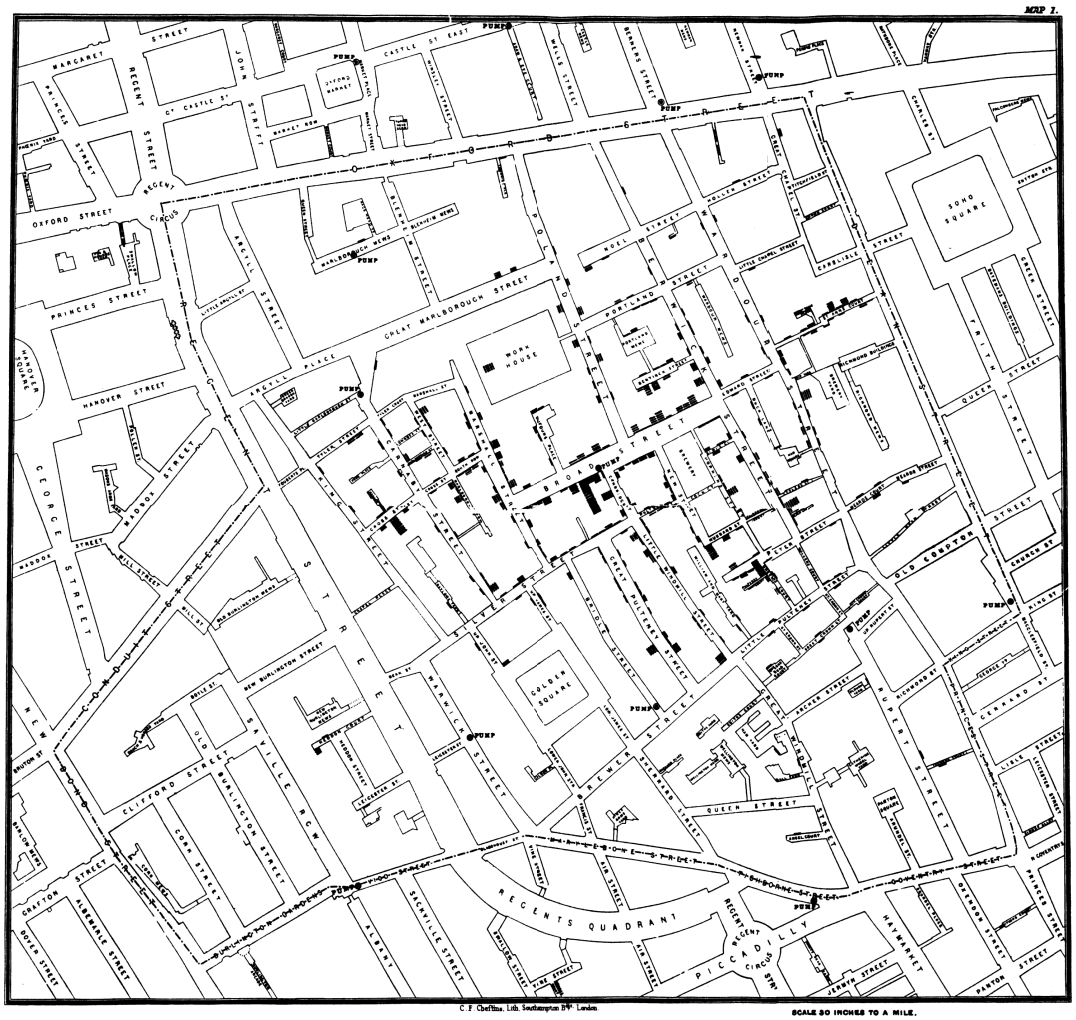


Figure 1.1: Broad Street cholera outbreak map, created by John Snow (1854).

According to Card et al. [CMS99] *scientific visualization* deals with scientific data, and *information visualization* deals with abstract data. Typically, visualization methods dealing with data having two- or three-dimensional independent spatial variables and at least one-dimensional dependent variables are *scientific visualization* methods; others are *information visualization*. Certain entries are also associated with other disciplines, like cartography. When considering land use, for example, we assign categories—e.g., forest, grass land, urban—to a 2D domain ("2D category"). For items having two attributes ("2-tuples"), scatter plots can be used, and for items having three ("3-tuples"), 3D scatter plots.

Sometimes it is hard to distinguish data having 0D dependent variables from data having 0D independent variables. Let us take as example the famous map created by

John Snow to analyze the Broad Street cholera outbreak (1854), see Figure 1.1; the individual cholera cases are drawn on a map like in the land use example but they are individual cases, each of them having several attributes, the dependent variables recorded are the location they died; these are positions on earth, which are 2-tuples. Things get changed if the data is recorded in a different way, like placing a 2D grid over the city and counting the cases, now for each position in the city we have a number of cases, which is a 2D scalar field. One could even ignore the number of cases and only record whether there was a case in the grid cell or not, this again results in categories over a 2D domain, like the land use example. Maybe the most accurate way to capture the data in Snow's map is the following: record the number of cases in each house (scalar values), together with the location of the houses and the entrance's orientation (2-tuple+scalar); now incorporating two data sources. However, this is still a simplification since it ignores the streets and house geometry.

Information is often given as a set of items having multiple attributes. Such data is called *multivariate data* (MV in the table). Various techniques have been developed to visualize multivariate data: scatter plots, parallel coordinates, dimensional stacking, and various glyph-based techniques; see the survey by Wong and Bergeron [WB97]. Visualization of multivariate data has made the way to the average Internet user. Gapminder Trendanalyzer [Tre], a web-based application, brought multivariate data visualization with animated bubble charts to a broad audience. Google Inc. acquired the software [May] and is pushing forward with Google public data explorer, which is in a beta stadium today. It provides support for basic diagrams like bar and line charts; but also for multivariate data visualization with bubble charts, which are 2D scatter plots with the ability to map an additional data dimension to color and another one to point size; in addition, they support time-based animation (0D domain (+1D animation), 2D values (+2D color and size)). Animation is often preferred by users [HR07, RFF+08]; however, Robertson et al. [RFF+08] found traces and small multiples superior for the analysis of time varying data with bubble charts. Multivariate data refers to the dimensionality of the dependent variables and multidimensional data refers to the dimensionality of the domain. In this thesis, the focus is multivariate data with 0D independent variables, and primarily 3D scatter plots are used for visualization.

Such a 0D independent variables visualization method can also be applied to multidimensional data (nD independent variables). This can be done by forming $(n+k)$-tuples out of nD sample position and the corresponding function value of the continuous function $f : \mathbb{R}^n \to \mathbb{R}^k$. Multidimensional visualization methods try to visualize a value given over the entire domain. In the simplest case the value is a scalar, and a corresponding visualization method for data on a 3D domain is volume rendering. Multivariate visualization methods however, mostly rely on the presence and absence of certain value combinations. Such data transformations therefore, are often not applied to visualize the same phenomena, but to visualize different aspects. For example the multivariate

histogram, used to design a transfer function for volume rendering (see Section 3.7), does not show the scalar function value over the spatial domain, but the occurrence of certain value-derivative combinations. The backward transformation can be achieved by defining a function (f') through interpolating between the samples. Another way to transform a multivariate data set into a multidimensional one is to use a kind of density estimate. Such a density estimate is the mechanism which usually is associated with multivariate representations. John Snow's Broad Street cholera outbreak map relies on a visual density estimate to be performed by the viewer's perception. To automatize this process, a kernel density estimate could be used. The reverse transform would be a density dependent sampling. Data transformations make the presented techniques applicable to a multitude of data.

The thesis presents how to visualize large data amounts, consisting of millions of individual multivariate data items through the use of graphics hardware. It is also shows how to employ 3D scatter plots for the analysis of multivariate data by navigation between 3D scatter plots.

1.1 Outline

When dealing with 3D scatter plots, depth perception plays a central role, this role and the role of 3D visualizations in the information visualization community are discussed in Chapter 2.

Chapter 3 describes two novel techniques which improve shape perception of 3D scatter plots. One is an illumination technique which can be used with dense scatter plots and the other is adaptive halo rendering at depth discontinuities which also works best with dense scatter plots but provides advantages even for very sparse scatter plots. The illumination technique was also published in [SW09]; the halo rendering technique was briefly presented in [SW12].

Multivariate data has often more than thee relevant attributes and cannot be displayed by a single 3D scatter plot. In Chapter 4 a technique is presented which can be used to navigate between different 3D scatter plots by using rigid body rotations and this way preserves context. The corresponding paper was published in [SW12]. Techniques presented in Chapters 3–6 are integrated in an application called Multidimensional Analyzer, which is also introduced in Chapter 4.

Data mining is used to analyze multivariate data automatically by applying machine learning (ML) algorithms. Decision trees can be learned by a supervised learning algorithm [Qui92]. Chapter 5 shows how visualization can be used to help in feature engineering and how decision trees can be used to gain insight into multivariate data by support of an automatic decision tree learning algorithm. A visual analytics technique using 3D scatter plot visualization has been developed to support Natural Language Processing [MS99] (NLP) experts in the development of new features used for classification

with decision trees. The NLP experts André Blessing and Hinrich Schütze evaluated the system, and the results were presented together with them in [SBSW09].

Chapter 6 again has the focus on depth perception but using different depth cues, namely stereopsis and convergence. A novel technique for anaglyph stereo is presented with the goal to create anaglyph stereo images without ghosting artifacts. Although stereoscopic rendering of 3D scatter plots makes sense [Yan99] and is included in the Multidimensional Analyzer application, the presented technique is not limited to 3D scatter plot renderings and can even be used to remove ghosting from stereoscopic or anaglyph photos. The technique was published in [SW11].

Thanks to the improvements in graphics processing unit (GPU) technology, many visualization algorithms can be realized nowadays on single PCs. However there still are use cases where a single machine is not capable of providing adequate visualizations in real-time. In Chapter 7 a visualization framework is presented which is capable of parallelizing algorithms on shared and distributed memory systems. Such parallelization is not necessary for most techniques introduced above, however the illuminated scatter plots technique presented in Chapter 3 uses multiple cores to reduce computation time and the continuous scatter plots technique [BW08a]—also using scatter plots—could benefit from such parallelization since it has high computational demands as well. The Java-based stream processing framework for Nexus, NexusDS [CEB+09], developed primarily by Nazario Cipriani, has been extended to support distributed real-time visualization; the results were published together with him in [SCW11].

The content of the publications mentioned above [SW09, SBSW09, SW11, SCW11, SW12] has been partly reused with permission in this thesis.

H. Sanftmann, N. Cipriani, and D. Weiskopf. Distributed context-aware visualization. In *Proceedings of the IEEE Pervasive Computing and Communications Workshops*, pages 251–256, ©2011 IEEE.

H. Sanftmann and D. Weiskopf. 3D Scatterplot Navigation. *IEEE Transactions on Visualization and Computer Graphics*, pages 1969–1978, ©2012 IEEE.

CHAPTER 2 THE NOTION OF 3D IN INFORMATION VISUALIZATION

This chapter introduces the cue theory of perception which provides a model for human depth perception. Then different view points on the use of 3D visualizations are examined. Subsequently, existing statistical graphics techniques are presented. The chapter concludes with a comparison of 2D and 3D scatter plots and implications on the presentation of 3D scatter plots.

2.1 Perceptual Psychology

Depth perception is the process which allows humans to perceive the 3D shapes of objects and the entire scene through the visual channel. This ability is based on depth cues. Here the classification of depth cues given by Ware [War04] is adapted, by separating monocular static cues in pictorial and non-pictorial ones as well as grouping cues caused by perspective projection, see Figure 2.1.

In the following depth cues are explained citing various literature. However, when describing depth cues, authors often caracterize them as "strong" or "important" in two senses. Here these are differentiated: the more "dominant" feature overrides the other in case of contradiction; humans gain more information through their visual system out of "informative" features. Occlusion is, according to Ware [War04], probably the most dominant depth cue. Cast shadows are according to studies of Wanger et al. [WFG92] more dominant than texture, perspective cues or the kinetic depth effect. Linear perspective refers to the fact that parallel lines converge. Relative size means that identical objects appear smaller at larger distance. Depth of focus terms the fact that out of focus objects are blurred. Proximity-luminance covariance (PLC) is according to Dosher et al. [DSW86]: "intensification of edges in proportion to their proximity to the observer". This cue is responsible for aerial perspective [Cut97]: distant objects have lower luminance contrast and color saturation. Monocular dynamic cues are known as structure from motion (SFM), maybe the most informative of the monocular cues [KSJ00]; according to Ware's

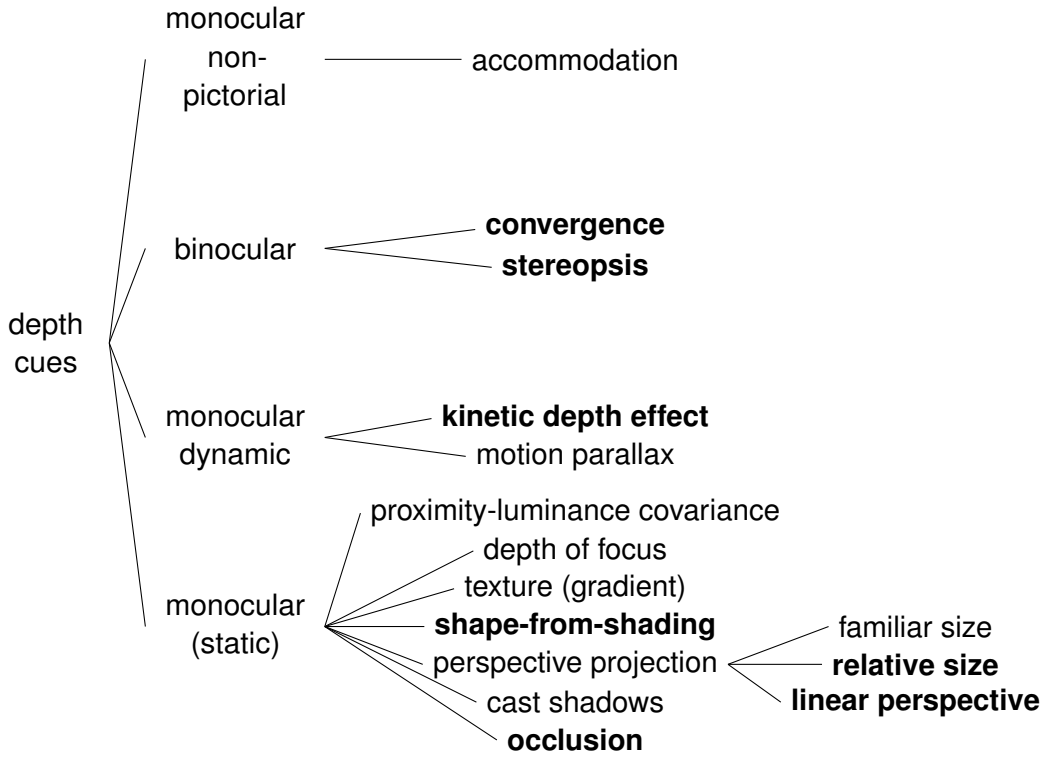


Figure 2.1: Depth cues are categorized. Depth cues which are treated with special attention in this thesis are set in bold.

argumentation [War08] even more informative than stereopsis. However, stereopsis aka. (retinal) disparity, is regarded as the most informative depth cue by others [GPK+09]. Motion parallax refers to the information gained through motion in an environment: e.g., driving car and looking sideward; or moving forward. Shadows cast by rotating 3D objects reveal their 3D shape; this is termed the kinetic depth effect. Few depth cues deliver absolute distance. One of them is familiar size: e.g., by knowing the size of a car we are able to estimate how far it is. Convergence, the sensation from the extraocular muscles, is absolute but only relevant at close distances [TMWK99]. Accommodation, sensed by the ciliary muscles which change focal length, is absolute but is also only relevant at close distances [WF71].

The classification of depth cues is a good starting point but perception is still not fully understood nowadays. As an example, Ware et al. [WGP98] show that dynamic changes in virtual eye separation (distance between cameras when generating stereoscopic images) are not noticed if they occur over a period of a few seconds, which contradicts the observation that convergence (at least for small distances) is absolute.

Perceptual scientists try to find out how effectively humans can perceive 3D structure by using these cues. Some studies show that humans are able to perceive metric structure

others discovered systematic distortions, as reported by Todd and Norman [TN03]. They found strong evidence that human observers do not have accurate perception of 3D metric structure from texture, motion, or binocular disparity. However, they admit that it is hard to identify and present all relevant information sources in artificial renderings which would be present in natural environments. In a follow up publication, Todd [Tod04] reported results on the 3D metric structure acquired by shape from shading to have large differences among measurements. However, the differences can be largely reduced by applying an affine shearing transformation which makes the resulting structures nearly identical. This indicates that human perception of 3D shape can significantly differ when considering 3D metric space but at the same time be quite accurate in other spaces. Which means, humans may not use the metric space to judge the shape of 3D objects, and these other spaces may be more relevant for human shape perception. Porrill et al. [PDT+10] analyzed the importance of depth cues in 3D metric task performance and found out that binocular cues contribute to performance in a full-cue condition and are the only effective cues in a reduced-cue condition where the head of the study participants was fixed and their field-of-view was restricted by a viewing tunnel.

2.2 Physiology of 3D Perception

The last section approached perception from the point of view of perceptual psychology. This section approaches the topic from a physiological point of view. A good introduction to neural science is given by Kandel et al. [KSJ00]. Clinical studies on humans sustaining an injury in a certain part of the brain provide a way to gain insight in the function of different brain areas. Other possibilities are experimental studies. Many of these experiments are destructive and are therefore carried out on monkeys. In recent years, technological development has led to new possibilities: positron emission tomography (PET) and functional magnetic resonance imaging (fMRI).

Early stages of visual processing are now understood quite well whereas processing at later stages is mostly unknown [KSJ00]. Light is captured by the photo receptors, rods and cones in the eyes whose signals are bundled by the retinal ganglion cells; depending on their receptive fields most of them fall into two categories: P (for *parvi* or small) and M (for *magini* or large). The signals are processed separately in the lateral geniculate nucleus and kept separate until they reach the primary visual cortex (V1). In an experimental study on monkeys, Schiller et al. [SLC90] found out that the M pathway does not transmit color information and has higher (luminance) contrast sensitivity to higher temporal frequencies; whereas the P pathway transmits color information and has higher contrast sensitivity to spatial frequency and lower temporal frequencies. Further processing is mainly separated in the dorsal (parietal) pathway, dominated by M input and the ventral (inferior temporal) pathway requiring both inputs. For further details see Kandel et al. [KSJ00].

Although monkeys are used for experimental studies to gain insight in human perception, the processing of visual information in monkeys is not identical to humans. Such a difference is observable in the brain region V4 (part of the ventral pathway). Humans with lesions of V4 cannot discriminate colors but maintain shape perception, which is exactly contrary to monkeys [KSJ00].

Various recent studies show that 3D shape is perceived differently by humans and monkeys. Vanduffel et al. [VFP+02] analyzed processing in brains of humans and monkeys by using the fMRI response to several 2D and (structure from motion) 3D stimuli. They found no difference in low- and mid-level processing; but found out that the brain area called lateral occipital complex (LOC) is activated in humans but not in monkeys when exposed to 3D SFM. They argue that this suggests that these processing areas are not present in monkeys. According to Murray et al. [MKO+02], 3D shape perception increases activity in the LOC while reducing the activity in the primary visual cortex, which is an earlier stage in perception. This implies that 3D rigid body motion can be processed more easily than non-coherent 2D motions. Murray et al. compared rotating dots point clouds sampled on 3D object surfaces with dots moving with scrambled velocities.

Our user study presented in Section 4.3 comparing interpolation of 2D positions to two consecutive rotations of scatter plots obtained similar results. The two scenarios started with the same 2D scatter plot and ended with scatter plots which just slightly differed from each other. However, significant better task performance was obtained for the rigid body rotation (SFM) scenario.

Finally, using fMRI Kourtzi et al. [KEGB03] found out that parts of the human LOC represents 3D shape and not 2D object contour. All these findings show that humans have some unique and novel "hardware" built in for 3D object representation and using this for information visualization may be rewarding.

2.3 Comparing 2D and 3D Visualizations

In this section, several view points analyzing 2D and 3D visualizations are presented with the application to 2D and 3D scatter plots in mind.

Maybe 3D metaphors are not familiar to the users and 2D metaphors should be used on a 2D screen? Ware [War04] points out that the desktop metaphor used in current user interfaces is not a 2D metaphor, it is 3D. Occlusion—a 3-dimensional space (3-space) feature—is used for the overlapping windows. The situation with scatter plots is the same; according to my taxonomy of scatter plot visualization techniques (see Table 4.1), 2D scatter plots do not have a depth axis and hence have no occlusion or any kind of sorting order. With an additional depth axis we are dealing with 3D scatter plots; and such a 3D desktop metaphor is working for everybody who is using a modern computer.

With the emerging 3D rendering technology various 3D information visualization

methods have been developed. Robertson et al. [RMC91] introduced Cone Trees to visualize hierarchies in 3D. The children of a node are laid out as cones in 3D, which could be rotated to select a child. The authors found out that the 3D representation is useful to maximize the effective use of the limited available screen space and interactive animation is useful to shift the user's cognitive load to the human perceptual system. The "animation" they used was a special one, it was a rigid body rotation in 3D, which is especially well suited to support human perception by the kinetic depth effect, see Chapter 4 for the benefit of using the kinetic depth effect over other animations. Tree-Maps [JS91] were introduced in the same year as Cone Trees. In contrast to them, Tree-Maps use a 2D representation and require no interaction for navigation, however zooming can be useful to display details. Cushion treemaps [vWvdW99] are based on applying illumination to Tree-Maps, in order to present the tree structure as shape from shading depth cue. Now, more than 20 years after the initial techniques were introduced, cushion treemaps are a popular method to visualize large hierarchies, whereas it has been shown by a formal user study that Cone Trees are outperformed by tree browsers [CM00] on search tasks.

Sebrechts et al. [SCL+99] evaluated task performance with a textual, a 2D, and a 3D interface of a tool visualizing search results. They found out that computer experience has the highest impact on the 3D interface performance. Additionally, the strongest learning effect was obtained with the 3D interface. They concluded that 3D visualizations cannot be evaluated in short term studies of novice users. However, these results might be findings valid for the specific visualizations tested and not generalize to other 2D or 3D visualizations. Bemis et al. [BLW88] tested whether 2D or 3D perspective is better to display air-traffic information; 3D perspective was significantly superior in terms of error and response time. Smallman et al. [SSJOC01] found out that this benefit is resulting from properties of the specific visualization, which not only differ in 2D vs. 3D but also in the availability and representation of information.

It is not obvious how to design a novel visualization technique which uses a 3D representation instead of a well-established 2D visualization without providing additional information. Hence in general other properties will change, too. Since comparisons are performed between two specific visualization techniques the results do not answer the question whether 2D or 3D representations are better suited to display certain data but answers the question for the specific visualizations.

However, for certain visualizations the question whether 2D or 3D is better suited can be answered. Siegerst [Sie96] analyzed whether 2D or 3D pie and bar charts are better to display the same lower-dimensional data. He reported that the relative magnitude can be estimated better from 2D pie charts and 2D bar charts can be read faster than their 3D counterparts. To display higher-dimensional data the advantages of 3D often outweigh its disadvantages. One recent example is given by Dang et al. [DWA10], who show how stacking graphical elements in the third dimension overcomes serious problems of color

coding in 2D plots. Their stacking approach is inspired by dot plots [Wil99]—used in statistics to represent one-dimensional sampling distributions—which are extended to show two-dimensional sampling distributions. The resulting visualization has similarity to John Snow's Broad Street cholera outbreak plot, where the latter aggregated the cases over houses and not a certain Euclidean distance and used the 2D plane for stacking instead of a third dimension.

This thesis focuses on 3D scatter plots. For them the question cannot be answered this easily since 3D scatter plots can display one data dimension more than their 2D counterparts. This fact was also noted by Artero and de Oliveira [AdO04], who observe that a 3D projection of multivariate data conveys more information than a 2D projection and introduce Viz3D by extending the RadViz [HGM+97] technique to three dimensions. Even when comparing 3D scatter plots to 2D scatter plot matrices, 3D scatter plots still contain more information than the three projections along the axes shown by a scatter plot matrix. However, the information contained in a 3D scatter plot cannot be provided to a human observer by a static image. The user needs to rotate the 3D scatter plot in 3D space to view it from different directions. The necessity of this interaction and how 3D objects are observed in our 3D world is illustrated by Abbott [Abb84] by a 2D analog, the Flatland. This ability to examine more projections and possibly gain more insights comes at a cost: there is more user interaction needed and the time to examine a 3D scatter plot is higher than for 2D representations.

The *topological landscapes* [WBP07] method uses terrain rendering to represent a scalar function over an *n*D domain. The method can be categorized as a scientific visualization method. To be more precise, topological landscapes reproduce the contour tree defined by the topological structure of the isosurface for different iso values. The contour tree itself is a tree with weighted nodes. Visualization of trees is a core topic of information visualization, making the method be part of both domains. Recently topological landscapes have been applied to visualize multivariate data [OHJS10] by defining a density function.

Minard's famous flow map [Min69] visualizes the march of Napoleon's army to Moscow and back, see Figure 2.2. The main forces were split at two locations and rejoined on the march back. The plot depicts the strength of the force on a map; additionally it shows the temperature as line graph annotated with the time for the way back in a separate connected plot. As data representation one could record the strength of the force, the location, and the temperature over the time; resulting in 4D values over a 1D domain. Basically, a trajectory on a map is used as graphical representation, which does not represent the time value. Hägerstrand [Häg70] introduced the space-time cube, where a third axis is used to show time in addition to space. However, user interaction is necessary to explore trajectories depicted in such a 3D coordinate system. One recent application of the space-time cube is in the domain of video visualization [BBS+08]. In a user study, Kristensson et al. [KDA+09] reported that the space-time cube method is

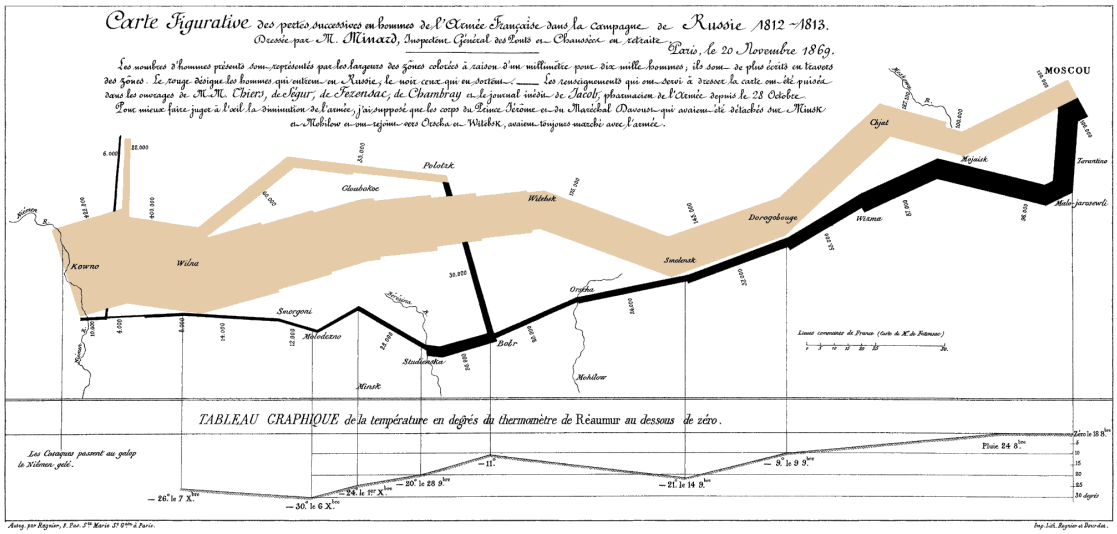


Figure 2.2: Visualization of Napoleon's march to Moscow, created by Charles Minard (1869).

twice as fast as annotated trajectories for duration related tasks.

Others try to answer whether 3D perspective or orthographic projection is better for certain tasks. St. John et al. [SJCSO01] state that 3D perspective display (on a 2D screen) is good for shape understanding and 2D for precise judgments of relative position. They performed several user studies which support this observation and analyzed several studies for the air-traffic scenario. Tory et al. [TKAM06] also compare 2D, 3D, and combined 2D/3D displays like orientation icon, ExoVis [Tor03], and clip planes. They report that 3D displays can be effective for approximate navigation and relative positioning when appropriate cues like shadows are present. However they find out that orientation icon and ExoVis displays are better for precise navigation. Tory et al. define the terms view and display to distinguish how the views are taken—orthogonal in main axis direction vs. perspective projection—from how they are displayed. And even the simplest 2D views were presented by arranging them on the faces of an isometric projected cube, which makes the interpretation of their results hard. Especially no conclusion about simple 2D views can be drawn, which might perform even better than ExoVis displays.

2.4 A Statistical Graphics Point of View

Scatter plots are just one kind of statistical graphics. This section presents other statistical graphics representations, discusses their information content, and illustrates when it is beneficial to use interactive 3D scatter plots and not their traditional 2D counterparts.

3D scatter plots can display one additional dimension compared to their 2D counter-

parts. What does one additional dimension mean? This additional dimension is more than one would expect in the first place.

We can analyze data observed from a random variable, and describe it by some statistical measures. Most common are the mean, the standard deviation, and the skewness.

Besides statistical values, describing data, several graphical representations of sample distribution exist. A rug plot simply shows the data samples with respect to the data axis. Histograms provide another visualization method for sampled data which can be compared to a continuous probability density function. A box plot is a graphical representation which can be used to visualize the sampling results of a random variable by showing the median, with the lower and upper quartile, the minimum and maximum value, and possible outliers.

Figure 2.3 shows the three graphical representations for the air quality data set [CCTK83] which consists of measurements on ozone, solar radiation, wind speed, and temperature measured in New York from May to September 1973 on a daily basis.

Box plots, histograms, and rug plots are useful to look at individual random variables. However, many data sets are—like the air quality data set—multivariate data sets; this means we have not just one random variable given for each sample but several variables. To understand multivariate data sets, we need to look at the interaction between variables; therefore, we need a different representation which displays the interaction between dimensions. Scatter plots can display the interaction between variables, see Figure 2.4. E.g. it can be observed that Wind and Ozone are negatively correlated, which can also be captured by their correlation coefficient (r=-0.612). However, the scatter plot provides more information than a single number, see the paper of Anscombe [Ans73] who promoted the use of scatter plots in statistical analysis in addition to calculations.

While 2D scatter plots display the interaction between two variables, 3D scatter plots show the interaction between three variables. Often people claim that 3D scatter plots show "only" one additional dimension compared to 2D scatter plots and this one additional dimension does not matter if the whole data set has hundreds of dimensions. This is simply a wrong argumentation. No one would argue that 2D scatter plots show "only" one additional dimension compared to rug plots. The same way 2D scatter plots are useful for displaying the interaction between two variables, 3D scatter plots are useful for displaying the interaction between three variables.

2.4.1 Limitation of this Perspective

If all the numerical values with respect to each dimension are distinct to an extent that they can be matched between the 2D scatter plots, the nD structure can be reconstructed from 2D scatter plots. This is not possible with rug plots. This observation has more a theoretical value since the exact numerical values are hard to distinguish for a human observer and even harder to relate the points from several scatter plots to an nD structure. However, this observation reflects the fact that there is not a new relation which can be

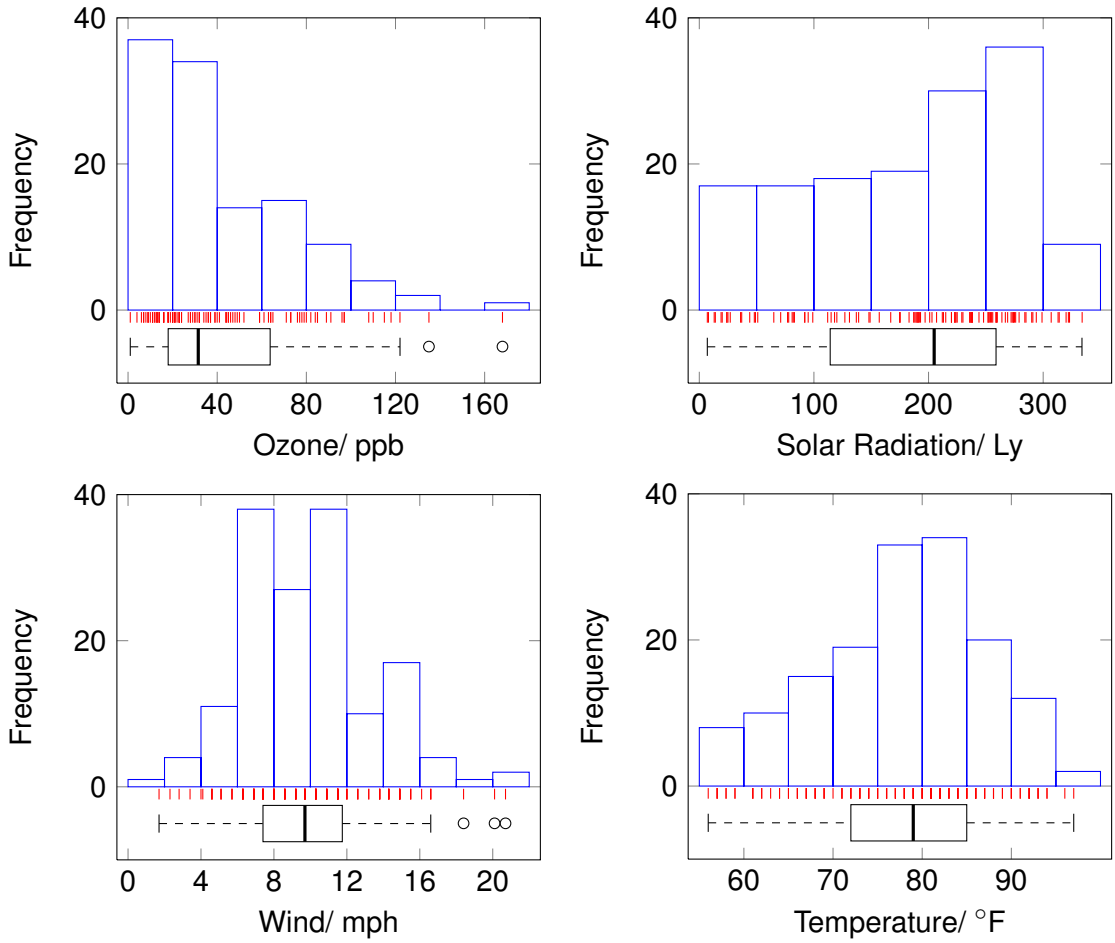


Figure 2.3: Statistical graphics show one individual variable at a time for ozone, solar radiation, wind speed, and temperature. Box plots are shown in black, histograms in blue, and rug plots in red color.

obtained by using nD scatter plots instead of 2D scatter plots; both reflect the correlation of points, which is not present in rug plots.

In some cases the interaction between three variables can be observed from two 2D scatter plots. When wind and ozone are negatively correlated and temperature and ozone are positively correlated, then wind and temperature need to be negatively correlated, too. However, even if there is no visible correlation in the 2D scatter plots there still can be a strong correlation (structure) in 3D. Which means that in simple situations 2D scatter plots are just sufficient to derive all significant results, but in more complicated cases the data can only be analyzed by looking at more than two variables at once.

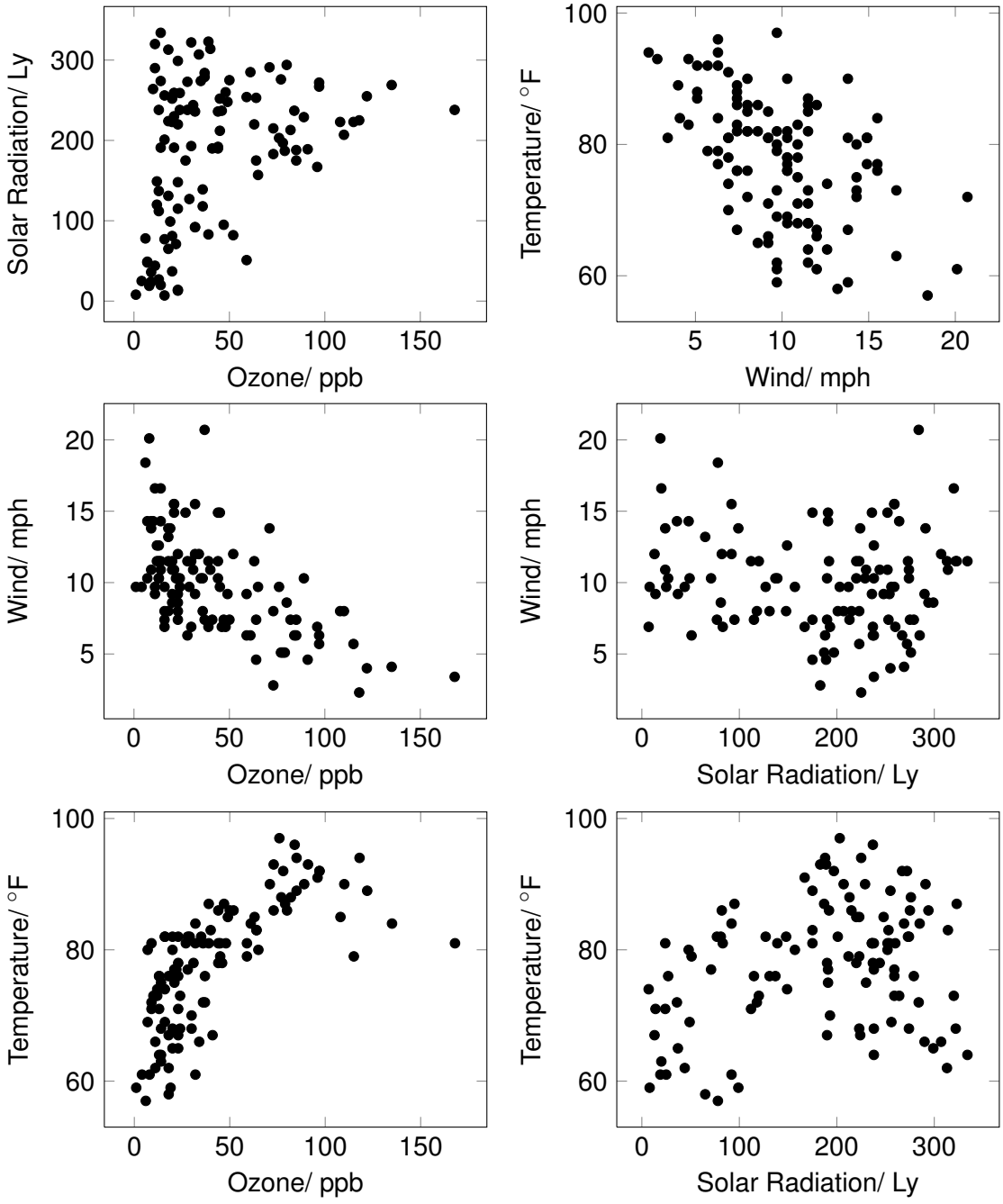


Figure 2.4: Scatter plots show the interaction between variables.

2.5 How to Show 3D Scatter Plots to Users

The previous section pointed out that 3D scatter plots contain more information than 2D scatter plots and are useful to capture the interaction between three variables. This is the reason for their application in visualization methods.

Early work on 3D scatter plots was concerned with rendering performance. Donoho et al. [DDG88] present a system that uses 3D scatter plots for multivariate data exploration. Point clouds up to 5000 points could be rotated interactively. Becker [Bec97] applies volume rendering to display dense 3D scatter plots. Points are packed into corresponding voxels that are rendered with splatting. The number of samples in a voxel is mapped to opacity; the average value of one data variable in each bin is mapped to color. Their system was capable of displaying 2500 voxels (1/3 of them holding data) at interactive rates. Reina and Ertl [RE04] employ texture-based volume rendering for getting an overview of large data sets and rendering of individual values to interact with sub-regions of the data set.

3D scatter plots have been and still are [TFO09] employed at various places to gain insights mainly because they provide an additional data dimension and reduce information loss [PEP+11]. However, the 3D nature of the internal representation of the data is often provided to the user only by the ability to rotate the viewport to generate new projections on a 2D screen [CRS+96]. This is the simplest way to present 3D scatter plots; it only gives the user the ability to interactively rotate the view in 3D space and by structure from motion he can perceive the 3D structure. This may be one of the main reasons why 3D scatter plots have received many critics for requiring too much user interaction, which may be valid in some cases.

Van Wijk [vW05] introduced a measure to judge the value of a visualization based on an economic model, which can be applied to evaluate the use of 3D scatter plots. In the following passage, the statements written in parentheses refer to terms introduced by the model. Considering the additional time needed for the 3D interaction, the exploration costs could be higher for 3D scatter plots (captured by the term kC_e in the model, where C_e is the *perception and exploration cost* and k the number of explorative steps). However, the acquired knowledge (ΔK) should be higher for 3D scatter plots since all the 2D projections which can be obtained by 2D scatter plots, can also be generated using 3D scatter plots. The profit difference between 2D and 3D scatter plots depends on the value of the acquired knowledge ($W(\Delta K)$) and the changing exploration cost described above ($-kC_e$). This mainly states that if you assign a high enough value to the acquired knowledge (defined by W), even the most basic 3D scatter plots outperform 2D scatter plots, assuming the data set is not too simple.

To maximize the utility of 3D scatter plots, the acquired knowledge should be maximized and/or the exploration cost should be minimized. First improvements to the spatial perception of 3D scatter plots were proposed by Donoho et al. [DDG88], like adjusting point size based on depth and stereopsis. The use of stereographic depth

in a CAVE environment was presented by Yang [Yan99]. Other improvements are the use of brushing and linking—introduced for 2D scatter plot matrices by Becker and Cleveland [BC87]—by Kosara et al. [KSH04]. They simultaneously show several 3D scatter plots with different axis mappings, which are linked through color mapping. The depth perception of 3D scatter plots is further improved by Piringer et al. [PKH04] by mapping distance to color and to point size. They additionally employ halos to support the identification of individual sample points. A hierarchical cluster visualization was presented by Linsen et al. [LVLRR08]. The clusters are detected in high dimensional feature space and projected to 3D space using principal component analysis (PCA) on cluster centroids. In a related work, the Least Square Projection method, introduced by Palovich et al. [PNML08], was extended to 3D [PEP+11] and replaced the PCA. In a quantitative study, they showed that 3D scatter plots outperform 2D scatter plots in terms of finding neighboring clusters to points but they also found out that 3D interaction is more difficult and more time-consuming.

3D scatter plots can also be enhanced by displaying lines which connect the 3D points to a ground plane. A related visualization is to display 3D bar charts over a 2D plane. However, this introduces clutter to the plot. Healey et al. [HAC01] mapped additional data attributes to the bars. For scatter plots it is also common to map additional attributes to point size, glyph type, or color. This technique is also employed at various places in this thesis. 3D scatter plots are just one of the possibilities to show multivariate data, however a quite appealing one. Before other attributes, they use one additional position attribute. Position can be perceived more accurately than all other attributes, like length, angle, slope, area, volume, color density (luminance), saturation, or hue [Mac86]. However, only two position attributes can be presented simultaneously, the third is mapped to depth.

The primary goal of the next chapter is to provide additional depth cues for 3D scatter plot visualization to maximize the acquired knowledge from a given view minimizing the number of viewpoints the user needs to explore and thereby minimizing the exploration cost.

CHAPTER 3 IMPROVING DEPTH PERCEPTION OF 3D SCATTER PLOTS

In contrast to 2D scatter plots, the existing 3D variants have the advantage of showing one additional data dimension, but suffer from inadequate spatial and shape perception and therefore are not well suited to display structures of the underlying data. In this chapter, techniques are presented which aim at improving perception of 3D scatter plots.

A method introduced in this chapter is an illumination technique for the point cloud representation of 3D scatter plots. The results were published in [SW09]. The technique lends itself to efficient GPU point rendering and can be combined with existing methods such as semi-transparent rendering, halos, and depth- or attribute-based color coding. It significantly improves shape perception especially in high density areas.

Halo rendering improves depth perception in lower density areas but is distracting when applied to high density scatter plots. This chapter also presents a novel, improved technique which renders halos only at significant depth discontinuities—the technique is briefly presented in [SW12]—which makes halo rendering applicable to high density scatter plots. The two techniques can be easily combined with each other.

3.1 Illuminated 3D Scatter Plots

As pointed out in Chapter 2, traditional 3D scatter plots require depth cues to effectively convey the 3D structure to the user. If some structures are present in the data, the introduced technique aims at supporting the user in identifying those structures and recognizing the relation of points in 3-space. In 3-space, there are 1-manifolds, 2-manifolds, and 3-manifolds, which can be automatically recognized based on a local principal component analysis. In contrast to traditional point-based rendering techniques, which focus only on the extraction of surfaces, the approach introduced classifies each data point according to linearity, planarity, and sphericity by performing an eigenvalue decomposition of the covariance matrix in the local neighborhood of each sample. Different lighting models are applied based on the classification of each point. Since

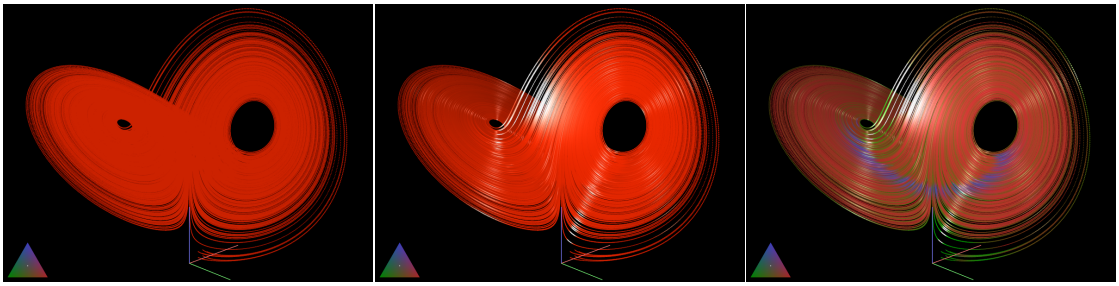


Figure 3.1: A Lorenz attractor is rendered differently. Left: traditional 3D scatter plot; middle: illuminated scatter plot; right: linear, planar, and spherical structures highlighted through mapping to green, red, and blue colors respectively. The base colors are chosen to have equal luminance. For the calculation luma is used which is a reasonable approximation in nonlinear RGB space.

the classification and the eigenvectors at close samples are correlated, the resulting illumination is smooth and gives the user the ability to recognize the shape by examining a shaded image. The novelty of this approach is the flexible dimension/codimension rendering of point clouds. The improvement of the shape perception is based on the shape-from-shading depth cue (see Section 2.1) and is complementary to previous techniques using other cues for depth perception, such as structure from motion, halo rendering (supported by occlusion), depth-based color mapping (related to aerial perspective), or relative object size [KSH04] [PKH04].

The technique does not target the representation of individual points; the goal is to present structures in dense scatter plots. Without lighting the 3D shape of the structures is hard to recognize (Figure 3.1, left), whereas the 3D structure becomes visible by illuminating the points (Figure 3.1, middle and right). Figure 3.1 shows a data set obtained by integrating over the Lorenz attractor. This data set is used throughout this chapter to illustrate several effects of illuminated scatter plots. Illumination also guides exploratory analysis since the 3D shape can be tracked by the eyes much better than uniformly colored samples. In Figure 3.1, specular highlights resulting from the illumination of planar structures are visible in the center of the image. There are also specular highlights that result from linear structures (e.g., the one from the right “hole” to the origin of the coordinate system), which would not be present without considering linear structures. In this way, the technique enhances 3D shape perception.

3.2 Related Work

This section covers previous work related to point-based rendering, illumination, and blending.

Typically, point-based rendering focuses on representing 2D surfaces in 3-space. A survey of point-based rendering techniques can be found in [KB04]. As one example, Schall et al. [SBS05] perform a principal component analysis of scattered points representing a smooth surface to filter outliers; they use point-based rendering to render the illuminated surface. In the technique presented in this chapter, the usage of the principal component analysis is extended to incorporate linear and spherical structures as well in the rendering. The point-based rendering technique by Hopf et al. [HLE04] is suitable for large 3D scattered point data sets, but is restricted to showing spherical structures.

Banks [Ban94] studies illumination in different codimensions. The special case of codimension-2 in 3-space is applied to streamlines by Zöckler et al. [ZSH96]. Here this illumination model is employed for linear structures in point clouds.

The covariance matrix is similar to a diffusion tensor: it is symmetric and positive semi-definite. Based on the eigenvalues a diffusion tensor can be classified in a space where linear, planar, and spherical structures act as spanning vectors of that space [WPG+97]. This classification scheme is adopted in the technique presented here. Kindlmann and Weinstein [KW99] also use this classification to illuminate diffusion tensors, differentiating between linear and planar structures. In contrast to this chapter, they do not blend between three separate illumination methods for the three basis structures and they do not support the assignment of different material properties to them.

3.3 Flexible Codimension Rendering

The goal is to illuminate 3D scatter plots. Differently classified structures should be assigned different material properties and different illumination models need to be used for them. This section will explain how scattered points can be classified, how the tangent and normal vectors can be calculated for them (which are needed for lighting), and how the lighting calculations are performed.

3.3.1 Eigenvector and Eigenvalue Calculation

A 3D scatter plot is given as a set of points $P = \left\{\boldsymbol{p}_i \in \mathbb{R}^3\right\}$ in 3-space. Illumination requires additional information on top of point positions. We need normal vectors to illuminate planar surfaces and tangent vectors to illuminate lines. Principal component analysis allows us to estimate normal and tangent directions of scattered data points. The principal components are derived by calculating the eigenvalues and eigenvectors of the covariance matrix.

The weighted covariance matrix according to [SBS05] is used:

$$C_i = \sum_{j=1}^{N} (\boldsymbol{p}_j - \boldsymbol{c}_i)(\boldsymbol{p}_j - \boldsymbol{c}_i)^T \chi\left(\frac{\|\boldsymbol{p}_j - \boldsymbol{p}_i\|}{h}\right)$$

where h is the kernel size, χ is a monotonically decreasing weight function, and $\boldsymbol{c}_i$ is the weighted average of all samples inside the kernel.

To account for varying point density, the kernel size is adjusted to cover a certain number of points instead of covering a constant volume. For each point a local neighborhood N_i^n of $\boldsymbol{p}_i$ is defined as the n closest points to $\boldsymbol{p}_i$. The distance to the farthest point defines the kernel size h_i^n for $\boldsymbol{p}_i$:

$$N_i(d) = \left\{\boldsymbol{p}_j \,|\, \|\boldsymbol{p}_j - \boldsymbol{p}_i\| < d\right\} \tag{3.1}$$

$$h_i^n = \max\left\{d \,|\, |N_i(d)| \leq n\right\} \tag{3.2}$$

$$N_i^n = N_i(h_i^n) \tag{3.3}$$

The following weight function χ is used:

$$\chi(x) = \begin{cases} 1-x^2 & \text{if } |x| < 1 \\ 0 & \text{else} \end{cases}$$

This is the triangle function applied to the squared normalized distance.

3.3.2 Classification

The final color $\boldsymbol{C}$ of a sample is calculated by applying different lighting models for 1-manifolds (l), 2-manifolds (p), and 3-manifolds (s) and weighting the results based on the classification of the sample:

$$\boldsymbol{C} = d_\mathrm{l}\boldsymbol{C}_\mathrm{l} + d_\mathrm{p}\boldsymbol{C}_\mathrm{p} + d_\mathrm{s}\boldsymbol{C}_\mathrm{s}$$

where $\boldsymbol{C}_X$ ($X \in \{\mathrm{l},\mathrm{p},\mathrm{s}\}$) is the color resulting from the illumination model and d_X the classification weight for the respective manifold type. Now the classification of a sample will be derived.

Similar to diffusion tensor imaging [WPG+97], classification can be performed based on the properties of the covariance matrix. Let $\lambda_0 \leq \lambda_1 \leq \lambda_2$ be the eigenvalues and $\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2$ the corresponding normalized eigenvectors of the covariance matrix. The point $\boldsymbol{p}_i$ can be classified based on the eigenvalues in the linear, planar, and spherical cases [WPG+97]:

$$c_\mathrm{l} = \frac{\lambda_2 - \lambda_1}{\lambda_0 + \lambda_1 + \lambda_2}$$

$$c_\mathrm{p} = \frac{2(\lambda_1 - \lambda_0)}{\lambda_0 + \lambda_1 + \lambda_2}$$

$$c_\mathrm{s} = \frac{3\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}$$

The values c_l, c_p, and c_s sum up to one, i.e., they can be interpreted as barycentric coordinates.

The user can adjust the classification through a transfer function by manipulating the weight coefficients w_X. The values c_X are multiplied with w_X and re-normalized (summing up to 1), resulting in the values d_X, which are taken for rendering instead of c_X:

$$d_X = \frac{c_X w_X}{\sum_{\text{all } \tilde{X}} c_{\tilde{X}} w_{\tilde{X}}}$$

To emphasize or suppress linear, planar, or spherical structures the user can modify the classification. The weight coefficients w_X can be adjusted by selecting a position in a triangle. The barycentric coordinates of the position define the weight coefficients. The triangle widget and the effect of different weight vectors can be seen in Figure 3.2.

3.3.3 Lighting

The color of the three manifold types is obtained by:

$$\boldsymbol{C}_X = \boldsymbol{M}_X^\mathrm{D}\boldsymbol{F}^\mathrm{A} + K_X^\mathrm{D}\boldsymbol{M}_X^\mathrm{D}\boldsymbol{F}^\mathrm{D} + K_X^\mathrm{S}\boldsymbol{M}_X^\mathrm{S}\boldsymbol{F}^\mathrm{S}$$

where $\boldsymbol{F}$ is the light color, $\boldsymbol{M}_X$ is the material color, and K_X the shading contribution; the superscripts $^\mathrm{A}$, $^\mathrm{D}$, and $^\mathrm{S}$ denote ambient, diffuse, and specular components respectively.

In the planar case, the eigenvector corresponding to the smallest eigenvalue, $\boldsymbol{v}_0$, is normal to the planar surface. The sign of the eigenvector is not determined. The normal vector is chosen to show in the direction of the viewing vector:

$$\boldsymbol{n} = \mathrm{sign}(\boldsymbol{v}_0 \cdot \boldsymbol{V})\boldsymbol{v}_0$$

where $\boldsymbol{V}$ is the viewing vector.

With the derived normal vector a normal-vector-based illumination model can be applied. In the presented system the Blinn-Phong lighting model [Bli77] is used:

$$\begin{aligned} K_\mathrm{p}^\mathrm{D} &= \max(\boldsymbol{n} \cdot \boldsymbol{L}, 0) \\ k_\mathrm{p} &= \boldsymbol{n} \cdot \boldsymbol{H} \\ K_\mathrm{p}^\mathrm{S} &= \begin{cases} (k_\mathrm{p})^q & \text{if } k_\mathrm{p} > 0 \text{ and } K_\mathrm{p}^\mathrm{D} > 0 \\ 0 & \text{else} \end{cases} \end{aligned}$$

where $\boldsymbol{L}$ is the light vector, $\boldsymbol{H}$ the halfway vector, and q the specular exponent.

For the linear case, setting the eigenvector corresponding to the largest eigenvalue $\boldsymbol{v}_2$ as the tangent vector, the diffuse and specular components can be calculated according to

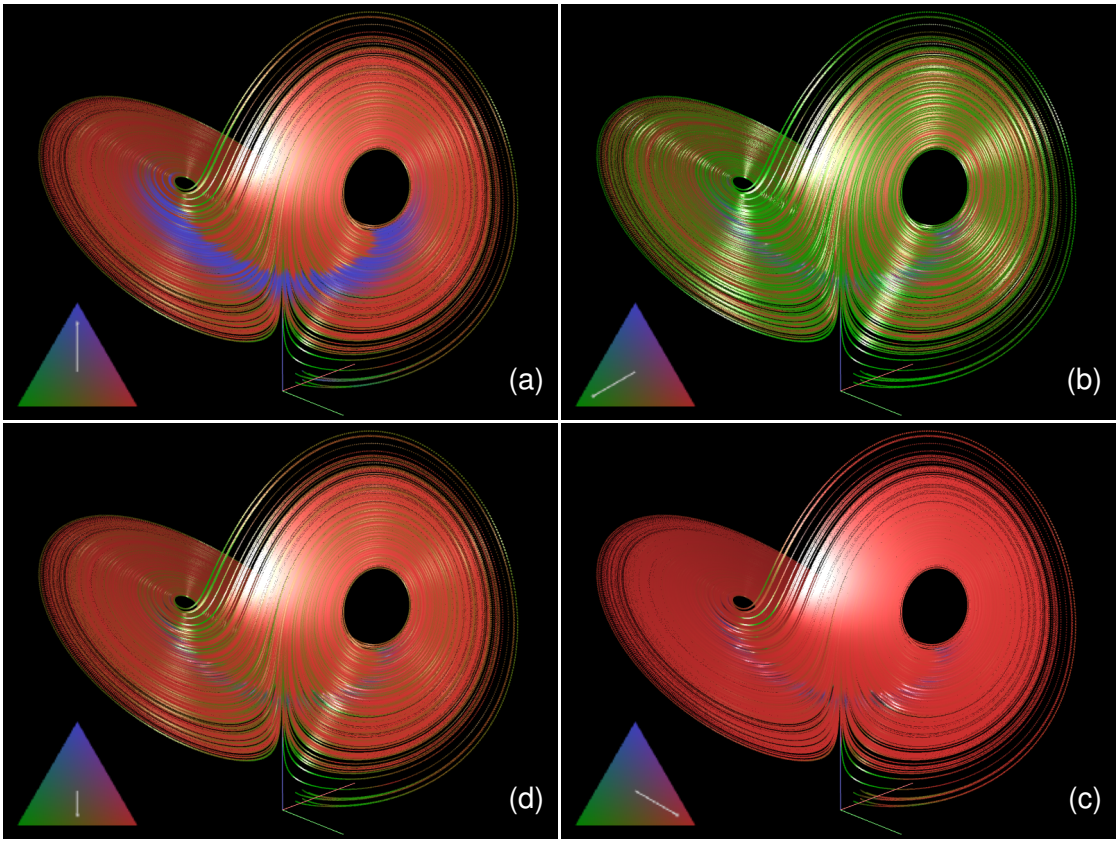


Figure 3.2: Scatter plots showing the effect of modifying the weight factors for the linear, planar, and spherical cases. The triangle in the lower left corner of each plot shows the user input. (a) emphasizing spherical structures; (b) emphasizing linear structures; (c) emphasizing planar structures; (d) suppressing spherical structures. The color coding is as follows: red corresponds to planar, green to linear, and blue to spherical structures.

Zöckler et al. [ZSH96]:

$$
\begin{aligned}
K_l^D &= \sqrt{1-(\boldsymbol{L}\cdot\boldsymbol{v}_2)^2} \\
k_l &= \sqrt{1-(\boldsymbol{L}\cdot\boldsymbol{v}_2)^2}\sqrt{1-(\boldsymbol{V}\cdot\boldsymbol{v}_2)^2}-(\boldsymbol{L}\cdot\boldsymbol{v}_2)(\boldsymbol{V}\cdot\boldsymbol{v}_2) \\
K_l^S &= \begin{cases} (k_l)^q & \text{if } k_l > 0 \\ 0 & \text{else} \end{cases}
\end{aligned}
$$

For the spherical case a surface cannot be determined, and only a constant term is applied to achieve a global intensity comparable to diffuse shaded linear or planar

structures. The contribution is user-specified and constant over the image:

$$K_s^D = \text{const.}$$
$$K_s^S = 0$$

3.3.4 Kernel Size

The number of neighbor samples for the covariance computation affects the eigenvalues and the eigenvectors. The size of the kernel h_i^n (see Equations (3.2) and (3.3)) acts as the support size of a lowpass filter. The effect of using different kernel sizes can be seen in Figure 3.3. Starting with one sample the eigenvectors/eigenvalues are undefined, resulting in the upper left plot. Here, equal eigenvalues are assumed leading to the spherical classification shaded with ambient blue. With two samples, the largest eigenvector is defined resulting in the linear case, which is shaded in green color. With four and more samples, all eigenvalues and eigenvectors are defined resulting in a combination of the three cases. With increasing kernel size, the eigenvalues and eigenvectors are lowpass-filtered with decreasing frequency spectrum, reducing noise but at the cost of blurring over the samples (see the last plot with $n = 2^{13} = 8192$ samples). In the real-time application, the neighborhood size is interactively specified by the user.

3.3.5 Boundaries

Boundaries can be of special interest in many situations. This section discusses how the method behaves at manifold boundaries.

To analyze the behavior at boundaries of volumetric distributions consider a box-shaped volume with uniform volumetric sample distribution. Inside the volume, all three eigenvalues are similar in size, which results in the volumetric classification $c_l \approx 0$, $c_p \approx 0$, $c_s \approx 1$. As we move toward one face of the volume the kernel covers a growing region outside the volume, where there are no samples to contribute to the covariance matrix. At the boundary, the classification will be $c_l \approx 0$, $c_p \approx 0.5$, $c_s \approx 0.5$, which is half volumetric and half planar.

When moving toward an edge of the box, the classification tends toward the linear case. Both cases can be illuminated according to the light direction in contrast to the volumetric case, which helps recognize the boundary of the volume (see Figure 3.4).

A similar effect occurs in edge areas for points distributed uniformly on a plane. When approaching the edge, the eigenvalues that are orthogonal to the edge diminish, which results in the linear case.

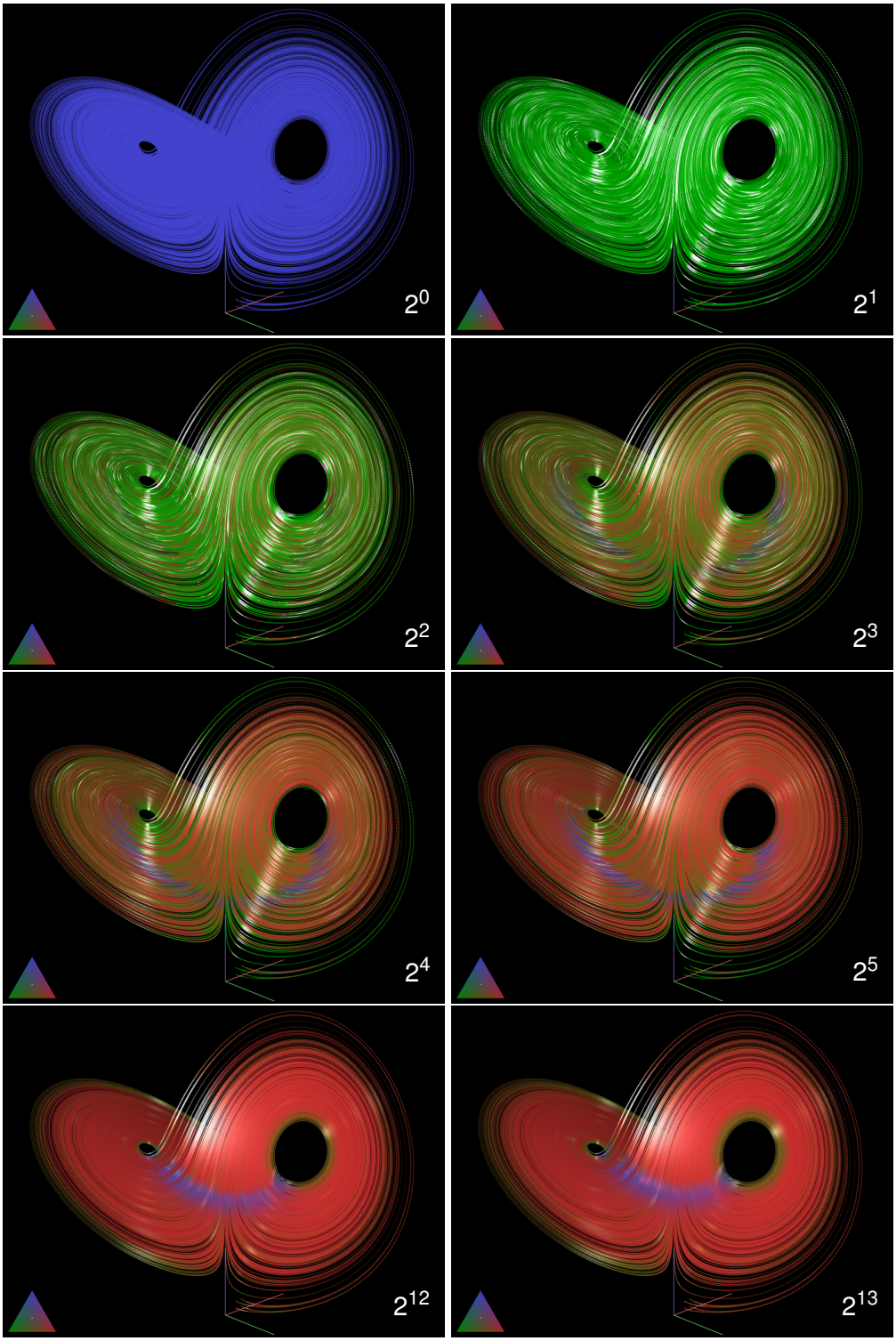


Figure 3.3: Scatter plots with different kernel sizes.

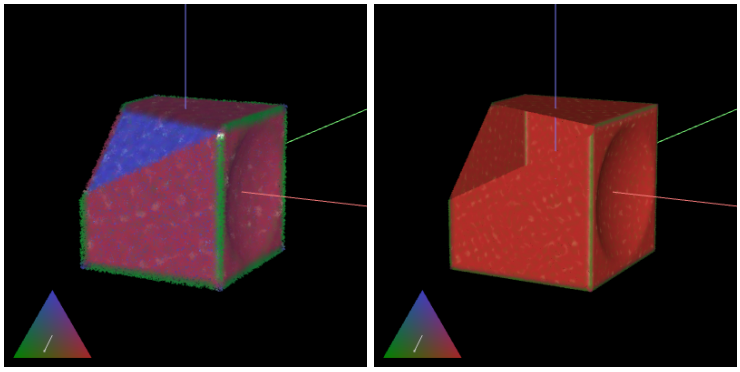


Figure 3.4: Scatter plot showing samples uniformly distributed in a boolean object consisting of a cube from which a sphere was subtracted (left) and on 2D surfaces of the same object (right). The edges of the box as well as the faces are distinguishable and illuminated by the algorithm (left). Similarly, the boundaries of the 2D surfaces are distinguishable as well (right). An oblique clip plane is applied to show the interior of the data sets.

3.3.6 Rendering

The data is rendered as points. In the following, two rendering variants are proposed. The first variant uses opaque points. With depth test enabled, just the nearest sample is visible at each sample location, providing an easy to understand image. Sometimes structures need to be examined which are occluded by other opaque points. One way of exploring the interior is by removing occluders. Typical approaches include clipping or cutaways [DWE03]. In Figure 3.4, a planar clipping geometry is used.

The second variant renders the data as a volume with the emission-absorption model, which is widely applied in direct volume rendering [EHK+06]. In general blending is not commutative; therefore, spatial sorting of the data would be necessary. However, there are order-independent blending methods that are effective for volume visualization (see Mora and Evert [ME04]). In this work, the special case of an additive blending model has been chosen, drawing the points semitransparent and blending all samples rasterized to the same location. Figure 3.5 shows the results of additive blending. The density of the samples is clearly shown, and additional illumination provides further shape cues.

3.3.7 Color Mapping

In general, color mapping can be used to encode additional data or information. In the previous examples, color was used to highlight the differently classified structures in the data by assigning green, red, and blue colors to $\boldsymbol{M}_{\mathrm{l}}^{\mathrm{D}}$, $\boldsymbol{M}_{\mathrm{p}}^{\mathrm{D}}$, and $\boldsymbol{M}_{\mathrm{s}}^{\mathrm{D}}$ respectively. The three material colors are chosen to have equal luma and to result in gray if mixed in equal parts

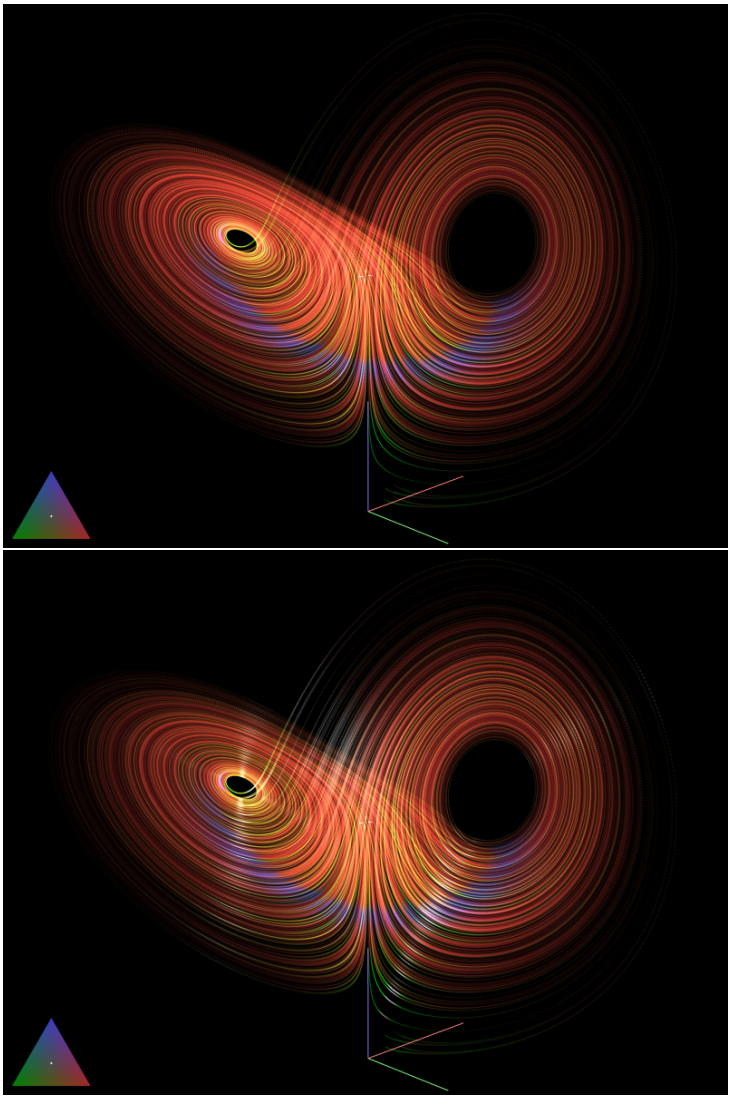


Figure 3.5: Visualization of the Lorenz attractor with additive blending. Upper: without illumination; lower: with illumination.

in non-linear RGB space. In this way, manipulating the weights w_X does not affect the brightness substantially.

Illumination provides the visual cues to recognize the shape of the data through the luminance channel. Therefore, the two chromatic channels can be used to display additional data dimensions $\boldsymbol{x}_i$. The diffuse components for the three cases are mapped to the same value depending on $\boldsymbol{x}_i$. That is $\boldsymbol{M}_{\mathrm{l}}^{\mathrm{D}} = \boldsymbol{M}_{\mathrm{p}}^{\mathrm{D}} = \boldsymbol{M}_{\mathrm{s}}^{\mathrm{D}} = m(\boldsymbol{x}_i)$, where m should map to isoluminant colors, avoiding interaction with the illumination. The motivation for this design choice is that the chromatic channels are suitable for visual grouping, whereas the achromatic channel serves to facilitate shape perception [War04, KSJ95]. For example, visual grouping with color mapping is applied in Section 3.7.

The color selection presented here is just an estimate since the perceived brightness also depends on the environment and the calibration of the display device, and also varies between users. An approach to derive completely isoluminant colors based on user perception on a specific device can be found in [KRC02] and is applied in Chapter 6.

3.4 Implementation

The application is implemented in C++ and employs OpenGL. The test platform was a PC with Core 2 Quad Q6600 CPU, 4 GB DDR2 main memory, and GeForce 8800GTX GPU.

Since the calculation of the covariance matrix at each point is computationally demanding with growing neighborhood, the application is separated into a pre-processing part and a real-time part.

In the pre-processing step, the weighted covariance matrix is calculated for different neighborhood sizes. As neighborhood size powers of 2 up to $2^{13} = 8192$ samples are used. To efficiently perform searching for the n nearest neighbors a spatial data structure is needed. Here an octree is used for its simplicity. Alternatively, a k-d tree [Ben75] could be used, as done by Camamarano and Jensen [CJ02] for a related point search problem in photon mapping. The algorithm has asymptotic complexity $\mathcal{O}(Nn\log(N))$, where n is the neighborhood size and N the number of samples. In the prototype implementation, pre-processing takes 52 seconds for the Lorenz attractor sampled with 800 k points shown in Figures 3.1 and 3.2 with $n \in \{2^i | i \in [0..6]\}$ neighborhood size.

During runtime, the system is capable of interactively changing the size of the kernel n in Equations (3.2) and (3.3) by selecting from the pre-calculated neighborhood sizes. The system also supports interactivity by efficient point rendering; for example, the rendering part constantly runs over 100 fps with 4x multisampling enabled on a 1600×1200 viewport with data sizes up to 800 k points.

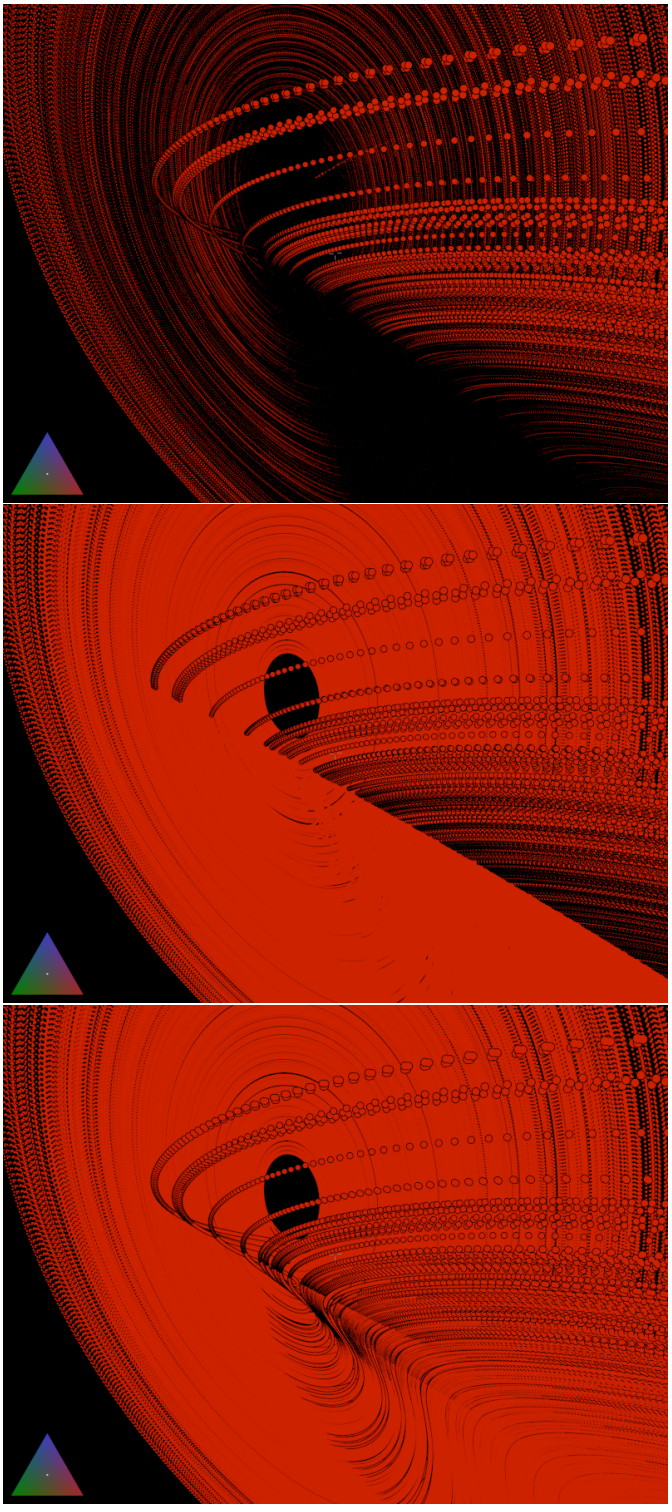
3.5 Halo Rendering at Depth Discontinuities

Halos have been applied to 3D scatter plots for a while [KSH04] [PKH04]. Drawing halos around points helps identify individual points even if they partially overlap. This works in low point density regions, where clutter is not too high. Naive drawing of halos around each point has limitations in high density areas. If the points are getting very close to each other halos occlude the points, simply because halos are bigger, see Figure 3.6(a). A simple solution is to leave away halos if the depth value exceeds a certain threshold, see Figure 3.6(b). In this section a more advanced solution is presented.

Edge-cuing techniques, presented by Tarini et al. [TCM06] for molecular visualization, can be adapted to 3D scatter plot visualization. An edge-cuing technique can be implemented with only a slight modification of the naive halo rendering: by applying a backward translation to the OpenGL projection matrix while drawing the halos. In the following we term the original unmodified projection matrix used to render the points $\boldsymbol{P}$ and the modified matrix used to render the halos $\boldsymbol{P}^{\text{new}}$. This renders halos just at significant depth discontinuities and facilitates shape perception. The benefits of this approach compared to the one of Tarini et al. are the simplicity, not requiring modifications to the rendering—like modifying depth values in a fragment shader, which prevents early Z checking—and the native support for multisampling buffers—allowing full-scene anti-aliasing (FSAA)—since the coverage values are calculated just like without the edge-cuing technique.

However, the modification of the projection matrix only works that easily for orthographic projection. For orthographic projection, the translation with a user defined constant d in eye space defining the depth discontinuity threshold for halo rendering is applied first, followed by the orthographic projection $\boldsymbol{P}$. It turns out that only one matrix entry of the `GL_PROJECTION` matrix is changed, the entry which couples the z-value of the output vector with the w-value of the input vector: $\boldsymbol{P}^{\text{new}}_{zw} = \boldsymbol{P}_{zw} + \frac{2d}{f-n}$, where n and f are the near and far clipping planes. This means that the transformation can also be decomposed as the orthographic projection $\boldsymbol{P}$ which is applied prior to a translation with $\frac{2d}{f-n}$ in z direction.

The application of a translation prior to projection would also alter the screen coordinates of the points when using perspective projection. Therefore, it is reasonable to model the translation after the projection. It is desirable to obtain halos similar to the orthographic projection, to be able to switch between the two projection methods more seamlessly. The application of the translation with $\frac{2d}{f-n}$ after the projection results also in only one difference to the `GL_PROJECTION` matrix, namely the entry which couples the z-value of the output vector with the z-value of the input vector: $\boldsymbol{P}^{\text{new}}_{zz} = \boldsymbol{P}_{zz} - \frac{2d}{f-n}$. Unlike for the orthographic case with the transformation constructed this way the halos depend on the near and far clipping planes. However, the technique is easy to implement, delivers quite reasonable results and provides a strong improvement compared to previous methods, see Figure 3.6(c).



(a)
halos are drawn everywhere

(b)
halos are left away for a certain depth value to minimize clutter

(c)
demonstrates the use of the proposed halo rendering technique which renders halos only at significant depth discontinuities

Figure 3.6: Comparison of halo rendering methods.

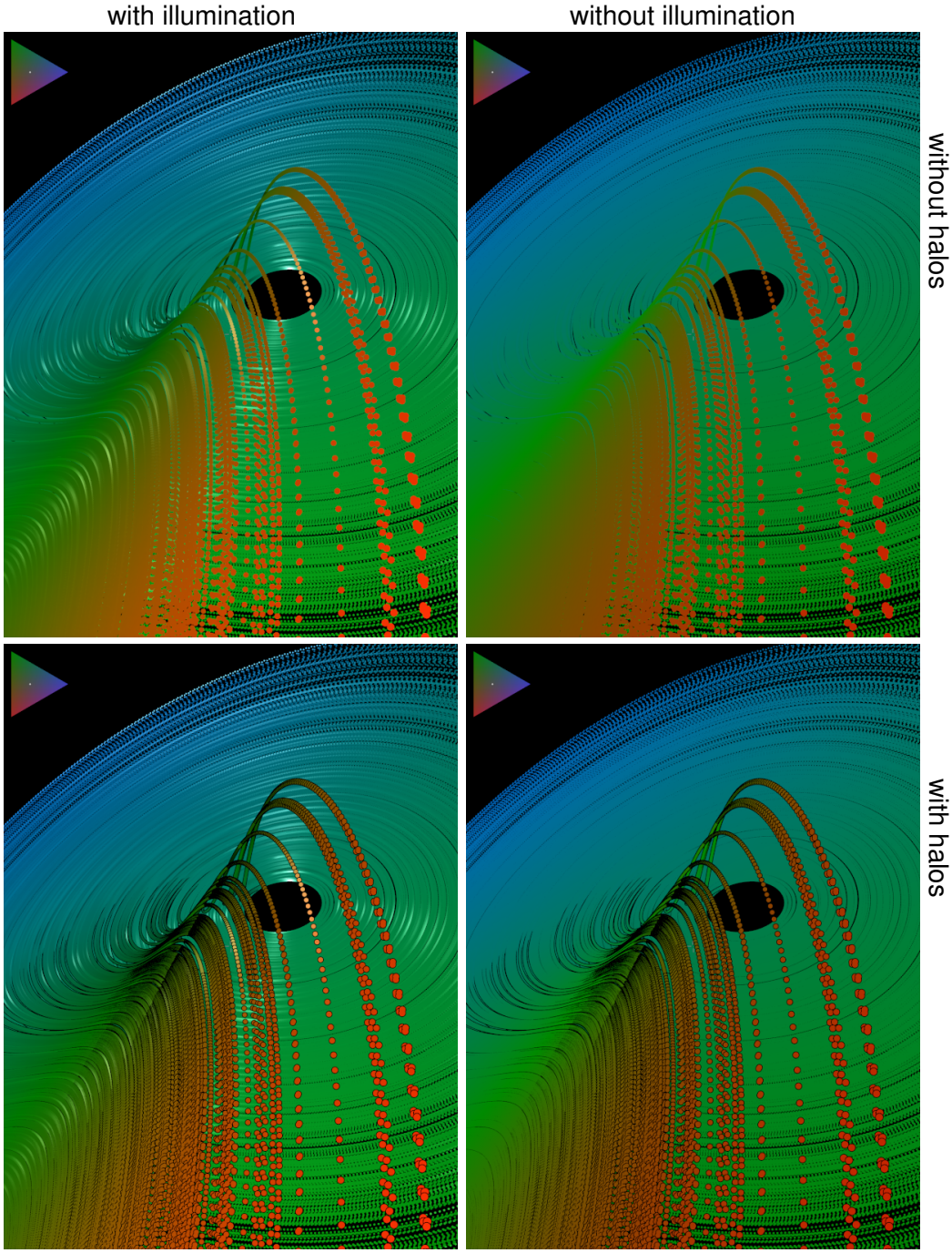


Figure 3.7: Mapping depth to color: near to red, far to blue. Point size is adjusted according to depth.

3.6 Combination and Comparison with State of the Art Methods

In Section 3.3.7 color mapping was introduced. The quantity which color is mapped to can also be a view-dependent attribute, like depth. The application of color mapping to depth is shown in Figure 3.7, which helps distinguish points having substantially different depth. Color mapping is effective even in the case of unconnected regions. Additionally point size can also be adjusted depending on the depth values, supporting point identification in the foreground.

Drawing halos around points makes them distinguishable in low density areas. The introduced halo rendering technique presented in the last section minimizes clutter in high-density areas by leaving away the halos. In contrast, illumination is well suited to support shape perception in high-density regions. As stated earlier, the focus here is on improving shape perception, which can be combined with complementary techniques like depth-based color coding, see Figure 3.7.

3.7 Examples

In this section, the application of the introduced visualization method to several examples of multi-dimensional data and even example applications beyond 3D scatter plots are presented.

The first example shows the *covtype* data set from the UCI Machine Learning repository (`http://archive.ics.uci.edu/ml/`), which is a 55-dimensional data set of forest cover. The FastMap algorithm is used to map the data set to 3-space. FastMap [FL95] maps points from n-dimensional space to k-dimensional space ($k \leq n$) with the focus on preserving distances between points. Figure 3.8 shows examples of the 3D scatter plot visualization, where the four wilderness areas are mapped to different colors. The two images in Figure 3.8 compare scatter plot visualization with and without illumination. This comparison demonstrates that illumination substantially improves the perception of the surface shapes surrounding the different clusters. The animation in the accompanying video (`covtype.avi`) further improves spatial perception by providing structure from motion.

The next example illustrates 3D scatter plots that can be used for the design of 3D transfer functions in direct volume visualization. According to Kniss et al. [KKH02], the first and second derivatives of 3D scalar fields allow for a better design of transfer functions. However, in their paper, they only show 2D scatter plots (in fact, 2D histograms), but not the full 3D structure of the data distribution. Figure 3.9 and the accompanying video (`bucky.avi`) illustrates the 3D scatter plot of the "bucky-ball" data set, a typical test data set in volume visualization with some 2 M voxels. With

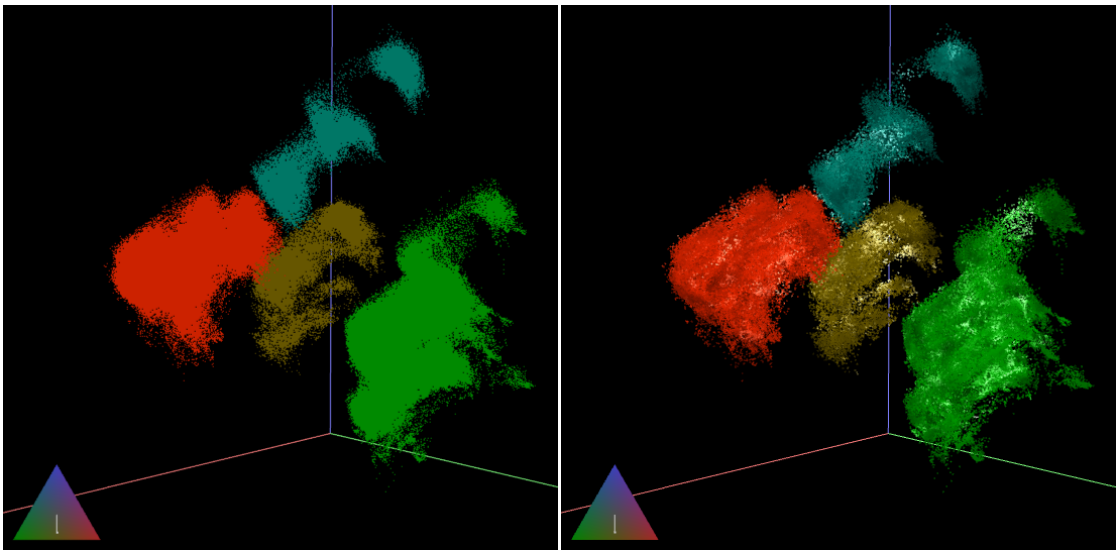


Figure 3.8: Scatter plot showing the covtype data set mapped to 3-space by the FastMap algorithm. Colors represent different wilderness areas. Left: without illumination; right: with illumination.

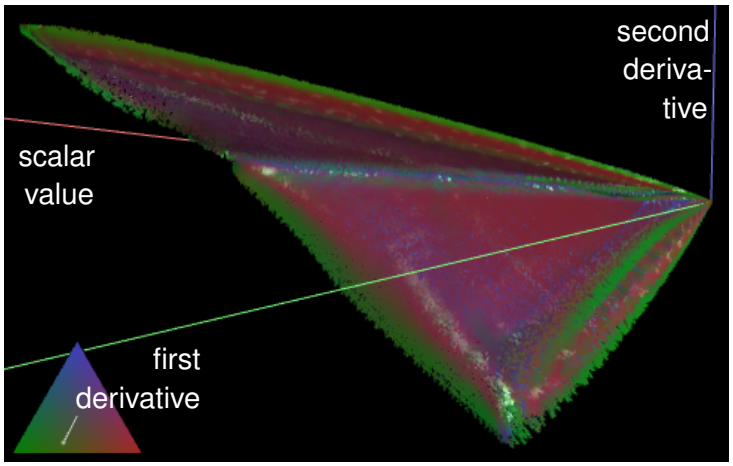


Figure 3.9: Bucky-ball data set showing the first and second derivatives over the scalar values (green, blue, and red axes respectively).

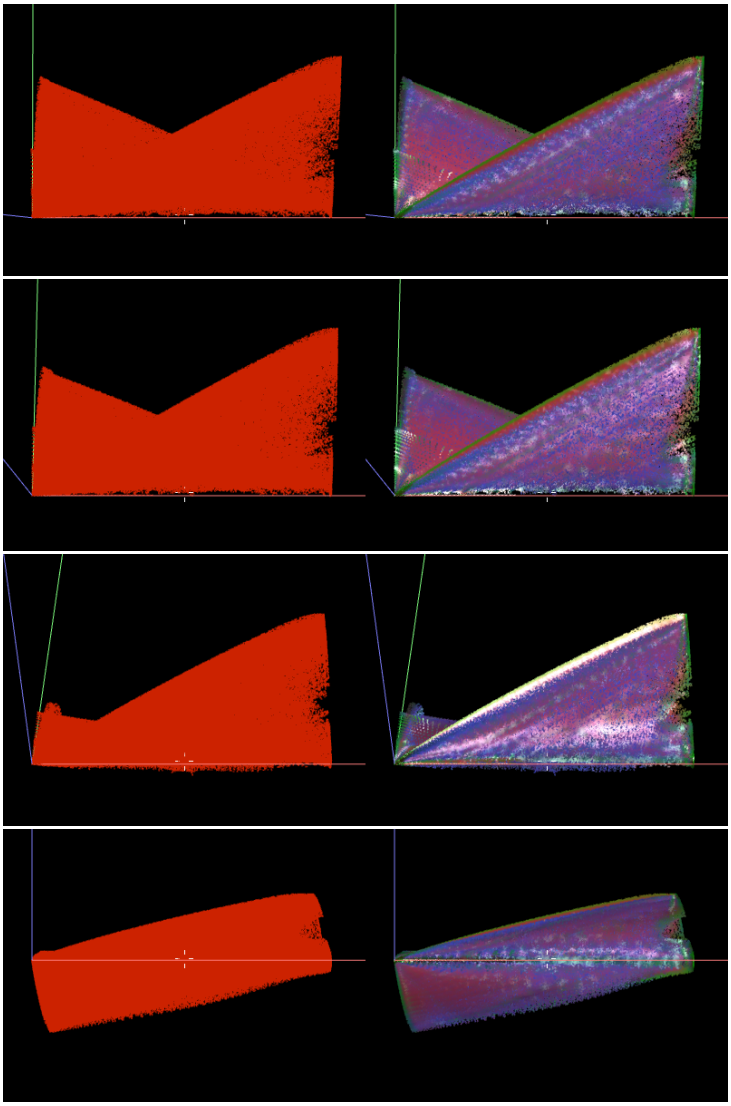


Figure 3.10: Several time steps from an animation displaying the transition between two scatter plot displays. The bucky-ball data set is visualized without illumination (left) and with illumination (right). The top image shows the first derivative and the scalar value. The bottom image shows the second derivative and the scalar value.

Figure 3.11: Virgo data set, rendered with additive blending. Green, red, and blue colors are used for linear, planar, and spherical structures respectively. Lighting is disabled.

illuminated 3D scatter plots the three quantities can be displayed simultaneously. In particular, the two "branches" of the data distribution with different signs of the second derivative become visible only in the 3D scatter plot.

Another application related to 3D scatter plots has recently been published by Elmqvist et al. [EDF08]. They use rotation to animate (see Capter 4) between 2D scatter plots and navigate through a scatter plot matrix. This approach could also benefit from illuminated 3D scatter plots by better spatial and shape perception, as illustrated in Figure 3.10 and in the accompanying video (`bucky.avi`).

Finally, the presented technique can be applied to point data that does not originate from a scatter plot. Figure 3.11 depicts a subset of the astrophysical Virgo data set consisting of 925 k points. The data set was rendered with additive blending. The data set has no large structures where lighting would be beneficial. Regions where stars are linearly or planarly structured can be distinguished based on color corresponding to the classification.

CHAPTER 4

3D SCATTER PLOT NAVIGATION

While the previous chapter has shown how to improve the depth perception of individual 3D scatter plots, here a novel technique is introduced, which can be used to navigate between 3D scatter plots. The technique makes strong use of human depth perception by relying especially on solid body rotations and the kinetic depth effect. The technique was presented in a similar form in [SW12].

Continuous transition between 2D scatter plots helps preserving context. This benefit has been observed before and successfully applied to the navigation between arbitrary 2D projections [FFT74] and to 2D scatter plot matrix navigation [EDF08] for multivariate data visualization. This chapter addresses the navigation between 3D scatter plots especially in the context of 3D scatter plot matrix navigation. A key feature of the presented technique is that it facilitates interpolation between arbitrary 3D scatter plots as long as one axis mapping is preserved.

The techniques mentioned above for 2D scatter plots share one interesting property: the motion of points during the transition from one view to another looks like a 3D rigid body rotation. This greatly helps in tracking the motion of points. However, most of the 2D projections lose this advantageous property when directly extended to 3D scatter plot interpolation. This issue is addressed by deriving a set of projections ensuring that the interpolation appears as a 3D rigid body rotation. The results of a controlled user study with 12 participants will be presented in this chapter showing that the new 3D rotation approach is significantly better than direct animation between scatter plot views for perceiving the correspondence of points in different scatter plots.

The usefulness of the novel approach is also demonstrated for a typical application in natural language processing. Here, explorative data analysis with navigation in 3D scatter plot matrices is useful for developing feature extraction methods.

4.1 Background

4.1.1 Scatter Plot Navigation

Two-dimensional scatter plots and scatter plot matrices are widely used in multivariate data analysis and provided by different software images like PRIM-9 [FFT74], Xmdv-Tool [War94], Polaris [STH02], XGobi [SCB98], or GGobi [SLBC03]. This section covers previous work related to explorative multivariate data analysis with scatter plots and classifies the techniques according to the taxonomy introduced in Table 4.1. This table reveals white spots addressed in this chapter (indicated by "HERE" in the table).

Table 4.1: Taxonomy of interactive scatter plot techniques for multivariate data analysis. Methods with mapping transitions that appear as rigid body rotations are underlined. Combinations that are not in focus of the specific technique are set in brackets.

viewport →	2 axes	3 axes	3 axes
axes mapping ↓	no rotation	no rotation	rotation
fixed	2D	2D+sorting	3D
1 axis swap	Roll. Dice	HERE	HERE
2 axes swap	(HERE)	HERE	HERE
1 axis manipulation	PRIM-9		
2 axes manipulation	GGobi / star coord.		
all dimensions	GT	(GT)	3D-GT

The taxonomy differentiates the techniques according to the number of axes and viewport interaction possibilities ("viewport" in the table) and the axes mapping ("axes mapping"). 2D scatter plots ("2D") have two—usually screen-aligned—axes assigned to different data dimensions and no viewport interaction in sense of rotation. In contrast, 3D scatter plots ("3D") display three data dimensions assigned to three (orthogonal) axes. The third dimension allows for 3D user interaction (with 2 or 3 rotational degrees of freedom) to change the viewport. Since computer screens present a 2D image, 3D structures need be conveyed by depth cues such as structure from motion or occlusion, which are easily supported by 3D scatter plots.

Even for 2D scatter plots, occlusion can be implemented by using another data dimension for "depth" sorting ("2D+sorting"). This can also be compared to a full 3D scatter plot where the only difference is the missing rotational interaction in 3D. This mapping also requires three axes: two for spatial position and one for sorting. Actually, to define a simple sorting order a depth is not required considering the different scales of measurement [Ste46]: nominal, ordinal, interval, and ratio scale. Depth is defined in a ratio scale and sorting is only ordinal; however, with a data dimension used to define a

sorting order, a ratio scale is given. 2D scatter plots are often used in conjunction with color coding and size adjustment. Necessarily overdraw occurs, even if the drawing order is not related to any data attribute. If the features are interpreted as depth cues, they possibly contradict. However, occlusion is the most dominant depth cue, and occlusion requires only an ordinal relation. Therefore, term "2D+sorting" is used in the taxonomy, which is meaningful even for 2D scatter plots where a depth is not defined. It should be recognized that changing the drawing order of points in a 2D scatter plot may be highly rewarding, because we can benefit from the occlusion depth cue, especially if the points are drawn as sprites and their boundary can be recognized. A similar argumentation was presented by Ware to describe the 3D nature of overlapping windows in current graphical user interfaces (GUIs) [War04]. Finally, if a renderer already is using occlusion and happens to define the sorting order using a data dimension, why should it not use other possible depth cues? Therefore, according to the taxonomy "2D+sorting" also includes full 3D scatter plots without the possibility to rotate the viewport relative to the scatter plot.

Flexibility in data mapping is introduced by extending the fixed mapping between certain data dimensions and scatter plot axes ("fixed" in the table) to a flexible exchange of mapped dimensions, either by swapping the dimensions of one or two axes ("swap") or by manipulation of dimensions ("manipulation").

The main difference between the "manipulation" and "swap" approaches, besides the interaction methods, is the way in which dimensions can be assigned to the axes. The "swap" approaches only assign one data dimension to each of the axes, respectively—except for the transitions, which serve only as a context-preserving mechanism; interpretation does not take place during transitions—whereas the "manipulation" approaches assign a linear combination of dimensions to the axes. Because of this more general mapping, the projections that can be generated by "manipulation" approaches are a super-set of the projections producible by "swap" approaches, but interpreting linear combinations of dimensions is much harder.

PRIM-9 [FFT74] allows one data axis to be manipulated at a time, whereas GGobi [SLBC03] allows manipulation of both axes in its 2D tour, which is an implementation of the technique introduced by Cook and Buja [CB97]. While interacting with the scatter plot in GGobi, the motion of the points can be perceived as 3D rigid body rotation. However, in contrast to the novel technique presented here, just one data dimension can be actively manipulated at a time; the other data dimensions change accordingly to maintain a valid projection by staying in a plane orthogonal to the manipulated dimension. Additionally, points are not sorted according to a third axis, but simply drawn in the order they are defined. This is not an issue if the points are drawn in one single color, but if more colors are used, 3D perception can suffer because the depth perception from the kinetic depth effect may be overridden by the depth cue from occlusion that can be detected from the differently colored points. With the technique presented here, a

third axis that is consistent with the motion of the points is always used to support 3D perception. Furthermore, the user is guided by using 3D scatter plot matrix navigation.

Star coordinates ("star coord.") introduced by Kandogan [Kan01] also produce plots similar to early versions of XGobi, see Cook et al. [CBCH95], which are biplots introduced by Gabriel [Gab71]. In later versions of GGobi the axes of the biplot have been extracted to the "axis tree" [CS07]. However, Kandogan presents a regular star shaped default dimension arrangement—like RadViz [HGM+97] introduced by Hoffman et al.—and an interaction technique similar to Cook and Buja's [CB97]. However, the manipulated dimension does not adjust the other dimensions mapping, which means, he gives up the rigid body rotation analog. Interestingly Kandogan does not mention any of these highly related works. According to the taxonomy, star coordinates manipulate the mapping of one dimension to 2 axes and have a 2D viewport and manipulations do not maintain the rigid body property.

With the grand tour [Asi85, BA86] ("GT"), the user is presented a sequence of projections that come arbitrarily close to any 2D scatter plot projectable from the multidimensional data set, with smooth in-between transitions. However, there is no user interaction involved, the axis mapping changes all dimensions during transition, and rigid body motion is not ensured when interpolating between projections. To improve a given projection, a technique named projection pursuit can be used. The grand tour is often used in combination with projection pursuit. While the grand tour is used to get an overview of a data set, projection pursuit is used to improve an interesting projection. GGobi uses such a combination of the two techniques. However, GGobi is only concerned with 2D projections of a high dimensional data set. Nason [Nas95] introduces 3D projection pursuit and Yang [Yan99] applies the grand tour in 3D ("3D-GT").

Finally, Elmqvist et al. [EDF08] guide the user in navigating between 2D scatter plots by changing one axis mapping at a time with their Rolling the Dice ("Roll. Dice") method. Each of the axes shows one data dimension before the transition and one dimension after the transition, but the mapping of one axis is swapped during the transition. The navigation approach presented here extends their technique by using three axes and supporting the swap of one or two axes during transitions.

4.1.2 Perception

For the scenario presented here, the perception of 3D objects and motion perception play equally important roles. Animation is employed to convey correspondence between dots that represent the same data point in two different scatter plots following the principle of Elmqvist et al. [EDF08]. However, animation itself should be used with caution as a means of graphical representation as discussed by Tversky et al. [TMB02]. They show that, in general, animation can be useful when it follows the principles of congruence and apprehension, which implies that similar transitions should be grouped [HR07]. The technique introduced here follows this general approach and applies it to low-level

perception by employing animation in the sense of structure from motion processing where perception of motion is tightly and effectively linked with spatial perception. Early work by Ullman [Ull79] describes a computational approach to derive 3D structure from motion (on screen). It contains a constraint for the interpretation of structure from motion, which he calls the *rigidity assumption*. In particular, he proved that the 3D structure as well as the motion can be calculated given at least four points in at least three views. His experimental findings indicate that the human visual system uses a similar computational approach. As summarized by Ware [War08], numerous subsequent works showed that structure from motion indeed is a strong depth cue, potentially more informative than stereoscopic depth, and certainly more informative than all other monocular cues [KSJ00]. The presented technique relies on low-level perceptual processing to extract 3D structure from motion, as summarized by Todd [Tod04], and uses it for effective perception of moving 3D objects made of points in scatter plots.

If animation was used in a 2D setting for scatter plots and two axis mappings were changed simultaneously, data points would have to move on trajectories that differ from those of 2D projections of a single 3D rigid object. Therefore, these points would move in quite an incoherent manner. Although tracking of multiple objects is possible [PS88], the number of objects is limited to a small number (around 5 objects, however very much dependent on the application); see the overview paper by Cavanagh and Alvarez [CA05]. The perception of a single 3D rigidly moving object is associated with much less cognitive load than the independent motion of 2D points. This is the key observation and motivation for this work: the perception of correspondence between scatter plots is best supported by animation with coherent rigid body motion.

One serious drawback of 3D scatter plots, when using perspective projection, is the fact that the position of a point in the data domain cannot be accurately estimated based on its screen position since the screen position varies with the depth value. Perspective projection is intentionally not used here to make the scatter plots look like 2D scatter plots when viewed along the major axes. From such static viewpoints, without interactive rotation, there are no disadvantages from 3D but only benefits: the advantage of a depth axis that provides a depth sorting is retained—this depth information is conveyed to the user through occlusion and is enhanced by the halo rendering technique presented in Section 3.5 and used here—but there is not any negative impact of 3D. However, it should be mentioned that perspective projection itself is beneficial to perception [WB08].

4.2 Interpolation Scheme and Projection Technique

This section presents a novel interpolation scheme and projection technique that enable the transition between 3D scatter plots. Assuming the scatter plots share one common data axis, the presented technique guarantees that the transition is perceived as a 3D rigid body rotation.

4.2.1 Interpolation Scheme

The interpolation scheme is used to interpolate between different 3D scatter plots. Mathematically, the interpolation scheme is a projection with one interpolation parameter, which projects from the original m-dimensional data domain to 3D space, as formulated by a generic projection

$$\boldsymbol{T} : \mathbb{R}^m \longrightarrow \mathbb{R}^3$$

Let us assume without loss of generality that the transition starts from the following projection:

$$\boldsymbol{T}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots \end{bmatrix}$$

In one case, only one of the dimensions needs to be changed. Without loss of generality, let us consider the transition to the following projection:

$$\boldsymbol{T}_0 \Rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots \end{bmatrix}$$

In the other case, two of three dimensions need to be changed. Here, let us consider the transition to the following projection:

$$\boldsymbol{T}_0 \Rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 1 & 0 & \dots \end{bmatrix}$$

A smooth transition between the 3D scatter plots is the goal. Therefore, let θ be the transition parameter which is interpolated between 0 and $\pi/2$ and (4.1) and (4.2) the matrices which are used to interpolate the positions of the points:

$$\boldsymbol{T}_1^\theta = \begin{bmatrix} \cos\theta & 0 & 0 & \sin\theta & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots \end{bmatrix} \tag{4.1}$$

$$\boldsymbol{T}_2^\theta = \begin{bmatrix} \cos\theta & 0 & 0 & \sin\theta & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & \cos\theta & 0 & \sin\theta & 0 & \dots \end{bmatrix} \tag{4.2}$$

The interpolation defined by (4.1) performs a transition from $\boldsymbol{M}_0$ to $\boldsymbol{M}_1$. *Proof:* $\boldsymbol{T}_1^0 = \boldsymbol{M}_0$ and $\boldsymbol{T}_1^{\pi/2} = \boldsymbol{M}_1$. The interpolation defined by (4.2) performs a transition from $\boldsymbol{M}_0$ to $\boldsymbol{M}_2$. *Proof:* $\boldsymbol{T}_2^0 = \boldsymbol{M}_0$ and $\boldsymbol{T}_2^{\pi/2} = \boldsymbol{M}_2$.

As an example, consider the point $\boldsymbol{p}$ in the m-dimensional data domain and let p_i be the i-th component of $\boldsymbol{p}$. We perform a change of one dimension by using the transition

$\boldsymbol{T}_1^\theta$. During the transition, $\boldsymbol{p}$ is projected to the point $\boldsymbol{q}_\theta$ in the 3D scatterplot: $\boldsymbol{q}_\theta = \boldsymbol{T}_1^\theta \boldsymbol{p} = [p_1 \cos\theta + p_4 \sin\theta, p_2, p_3]^{\mathrm{T}}$. The transition starts at $\theta = 0$, where $\boldsymbol{q}_\theta = [p_1, p_2, p_3]^{\mathrm{T}}$, and ends at $\theta = \pi/2$, where $\boldsymbol{q}_\theta = [p_4, p_2, p_3]^{\mathrm{T}}$. From the trigonometric expressions in the coordinate functions of $\boldsymbol{q}_\theta$ it becomes clear that the point does not simply move on a linear path from the one position to the other with constant velocity during the transition.

4.2.2 Projection Technique

The interpolation scheme presented above ensures that interpolation from one 3D scatter plot to another can be performed, but it does not guarantee that the transition looks like a 3D rigid body rotation. This section shows that there are projective transformations that make the interpolation look like a 3D rotation of a rigid body, after projection to the image plane.

Let $\boldsymbol{P}_z$ be the orthogonal projective transformation that projects to the $z = 0$ plane:

$$\boldsymbol{P}_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

This projection operator is applied to model the projection to 2D image space. Furthermore, $\boldsymbol{R}_u^\gamma$ denotes the rotation around the u-axis in $\mathbb{R}^3$ with angle γ.

The interpolation schemes ($\boldsymbol{T}_1^\theta$ and $\boldsymbol{T}_2^\theta$) together with the following projective transformations are equivalent to a rigid body rotation of points $\boldsymbol{x} \in \mathbb{R}^3$ (with the angle θ around the y-axis) followed by the projective transformation $\boldsymbol{P}_z \boldsymbol{R}_z^\alpha$:

$$\tilde{\boldsymbol{x}} = \boldsymbol{P}_z \boldsymbol{R}_z^\alpha \boldsymbol{R}_y^\theta \boldsymbol{x} \tag{4.3}$$

Theorem 1. *The interpolation scheme $\boldsymbol{T}_1^\theta$ followed by the projective transformation $\boldsymbol{P}_z \boldsymbol{R}_z^\alpha \boldsymbol{R}_x^\beta$ is equivalent to the rigid body rotation followed by the projection in (4.3):*

$$\boldsymbol{P}_z \boldsymbol{R}_z^\alpha \boldsymbol{R}_x^\beta \boldsymbol{T}_1^\theta \boldsymbol{d} = \boldsymbol{P}_z \boldsymbol{R}_z^\alpha \boldsymbol{R}_y^\theta \boldsymbol{x}$$

Equivalence is defined as follows: for fixed α and β and for each data item in the high-dimensional vector space, $\boldsymbol{d}$, there exists a point $\boldsymbol{x}$ for which the equation holds for all $\theta \in \mathbb{R}$.

Theorem 2. *The interpolation scheme $\boldsymbol{T}_2^\theta$ followed by the projective transformation $\boldsymbol{P}_z \boldsymbol{R}_z^\alpha \boldsymbol{R}_y^\beta$ is equivalent to the rigid body rotation followed by the projection in (4.3):*

$$\boldsymbol{P}_z \boldsymbol{R}_z^\alpha \boldsymbol{R}_y^\beta \boldsymbol{T}_2^\theta \boldsymbol{d} = \boldsymbol{P}_z \boldsymbol{R}_z^\alpha \boldsymbol{R}_y^\theta \boldsymbol{x}$$

Proofs of Theorems 1 and 2 are provided in Section 4.2.3.

Theorems 1 and 2 state that there are two degrees of freedom for the selection of the projection: α and β. Since general rotations have three degrees of freedom, one degree of freedom is lost, restricting the possible class of projections. The parameter α describes a rotation in the image plane, around the viewing axis. Since θ is interpolated from 0 to $\pi/2$, the user will perceive a corresponding 90 degree rotation around the y-image-axis, rotated by α in the image plane. If one axis changes, Theorem 1 implies that the other degree of freedom is perpendicular to the rotation direction during the interpolation because β describes a rotation around the y-axis (prior to the rotation with α). If two axes change, Theorem 2 indicates that the other degree of freedom is identical to the rotation direction during interpolation because β describes a rotation around the y-axis (prior to the rotation with α).

The practical implication of the above mathematical consideration is as follows. For consistent 3D rigid body rotations, one cannot start from any orientation of the initial 3D scatter plot and then make a transition to any target 3D scatter plot. In fact, the data dimension(s) to be exchanged during transition dictate(s) restrictions to axes orientations, in the sense that it / they define(s) the subspace of view directions from where the dimension(s) exchange will look like a rigid body rotation. In the case one axis mapping is changed, the changing axis needs to lie in the image plane; and in the case two axes are changed, the non-changing axis needs to lie in the image plane. One navigation approach could then perform an automatic view adjustment in the application while navigating between 3D scatter plots, ensuring that the restrictions of axes orientations are met. An alternative approach leaves the "snapping" to valid start orientation to the user.

The projection to the image plane $\boldsymbol{P}_z$ simply maps the depth component to zero. This implies that the depth component is not important at all, and this is the case for the kinetic depth effect. However, as mentioned before, providing additional depth cues when visualizing 3D data on a 2D screen is important for shape perception. The 3D scatter plot renderer implemented here uses occlusion and halo rendering to convey depth information. Therefore, not only the 2D screen positions of the points need to be consistent to the rigid body rotation but also the depth of the points prior to projection. The depth value consistent to the rigid body rotation is given by the z-component of $\boldsymbol{R}_z^{\alpha}\boldsymbol{R}_y^{\theta}\boldsymbol{x}$, see Section 4.2.3 for the calculation of $\boldsymbol{x}$.

Theorems 1 and 2 do not state that there is only one rigid body that is consistent with the point movements. In fact, there is another interpretation of this interpolation and projection: $\boldsymbol{P}_z\boldsymbol{R}_z^{\alpha}\boldsymbol{R}_y^{\beta}\boldsymbol{T}_2^{\theta}\boldsymbol{d} = \boldsymbol{P}_z\boldsymbol{R}_z^{\alpha}\boldsymbol{R}_y^{-\theta}\boldsymbol{x}'$ with $\boldsymbol{x}' = [\boldsymbol{x}_x\, \boldsymbol{x}_y\, -\boldsymbol{x}_z]^{\mathrm{T}}$. This reflects the fact that given the orthographic projection one cannot differentiate if one rigid body is rotated in one direction or a corresponding other rigid body is rotated in the opposite direction. However, using additional depth cues the interpretation is made unambiguous.

4.2.3 Proofs

This section will provide the proofs that the presented interpolation schemes ($\boldsymbol{T}_1^\theta$ and $\boldsymbol{T}_2^\theta$) together with specific projective transformations are equivalent to a rigid body rotation of points $\boldsymbol{x} \in \mathbb{R}^3$ (with the angle θ around the y-axis) followed by the projective transformation $\boldsymbol{P}_z\boldsymbol{R}_z^\alpha$:

$$\tilde{\boldsymbol{x}} = \boldsymbol{P}_z\boldsymbol{R}_z^\alpha\boldsymbol{R}_y^\theta\boldsymbol{x} \tag{4.3}$$

with

$$\boldsymbol{R}_z^\alpha\boldsymbol{R}_y^\theta = \begin{bmatrix} \cos\theta\cos\alpha & -\sin\alpha & \sin\theta\cos\alpha \\ \cos\theta\sin\alpha & \cos\alpha & \sin\theta\sin\alpha \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

leads to

$$\begin{aligned} \tilde{x} &= x\cos\theta\cos\alpha - y\sin\alpha + z\sin\theta\cos\alpha \\ \tilde{y} &= x\cos\theta\sin\alpha + y\cos\alpha + z\sin\theta\sin\alpha \\ \tilde{z} &= 0 \end{aligned}$$

where $\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \boldsymbol{x}$ and $\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \tilde{\boldsymbol{x}}$

Let us first consider the $\boldsymbol{T}_1^\theta$, then the $\boldsymbol{T}_2^\theta$ interpolation.

Theorem 1. *See Section 4.2.2.*

Proof. Let

$$\tilde{\boldsymbol{x}} = \boldsymbol{P}_z\boldsymbol{R}_z^\alpha\boldsymbol{R}_x^\beta\boldsymbol{T}_1^\theta\boldsymbol{d}$$

In the following equations, let $\grave{\sigma} = \cos\sigma$ and $\acute{\sigma} = \sin\sigma$.

$$\boldsymbol{R}_z^\alpha\boldsymbol{R}_x^\beta\boldsymbol{T}_1^\theta = \begin{bmatrix} \grave{\theta}\grave{\alpha} & -\grave{\beta}\acute{\alpha} & \acute{\beta}\acute{\alpha} & \acute{\theta}\grave{\alpha} & 0 & 0 & \dots \\ \grave{\theta}\acute{\alpha} & \grave{\beta}\grave{\alpha} & -\acute{\beta}\grave{\alpha} & \acute{\theta}\acute{\alpha} & 0 & 0 & \dots \\ 0 & \acute{\beta} & \grave{\beta} & 0 & 0 & 0 & \dots \end{bmatrix}$$

leads to

$$\begin{aligned} \tilde{x} &= d_1\cos\theta\cos\alpha - d_2\cos\beta\sin\alpha + d_3\sin\beta\sin\alpha + d_4\sin\theta\cos\alpha \\ \tilde{y} &= d_1\cos\theta\sin\alpha + d_2\cos\beta\cos\alpha - d_3\sin\beta\cos\alpha + d_4\sin\theta\sin\alpha \\ \tilde{z} &= 0 \end{aligned}$$

We obtain $x = d_1$, $y = d_2\cos\beta - d_3\sin\beta$, and $z = d_4$. □

Theorem 2. *See Section 4.2.2.*

Proof. Let

$$\tilde{\boldsymbol{x}} = \boldsymbol{P}_z \boldsymbol{R}_z^{\alpha} \boldsymbol{R}_y^{\beta} \boldsymbol{T}_2^{\theta} \boldsymbol{d}$$

Then

$$\boldsymbol{R}_z^{\alpha} \boldsymbol{R}_y^{\beta} \boldsymbol{T}_2^{\theta} = \begin{bmatrix} \grave{\theta}\grave{\beta}\grave{\alpha} & -\acute{\alpha} & \grave{\theta}\acute{\beta}\grave{\alpha} & \acute{\theta}\grave{\beta}\grave{\alpha} & \acute{\theta}\acute{\beta}\grave{\alpha} & 0 & \dots \\ \grave{\theta}\grave{\beta}\acute{\alpha} & \grave{\alpha} & \grave{\theta}\acute{\beta}\acute{\alpha} & \acute{\theta}\grave{\beta}\acute{\alpha} & \acute{\theta}\acute{\beta}\acute{\alpha} & 0 & \dots \\ -\grave{\theta}\acute{\beta} & 0 & \grave{\theta}\grave{\beta} & -\acute{\theta}\acute{\beta} & \acute{\theta}\grave{\beta} & 0 & \dots \end{bmatrix}$$

leads to

$$\begin{aligned} \tilde{x} &= d_1 \cos\theta \cos\beta \cos\alpha - d_2 \sin\alpha + d_3 \cos\theta \sin\beta \cos\alpha \\ &\quad + d_4 \sin\theta \cos\beta \cos\alpha + d_5 \sin\theta \sin\beta \cos\alpha \\ \tilde{y} &= d_1 \cos\theta \cos\beta \sin\alpha + d_2 \cos\alpha + d_3 \cos\theta \sin\beta \sin\alpha \\ &\quad + d_4 \sin\theta \cos\beta \sin\alpha + d_5 \sin\theta \sin\beta \sin\alpha \\ \tilde{z} &= 0 \end{aligned}$$

We obtain $x = d_1 \cos\beta + d_3 \sin\beta$, $y = d_2$, and $z = d_4 \cos\beta + d_5 \sin\beta$. □

4.3 User Study

The following user study evaluates the effectiveness of the introduced 3D rigid body rotation approach and compares it to direct transition between scatter plots. The more problematic of the two situations is considered: two axes are changed. The research question was the following: should one first rotate the camera to a subspace where the consecutive transition will look like a rigid body rotation or should one perform the transition directly from the actual viewpoint (without rigid body behavior)? The hypothesis was that 3D rigid body rotation would allow users to identify correspondence between elements in the scatter plots with less error than when direct transition was performed.

4.3.1 Task and Stimuli

The participants had to identify corresponding clusters from a multidimensional data set shown in scatter plots before and after the transition. The synthetic data sets were generated from the same random distributions and contained 10 clusters. The number of clusters was chosen to be larger than the typical number of objects that can be tracked independently (see [CA05]). Each of the clusters consisted of 50 points from a normal distribution. The cluster centroids were uniformly distributed in the domain. To add a

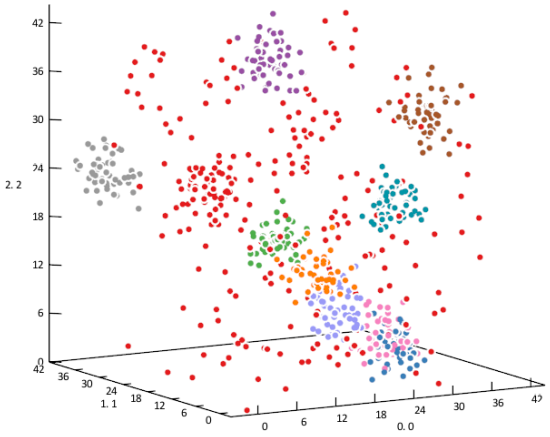


Figure 4.1: Scatter plot showing synthetic data used during the user study, presented from the initial viewport.

degree of distraction 100 noise points uniformly distributed over the domain were also added. Figure 4.1 shows an example, with color coding of the clusters.

For the direct transition case, the initial scatter plot was shown for 4 seconds. Then, the points were rendered gray and a transition was performed that changed the axes 0 and 1; this transition took 1.83 seconds (110 frames at 60 fps). After the transition, a single cluster was highlighted and the user had to select the corresponding cluster from a second scatter plot drawn next to the animated one showing the initial plot. The selection process had no time constraints. The result of the selection was recorded for the subsequent analysis of the study.

The rigid body rotation task was similar. The only difference was that a rotation to align the axes 0 and 1 with the y-axis of the screen by performing a 10 degree rotation within 0.33 seconds (20 frames) was applied prior to the transition, which was shortened to 1.5 seconds (90 frames). By this alignment, the transition looked like a 90 degree rigid body rotation and the overall duration was identical to the direct transition task.

4.3.2 Participants

The study was performed with 12 participants (7 male, 5 female). The participants were students of our university. Six were computer scientists, six were engineers. The average age was 26 (minimum 21, maximum 32). They were paid €10 for participating in the study. Six stated that they were familiar with visualization techniques or had attended a lecture with this topic. All participants had normal or corrected-to-normal vision without color vision deficiency, which has been confirmed by an Ishihara test and a Snellen chart to estimate visual acuity.

4.3.3 Study Procedure

The study was conducted in a laboratory that was insulated from outside distractions. A 22 inch BenQ G2200WT TFT screen at a display resolution of 1680×1050 pixels with 24 bit color depth driven by a Windows PC was used. Subjects were introduced to the methods and had time to gain experience and develop strategies to solve the tasks. Feedback was given in terms of right or wrong choice they made. This training phase took about 15 minutes. After the training phase, they had to perform 100 tasks with both methods without feedback.

The same set of 100 different pseudo-random synthetic data sets was generated for the two methods. The same starting camera position was used for rotation and direct transition, and the ending position differed only by 10 degrees. This repeated-measures design had two within-subjects variables: *Method* (i.e., direct transition vs. rigid body rotation) and *DataSet* (i.e., the different data sets). Since the differences in difficulty of the data sets are not relevant, the *Method* variable was randomized and balanced between subjects but not the *DataSet* variable, minimizing the order effects on the paired samples.

Switching randomly between the methods, after each task, would have been quite demanding for the participants; therefore, the tasks of the same method have been grouped. The participants were randomly assigned, to start with the rotation or the direct transition tasks. To further minimize learning effects the 100 tasks were split in two parts and the AB/BA sequence was replaced by an ABAB/BABA sequence (where A and B correspond to direct transition and rigid body rotation, respectively). The test procedure was similar to a crossover design with two periods and two treatment sequences. One round of A or B took about 10 minutes, leading to an overall average study duration just below one hour per participant.

4.3.4 Study Results

From the complete set of 12×100 tasks, there were 198 cases where direct transition performed better than rigid body rotation and 251 cases where rotation was better; in 308 cases, both were correct and in 443 cases, both were wrong. An often used descriptor for success in Bernoulli experiments are the odds $\text{odds}(E_{\text{direct}}) = (198 + 308)/(1200 - 198 - 308) = 0.729$; $\text{odds}(E_{\text{rot}}) = (251 + 308)/(1200 - 251 - 308) = 0.872$; and the odds ratio comparing the odds of succeeding the tracking of the cluster for using the rotation method with the odds of using the direct transition method is $R_{\text{rot,direct}} = \text{odds}(E_{\text{rot}})/\text{odds}(E_{\text{direct}}) = 1.196$. That means the odds of succeeding with the tracking are 19.6% higher for the rotation method than for the direct transition method.

The experimental variables *Method* and *DataSet*, their interaction *Method* $\times$ *DataSet*, and the main effect of subjects was modeled. A simple ANOVA cannot be applied in this case, since the error component cannot be assumed to be normally distributed. Instead

logistic regression can be used, which is a generalized linear model [NW72], to account for the binomial distributed outcomes. The likelihood ratio test showed that *Method* was significant ($\chi^2(1) = 5.80, p = 0.016$). There was no significant interaction between the *Method* and *DataSet* variables.

The learning or fatigue effects were also analyzed. The variance captured by the *DataSet* consists of two parts: one is the "difficulty" difference between the tasks, the other is the learning effect. Since the tasks were generated randomly, task difficulty should be independent from task ordering. To analyze the effect of learning or fatigue, the task ordering can be used as additional effect in the regression to separate the learning effect from the (randomized) difficulty. Therefore, a linear dependency on the *TaskOrder* variable (an integer variable that describes the ordered index of the task for a participant) can be modeled, which is an additive effect on the log of the odds in the logistic regression model. The estimated parameter for the *TaskOrder* was $\beta_{\text{TaskOrder}} = 0.0028$ (log of the odds between consecutive tasks), which is a positive learning effect and corresponds to an increase of correctly identified correspondences of 6.9 percent points during the 100 tasks. The likelihood ratio test on the *TaskOrder* was significant ($\chi^2(1) = 3.91, p = 0.048$).

4.3.5 Discussion

The main result of the user study is that rigid body rotation is more effective than direct transition by unconstrained interpolation between scatter plots.

Another result is that there are learning effects for both types of transition schemes. The learning effect may be caused by the fact that the participants were no expert users of 3D scatter plots. Therefore, it can be expected that the overall performance of 3D scatter plot navigation will further improve with increasing experience.

4.4 Scatter Plot Matrix Navigation

4.4.1 Technique

The technique presented in this section fills the white spots in Table 4.1 labeled with "HERE" and can be seen as the 3D extension to the 2D scatter plot matrix navigation technique by Elmqvist et al. [EDF08]. The novel interpolation scheme and projection technique can be used to navigate between arbitrary 3D scatter plots, as long as one axis mapping is preserved, and therefore are ideally suited as key components in the novel 3D scatter plot matrix navigation. Unlike in the approach of Elmqvist et al., the transitions would not automatically look like 3D rotations and special viewpoints need to be set, limiting the two rotational degrees of freedom (no rotation in the image plane) to one.

Given an n-dimensional data set, there are n^2 2D scatter plots that can be arranged in a scatter plot matrix. For the extension to 3D, there are n^3 3D scatter plots that can be

arranged as $n \times n \times n$ 3D scatter plots within a cube. However, such a presentation would suffer from occlusion; and selection of a 3D position (picking) would be cumbersome with a 2D input device. With the introduced interpolation scheme navigation is limited to changes of one or two axes. Therefore, full 3D navigation is not necessary. By keeping one specific axis constant one can reach $n \times n$ 3D scatter plots. Hence, there are $3 \times n \times n$ 3D scatter plots that are reachable (by changing at most two axes mappings) from a given scatter plot (not all of these are different). Only the scatter plots that preserve at least one dimension needs to be displayed. Scatter plots that preserve the dimension mapped to the x-axis are projected to the back face of the cube in x-direction. The y- and z-directions are handled accordingly. In total, three 2D scatter plot matrices are arranged on the back side of a cube. This representation is termed *3D scatter plot matrix*. Like a 2D scatter plot matrix can be used to navigate between 2D scatter plots, a 3D scatter plot matrix can be used to navigate between 3D scatter plots.

Figure 4.2 shows an example of a 3D scatter plot matrix. Here, data dimension 3 is mapped to the y-axis and all 2D projections of the 3D scatter plot matrices that preserve the y-axis mapping are projected to the back face perpendicular to the y-axis of the cube; cf. the upper right part of the figure. This mapping from 3D to 2D is reminiscent of the ExoVis method [Tor03]. This approach allows us to represent all reachable 3D scatter plots by their 2D projections without overlap and occlusion. Therefore, elements can be readily selected by a 2D input device, i.e., just by clicking on the desired destination with the mouse. The selected element in the 3D scatterplot matrix is used as target 3D scatterplot for the transition. The interpolation scheme, presented in Section 4.2, is used to perform the transition in the 3D scatterplot view. The projection technique can be used to first rotate the view into a subspace where the transition looks like a rigid body rotation. In the current implementation, this is done by a user trigger but can also be done automatically prior to the transition. The rotation of the 3D scatter plot matrix is linked to the 3D scatter plot view. Therefore, the 2D scatter plot projections look like the main 3D scatter plot, when looking along one of the major axes. The 3D scatter plot can be rotated by dragging with the mouse or by shortcuts that bring the scatter plot axes in alignment with the screen axes and provide views that are identical to 2D scatter plot views with the additional depth sorting.

3D scatter plots allow us to map three data dimensions to the axes of the plot. It makes no sense to map identical dimensions to different axes. Likewise, the utility to move an already mapped dimension to a different axis is limited. It may be easy to see such configurations in 2D scatter plot matrix navigation; it is easy to miss them in 3D navigation. To help the user find the remaining, appropriate axis mappings, the navigation is restricted to these mappings which are also marked gray in the 3D scatter plot matrix (Figure 4.2).

Dense 2D scatter plots suffer from overplotting, see the upper image in Figure 4.3. This issue can be alleviated to some extent when using 3D scatter plots, to be more

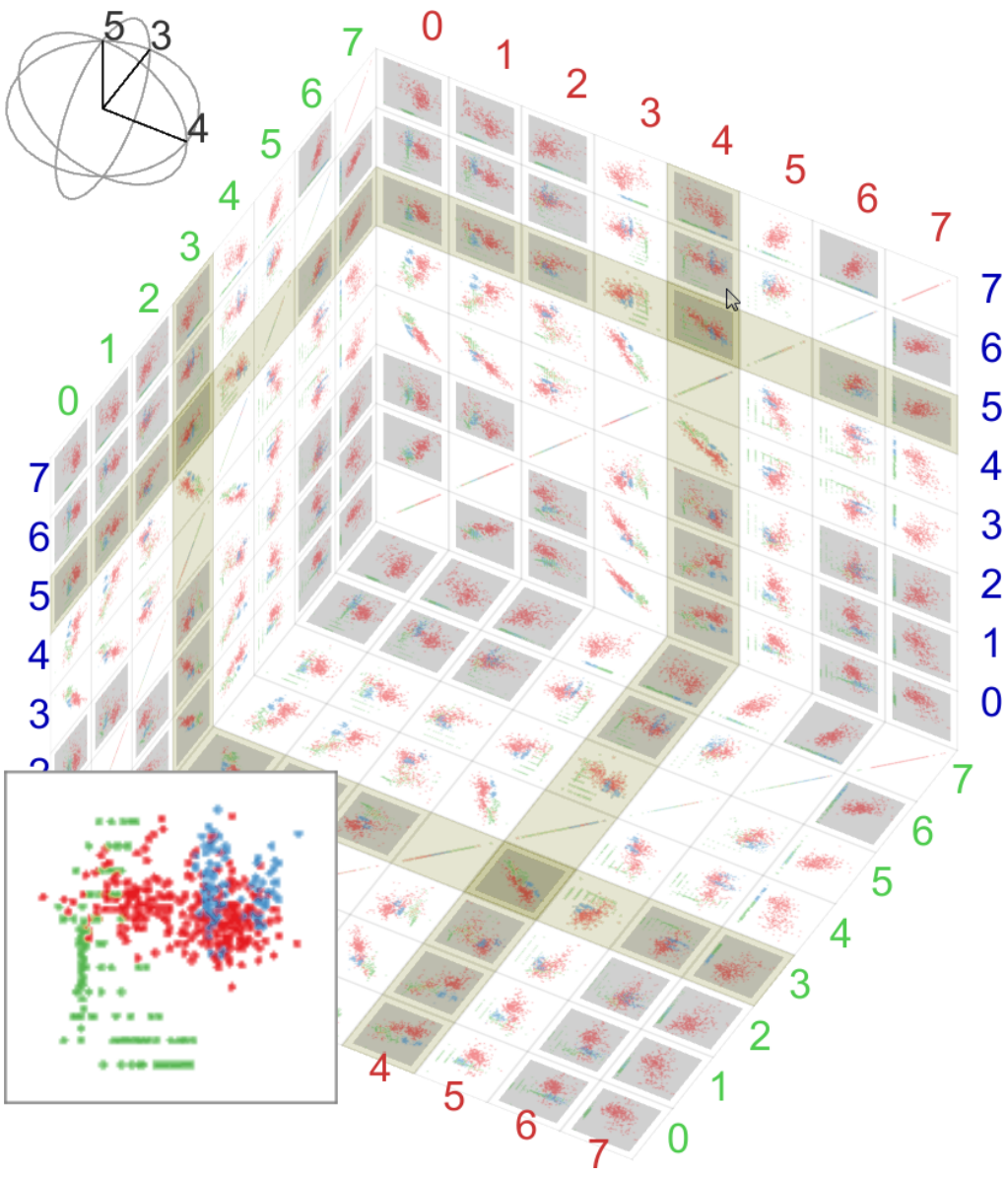


Figure 4.2: Three-dimensional scatter plot matrix showing the 8-dimensional "olive oil" data set. The *x*-, *y*-, and *z*-axes are annotated with red, green, and blue colored numbers, respectively. The currently selected 3D scatter plot is marked with yellowish strips and maps the data dimensions (4, 3, 5) to the (*x*, *y*, *z*) axes. Plots to which one can navigate are marked gray. In the lower-left corner, a widget shows the magnification of the plot under the current mouse position. A biplot [Gab71] is included in the upper-left corner.

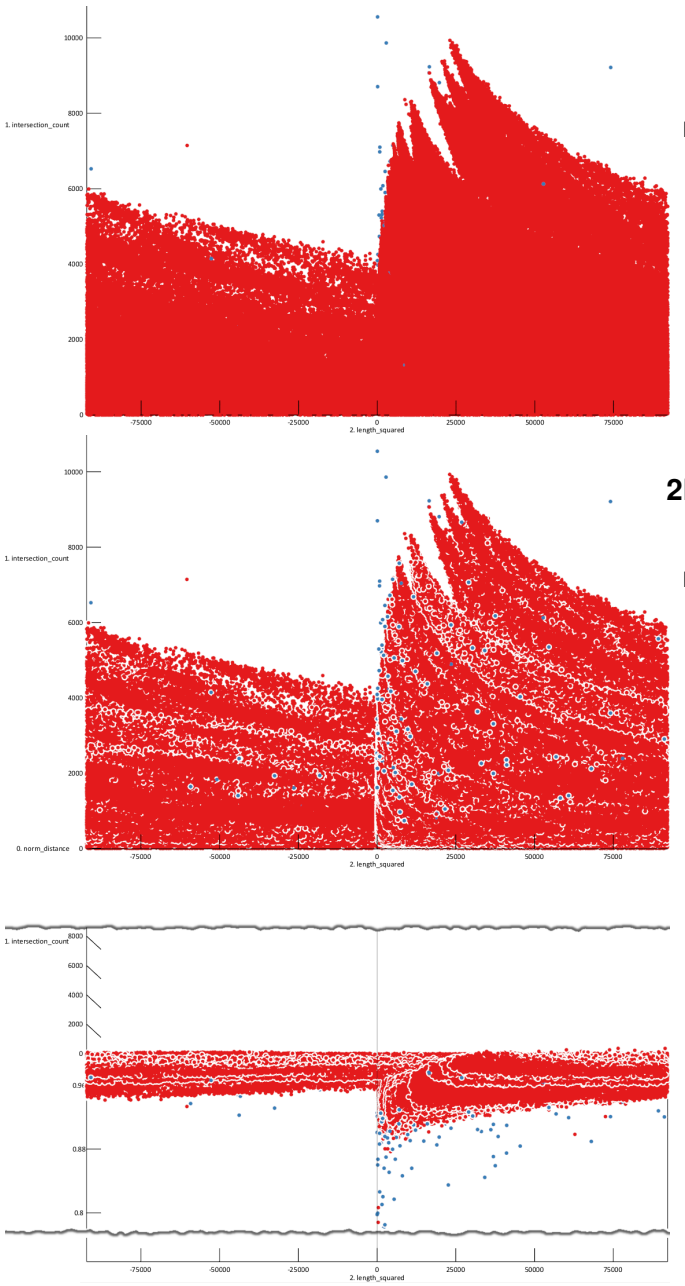


Figure 4.3: Comparison of 2 and 3 axes mappings and one rotational degree of freedom. In the upper two images, the horizontal and vertical image axes show the same data dimensions, *length_squared* and *intersection_count*, respectively. There is no depth axis used in the upper image. In the center image, data dimension *norm_distance* is assigned to the depth axis; halos emphasize the variation with respect to *norm_distance* in the data. The lower image uses the same axis mapping as the center image and shows the data set from above.

precise, 2D+sorting according to my taxonomy. As mentioned in Section 2.5, depth cues are important when presenting 3D data on a 2D screen. By using the depth component for sorting and rendering halos around the points, using the halo rendering technique introduced in Section 3.5, more information is available from the 2D image by facilitating shape perception, see the center image in Figure 4.3 and the accompanying video (`scatMat3d.avi`) for an animated example.

The 3D shape would be even more highlighted using additional depth cues. Especially the application of the illumination technique presented in the last chapter could be valuable. Since the user is free to map any dimensions to the axes of the scatter plot and to navigate between 3D scatter plots a pre-calculation is not possible. A possible solution would be to calculate the illumination on the fly for example by using the distributed processing framework introduced in Chapter 7.

4.4.2 Applications

Here two application examples for the introduced scatter plot matrix navigation technique are presented.

Illustrative Example of 3D Scatter Plot Matrix Navigation

The "olive oil" data set [FALT83] is used to demonstrate 3D scatterplot matrix navigation because this data set is well known and has a medium number of data items and dimensions. The data set consists of 572 olive oils coming from 3 regions of Italy. The composition of each oil with respect to 8 fatty acids (*palmitic*, *palmitoleic*, *stearic*, *oleic*, *linoleic*, *linolenic*, *arachidic*, and *eicosenoic*) is given as data attributes.

Figure 4.4 illustrates how scatter plot matrix navigation can be used to explore multivariate data sets by showing several steps of a navigation example. Given the goal of separating the three clusters, the task can be decomposed to first separate one of the clusters from the other two using one dimension of the scatterplot and then the other two clusters from each other using two other data dimensions.

The first subfigure starts with a 3D scatter plot in which the dimensions *palmitic* (0), *palmitoleic* (1), and *stearic* (2) are mapped to the x-, y-, and z-axes, respectively. The three clusters are not separated in the scatter plot. In the 3D scatter plot matrix, one can see that the dimension *eicosenoic* (7) separates the red cluster from the other two. Therefore, it is chosen to be mapped to the x-axis (subfigure 2). The scatter plot is rotated to the subspace where the transition that replaces dimension 0 by dimension 7 will appear as a rigid body rotation (subfigure 3). The transition is shown in subfigures 4–7. By now the red cluster is clearly separated and mapped to the x-axis of the scatter plot, and one can try to separate the green and blue clusters using the other two axes. Now the 3D scatter plot is rotated to show the y- and z-axes. From the 3D scatter plot matrix, one can see that *arachidic* (6) and *linoleic* (4) separate the two clusters (subfigure 8). These

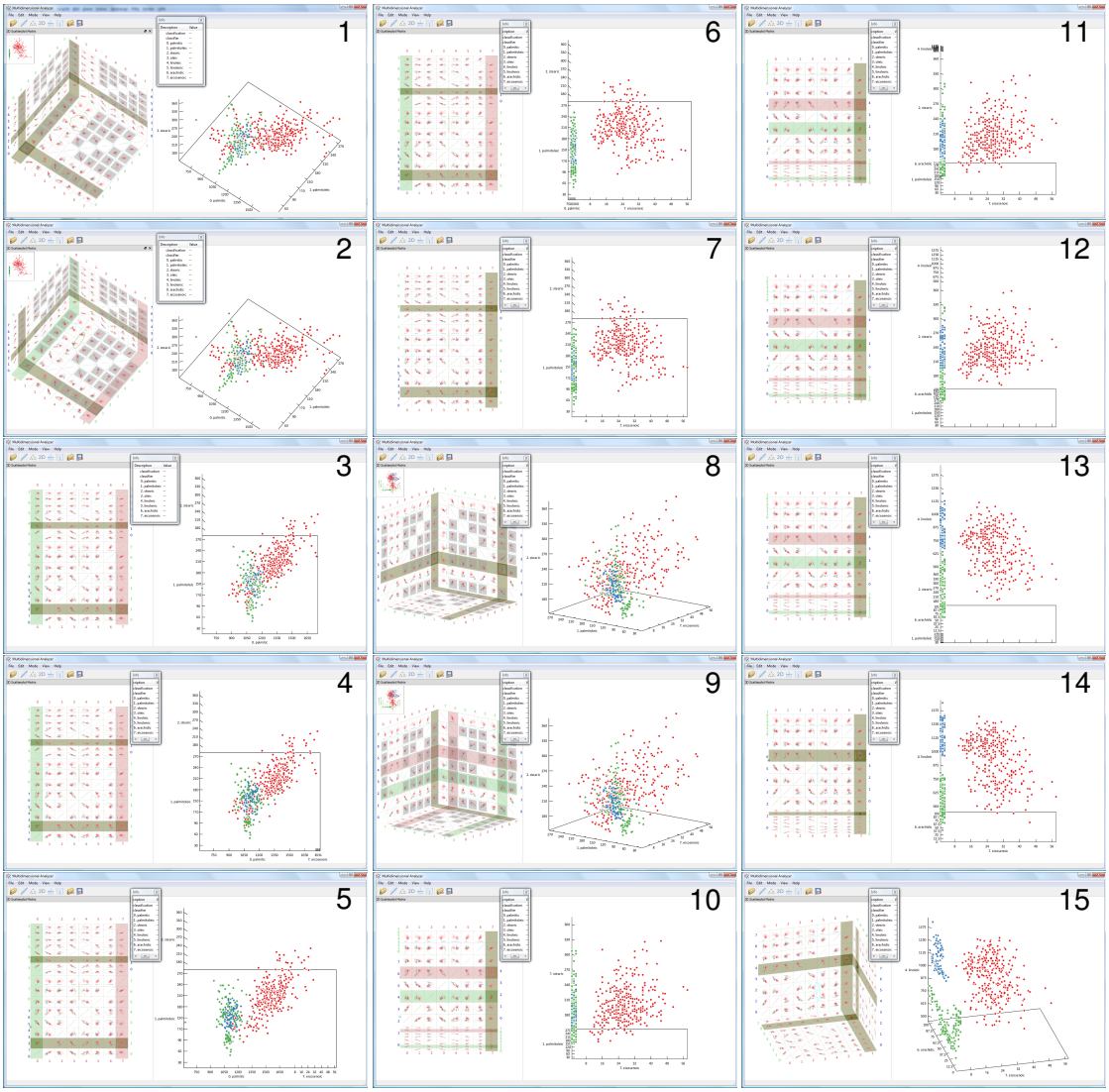


Figure 4.4: Scatter plot matrix navigation showing how the three clusters of the “olive oil” data set can be separated.

two are chosen to be mapped to the y- and z-axes, respectively (subfigure 9). The scatter plot is rotated to the subspace where the transition that replaces dimensions 1 and 2 by dimensions 6 and 4 will appear as a rigid body rotation (subfigure 10). The transition is shown in subfigures 11–14. At the end of the transition, the clusters are also separated in the projection. Subfigures 14–15 give an impression of the 3D locations of the points. The dimensions *eicosenoic*, *linoleic*, and *arachidic* have been identified to separate the three regions. This is not a new finding [CCH04]. However, this example shows how the 3D scatter plot matrix navigation can be used to systematically navigate towards a 3D scatter plot where the clusters are separated. The 2D scatterplots presented in the 3D scatterplot matrix help identify interesting 3D scatterplots that can be used as targets for next transitions. Separating clusters is just one possible, yet typical application. In general, the technique allows the user to obtain a preview of a target scatterplot, which helps the user in exploring a dataset.

Feature Extraction for Natural Language Processing

This application scenario covers exploration of multidimensional data with many data points, so that a dense coverage of scatter plots can be expected. More specifically, a domain expert from the field of natural language processing used the system and provided feedback. In the case studied, the expert user wanted to apply some novel feature extraction methods and develop new features for the PAN plagiarism corpus (PAN-PC-09) (`http://www.webis.de/research/corpora`). The derived data set is 11-dimensional and contains over a million items, most of which are classified negative and a few of which are classified positive.

The expert was familiar with the multidimensional analysis tool GGobi, but could not open such a large set with the application. Therefore, he was interested in using the novel application for analysis. Figure 4.3 shows dense scatter plots of the data set.

By looking at the scatter plot matrix, it became obvious that the eight features *topLSV1–topLSV8* are nearly identical. First just 2D scatter plot matrix navigation was offered to the expert, by disabling the third axis and fixing the rotation of the viewport, which was similar to the technique by Elmqvist et al. [EDF08]. The scatter plot navigation was found useful because the domain expert saw in advance where he would arrive after performing a movement—in contrary to GGobi, where navigation is more cumbersome because there is no preview. After the 2D session, I asked if he had missed a third dimension, but he answered that 2D was right for him. We then continued with the study and enabled the third dimension. The expert found the 3D scatter plot more informative than the 2D plots because he got an impression of the density of the points due to the halo technique introduced in Section 3.5. He found out that the red points have a high density and have a quite sharp border. He also found the 3D scatter plot quite useful because the additional data dimension and the easy navigation through rotation.

The data dimension *length_squared* was mapped to the x-axis of the plots shown in Figure 4.3. Obviously, this quantity should not be negative; furthermore, one can see from the point distribution—especially when looking at the 3D scatter plot augmented with halos at depth discontinuities—that the plot on the right side at the maximum *length_squared* values matches the distribution on the left side at the maximal negative *length_squared* values. This may lead to the hypothesis that the dimension *length_squared* was stored as an unsigned number that was interpreted as a signed one during processing. However, when looking at the bottom of Figure 4.3, it is observable that the pattern does not stop at *length_squared* $= 0$ and the final conclusion can be drawn that there must have been an arithmetic overflow during processing. This example demonstrates the additional information that can be obtained from 3D scatter plots compared to 2D scatter plots, even on a 2D output device, when using depth cues which are able to give an impression of 3D shape. According to my taxonomy presented in Table 4.1, it is obvious that none of the techniques being in the 2-axes column are capable of revealing this 3D shape since they miss a depth axis.

4.5 Multidimensional Analyzer

The algorithms and rendering techniques introduced in this chapter and the halo rendering technique introduced in Section 3.5 form the basis for *Multidimensional Analyzer* (MDA) a framework for multivariate data analysis and visualization which is further expanded in the other chapters of this thesis. MDA is implemented in C++ using Qt and OpenGL. All figures presented in this chapter show screenshots of MDA.

Multivariate data stored as CSV-files can be loaded into MDA. Besides the scatter plot matrix navigation, MDA can project the data based on PCA or based on a projection computed by the FastMap algorithm. By default MDA simply scales the input data to a unit hypercube; however, it also supports histogram equalization. The GUI allows easy adjustment of the point size. For the halo rendering the halo size and the depth offset can also be adjusted. Additionally MDA supports the storage and loading of the current visualization parameters, which helps reproducing results.

To enable flexible, efficient, and high-quality rendering of large amounts of data, MDA sores all points in their high-dimensional representation as shorts in a vertex buffer object (VBO) along with another attribute, which is used for color coding. A vertex shader projects the high-dimensional data to 3-space, defined by three high-dimensional vectors.

The frame rate of MDA is dependent on the size of the scatter plot. The test platform was a PC with Core 2 Quad Q6600 CPU, 4 GB DDR2 main memory, and GeForce 8800GTX GPU. A small scatter plot consisting of 572 points (olive oil data set) runs at 112 fps, a large plot with 1 122 741 points (NLP example) with 14 fps in full-screen on a 1920×1200 resolution screen with 16x multisample anti-aliasing (MSAA).

CHAPTER 5 VISUALIZATION WITH DECISION TREES

This chapter also deals with aspects of visual analysis of multivariate data using scatter plots. However, it introduces two new techniques providing a way to view the data together with a decision tree classifier.

In *supervised learning*, the purpose of a *classifier* is to predict a *target attribute* based on given attributes of an *item*. The given attributes are also called *features* and form the *feature vector*. The classifier is trained by some examples for which the feature vector and the target attribute is given. These examples are called the *training set*. There exist different classifiers; here, the focus is on decision tree classifiers and a brief review of previous work related to other examples, in particular, support vector machines (SVMs) and linear discriminant analysis is provided. Decision trees use a two stage process: first the tree is learned from the training set and later the tree is used to classify items. Decision tree classifiers are often used for data mining and analysis because they use a white box model. This makes it easy to understand how individual data items are classified. Here, the focus is on univariate, binary decision trees because they are most commonly used. A data item is classified by a decision tree by traversing the tree from the root node. At each inner node, one decides to descend to the left or to the right child by comparing the value of single feature to a threshold value defined at the inner nodes of the tree. At the leaf nodes, the classification of the item is obtained.

Manning and Schütze [MS99] present a good introduction to machine learning approaches in general. Safavian and Landgrebe [SL91] provide a mathematically funded overview of decision tree classifiers. The survey by Rokach and Maimon [RM05] is easy to follow and explains the concepts of decision tree classifiers, how decision trees are built, how they are used to classify data items. There are application domains where many features are available [GE03] to describe the data items. Sometimes, these features are correlated to each other or not related to the target attribute, or simply too numerous to be efficiently used by a decision tree learning algorithm. In such cases, the features should be restricted to a smaller set; this process is called *feature selection* [GE03]. Hand-crafted features provide a way to bring in domain knowledge to the classifier.

Therefore, Guyon and Elisseeff [GE03] recommend constructing a better set of features if domain knowledge is available, even if enough generic features are given. This procedure of finding features that describe data aspects is called *feature engineering*. Scott and Matwin [SM99] describe feature engineering for text classification, where there is already a vast amount of features given by the "bag of words" representation.

The first new technique introduced in this chapter uses the decision tree as a navigation structure to help understand a given decision tree but also allow for exploration of a multivariate data set itself. The second one displays multivariate data items with scatter plots together with the classifier boundary defined by the decision tree with the goal to assist a certain feature engineering task. Both techniques benefit from domain knowledge and are combined with other visualization techniques which link this domain knowledge to the scatter plot representation.

5.1 Scatter Plot Navigation with Decision Trees

A new navigation technique for interactive analysis of multidimensional data spaces is introduced. The technique combines decision trees with scatter plots in coordinated views. Instead of looking at individual data items when classifying an item like presented in the introduction to this chapter one looks at all data items at once, by displaying them in a scatter plot. While the user is moving between the nodes of the tree, an animated scatter plot is shown that displays the relevant data dimensions associated with these nodes, with smooth transitions between nodes. While descending the decision tree, data items not belonging to the selected subtree are blended out. One additional issue needs to be solved. 2D scatter plots can display two data dimensions at a time, and even with 3D scatter plots only three data dimensions can be displayed, but decision trees can deal with multivariate data having dozens of dimensions. Fortunately, decision trees—unlike support vector machines—use only one data dimension at individual inner nodes. The idea of the presented approach is to exchange the scatter plot dimensions during the traversal of the decision tree such that at each inner node of the tree the respective dimension is mapped to one of the scatter plot axes. To maintain context rotational transitions are employed between scatter plots, using the scatter plot navigation technique presented in the last chapter.

The approach supports two slightly different directions of analysis. First, the navigation method allows the user to understand the decision tree and, thus, the classification method at hand. Such analysis could also help in feature engineering tasks, see Section 5.2. Second, it serves as a method to obtain insight in the multidimensional data set. Exploration is facilitated by guiding the user through the enormous number of combinations of data dimensions that could potentially be mapped to the two data axes of a scatter plot.

Finally the usefulness of the approach for two typical application examples is shown.

The first example is from trajectory classification in computer vision, where the technique can be used to navigate in high-dimensional trajectory data sets, guided by decision trees. As second example, it is illustrated how the technique can be used to gain insight into a common benchmark data set in multidimensional data analysis.

5.1.1 Background

Classification techniques are typically employed for multidimensional data and, therefore, visual analysis can be applied to the combination of classification and such data. For example, Caragea et al. [CCH01] use scatter plots to gain insights into SVM classifiers. They present several 2D scatter plots showing the data items with highlighted support vectors and histograms of predicted values from the SVM classification. Poulet [Pou04] linked scatter plot matrices and parallel coordinates to histogram SVM classifier visualization. Hamel [Ham06] shows how a decision surface of an SVM can be visualized as a curve in a 2D self-organizing map (SOM).

Visualization is not restricted to SVMs, as for example, shown for linear discriminant analysis [CLKP10]. Visualization of the classifier boundary is also described in a recent work by Migut and Worring [MW10]. They display a separation line that approximates the classifier boundary for the given data elements.

Teoh and Ma [TM03] present a technique to construct and analyze decision trees using star coordinates and parallel coordinates. Barlow and Neville [BN01] present various visualizations showing the number of data items and the classification proportion which are mapped to line width and node color of the decision tree and bar and line charts for the evaluation of the classification results. Ankerst et al. [AEK00] present a visualization technique that supports users in generating decision tree classifiers manually based on bar visualizations showing the distribution of classes for each data dimension. The technique is more labor-intensive than automatic classifier training, but offers the possibility to gain a better understanding of the decision tree and often produces smaller trees with similar accuracy as automatically trained ones, at least for data sets having only numerical attributes. Poulet and Do [PD08] extend their approach to generate bivariate decision trees—which use two dimensions of the data set for internal nodes and are less common—based on defining separation planes in a scatter plot selected from a scatter plot matrix. The freely available data mining software Weka (`http://www.cs.waikato.ac.nz/~ml/weka/`) offers a very similar way to build bivariate decision tree classifiers interactively. The last two approaches target the interactive building of bivariate decision trees, whereas the focus here is on the exploration of multidimensional data given a univariate decision tree. Moreover, the approach presented here differs in the usage of the scatter plot: for the interactive classification, the user chooses the dimension that should be mapped to the axes manually, whereas here the nodes of the decision tree are used to select relevant dimensions. Finally, here smooth animation during changes of axis mapping is performed instead of switching to another

scatter plot. Sectioned scatter plots [Urb08] introduced by Urbanek show the partition boundaries defined by a decision tree in a 2D scatter plot. Urbanek also proposes to map variables which are adjacent in the tree to the axes of the scatter plot, which is ensured by our mapping, but recommends using only the nodes which are close to the root of the tree. Furthermore, Urbanek does not define which dimensions of the data should be mapped to the axes at a given node of the decision tree, which is required to navigate between nodes. Similarly to the above mentioned approaches, Urbanek targets to support the building of decision trees by examining cut points not the exploration of multivariate data. BaobabView [vdEvW11], recently presented by van den Elzen and van Wijk, shows the distribution of the classes as a flow map [Min69, PXYH05] and provides Stremgraphs [BW08b] at the inner nodes which show the distribution according to the attribute chosen. It also focuses on decision tree building and similarly to Ankerst et al., it provides univariate statistical views to support this task. I am not aware of any other previous work in visualization that would make specific use of decision trees for navigation in plots of multidimensional data.

5.1.2 Decision Tree Navigation

This section now describes the new navigation technique based on decision trees.

The idea of decision tree navigation is that the tree can be interactively visited by the user (moving between the nodes along the edges of the tree) and, fully automatically, a linked view of the corresponding scatter plot is shown with the appropriate data axes.

When the user is at a certain node, the dimension of that node should be mapped to an axis of the scatter plot. The technique could map the dimension of the node the user chooses to descend (left vs. right child) to the other axis, but this would require the application to change the mapping each time the user modifies the direction of descend. Instead, the technique maps the parent node to the other axis, see Figure 5.1 for an illustration. Please note that this kind of mapping is not restricted to binary decision trees, but can be applied to any tree structure.

A smooth transition between scatter plots during tree traversal is required to show correspondence between the plots. Analogously to the previous chapter, the animated data points follow the motion of a corresponding rigid body in virtual 3D space.

To navigate with the decision tree, the user needs to select the direction he wants to proceed (upward or downward) and when descending the direction to descend (left or right). Additionally, zooming and panning are possible inside the scatter plot.

During tree traversal, the method grays out and scales down data items that are not part of the current sub-tree. This way, the user can focus on the relevant subset of the data but still retain context information.

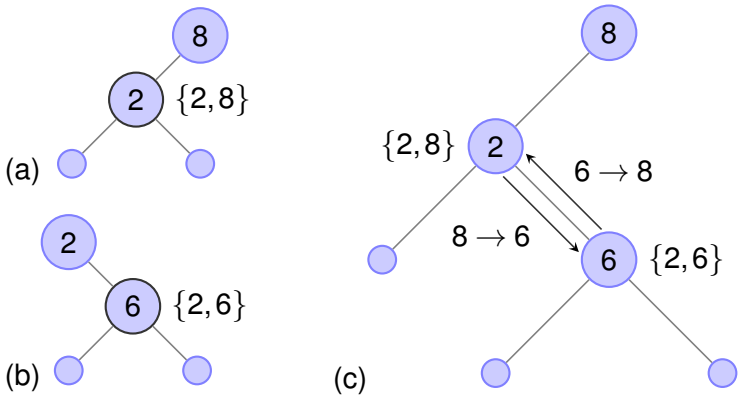


Figure 5.1: The figures show several parts of decision trees. The dimensions of the nodes are denoted by numbers. The dimensions that are mapped to axes when one is at a given node of a decision tree are written next to the nodes in brackets. As shown in subfigures (a) and (b), only the node's dimension and the dimension of the node's parent are relevant. Subfigure (c) shows the dimensions that need to be changed during transition of the nodes; just one axis mapping is affected.

Using 3D Scatter Plots

The technique can also be applied using 3D scatter plots. With 3D plots an additional axis is available, and a different axis mapping strategy can be applied. Instead of mapping the current node and its parent to the axes, the current node and its children can be mapped to them. This way there is a kind of look ahead available. The 3D scatter plot navigation technique is especially well suited to interpolate between the mappings, since at most two axis mapping change at a time.

To focus on the new technique and simplify the illustration on print media, here only the combination with 2D scatter plots is presented.

5.1.3 Application and Evaluation

Scatter plot navigation with decision trees has been implemented within MDA, see Section 4.5. MDA supports loading a decision tree from file, the core elements of decision tree navigation (i.e., the decision tree itself and the link scatter plot views) and, additionally, further coordinated data views, which may be application-dependent. Additionally the scatter plot indicates which points belong to the training set by a small white dot and displays false classifications by a large triangle. Now the approach is illustrated for two typical application scenarios.

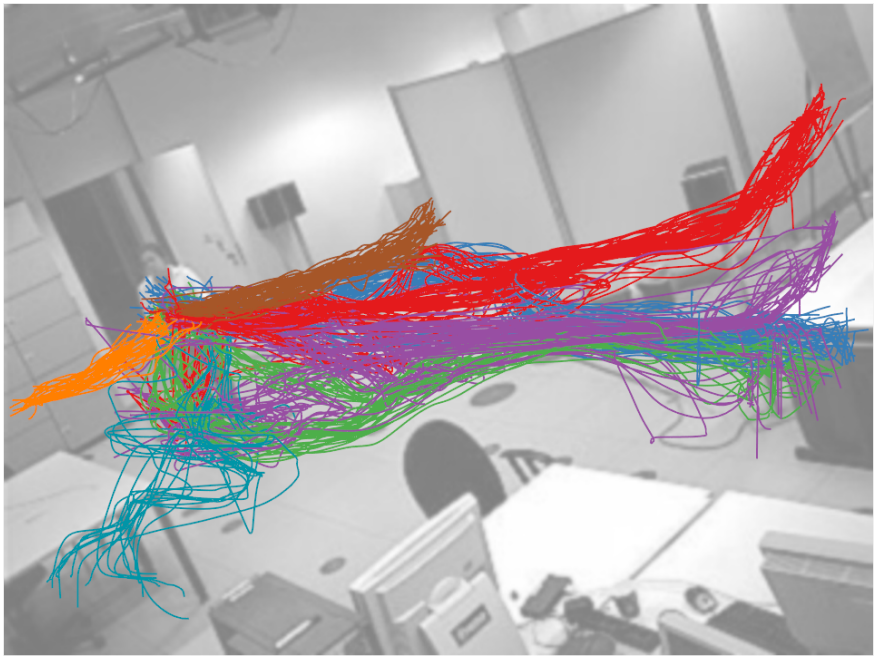


Figure 5.2: Trajectories superimposed on an image. The 7 colors denote differently classified trajectories.

Understanding Decision Trees

The first example is an application of data analysis in computer vision. This application was used to evaluate decision tree navigation in a qualitative study with a domain expert. She was interested in classifying trajectories of people walking into a given room (see Figure 5.2). A semi-automatically labeled set of trajectories each belonging to one of 7 classes was taken as input. The 321 trajectories were sampled at 250 positions and transformed to a feature vector representation consisting of the final x and y screen coordinates (*x end* and *y end*), a standard deviation of position in x and y screen directions (*x std dev* and *y std dev*) calculated at every 10th position using a local window of 21 positions on the path and a local direction (*angle*) based on the direction between 21 samples calculated at the same positions. The resulting feature space was ((250-20)/10)*2+2=48 dimensional. Matlab's classregtree method was used to create the classifier with supervised learning, which uses Breiman's Classification and Regression Trees method [BFSO84]. The resulting classifier is a binary decision tree, where each branching node is split based on the values of a coordinate of the input vector. The trained decision tree used 10 data dimensions and consisted of 12 inner nodes, see Figure 5.3. The domain expert was interested in analyzing the resulting decision tree. She was used to Matlab decision tree plots similar to the plot in the "Decision Tree" widget in Figure 5.3. She regularly plotted trajectories over an input

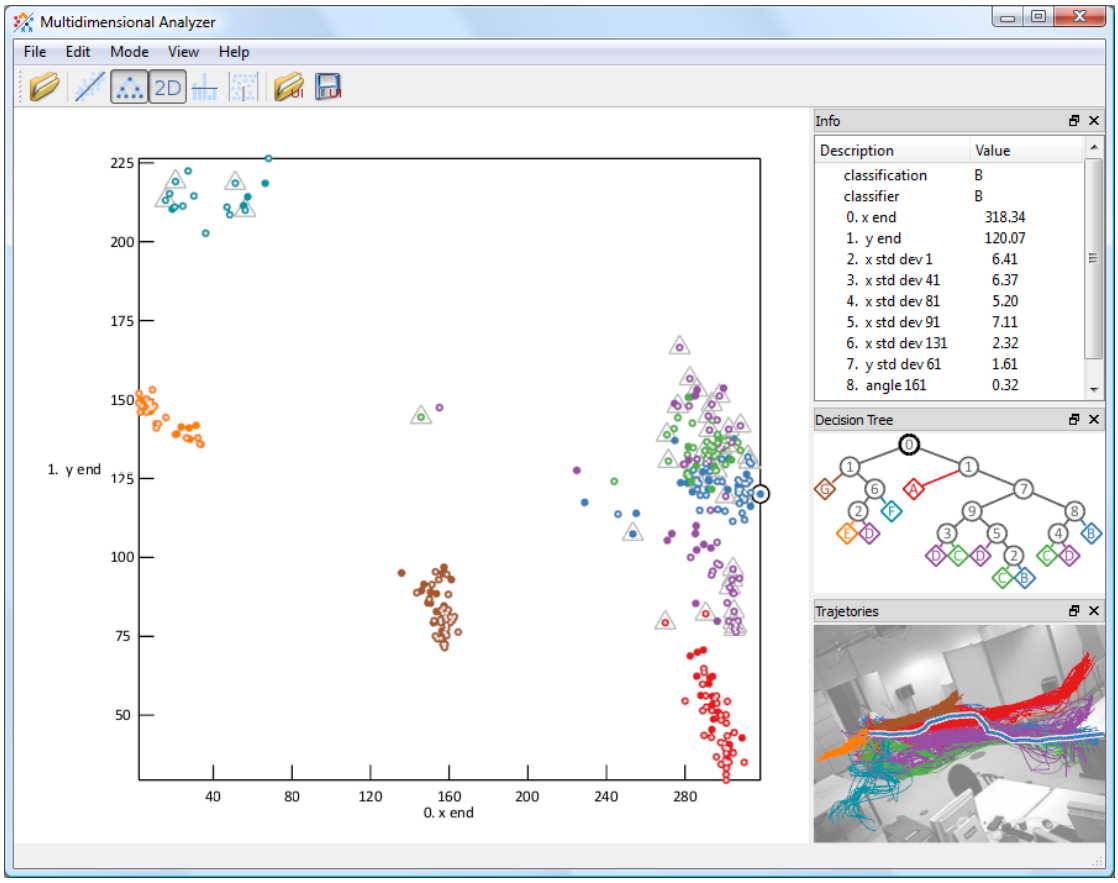


Figure 5.3: Screenshot of MDA showing a scatter plot in the main part of the window (left). In the center-right, a decision tree is presented; inner nodes are drawn as circles labeled with the dimension considered, leaf nodes are drawn as squares labeled with the classifier result. In the upper-right corner, the values for the different data dimension of the selected data item (marked with a black ring in the scatter plot) are printed. Colors represent the classification. Points of the scatter plot (left) that were not part of the training set are marked with white dots inside; classifier errors are marked by gray triangles. In the bottom-right, the trajectories are rendered in their 3D spatial context. All four views are linked, propagating the selection of data items between the views. See the accompanying video (`decTree.avi`).

image like in Figure 5.2 in order to analyze the trajectories. However, she had no prior experience with multidimensional analysis tools like GGobi. Since there is a high density of trajectories in the image, she typically plotted just one or a small group of trajectories during analysis. This was cumbersome because each trajectory had to be examined together with additional information like the features used by the decision tree. The objective of this process was to identify incorrectly labeled trajectories.

With MDA, the domain expert investigated if the classifier performed as expected and, if not, she tried to find incorrectly labeled and incorrectly classified trajectories. The following interaction modes were explained to the expert:

- Within the scatter plot: zooming and panning by mouse navigation.
- Switching color coding between labeling and showing decision tree results.
- Selecting a trajectory in the scatter plot with the mouse.

A different data set consisting of 241 trajectories had 9 clusters. A high-dimensional feature vector using all 250 x and y positions in addition to the features described above was used. A decision tree was trained which used 15 data dimensions and consisted of 17 inner nodes. Figure 5.4 shows the tool during the evaluation session. The first subfigure (top left) shows the scatter plot corresponding to the root of the decision tree, with the data dimensions (8, 12) mapped to the (x, y) axes. The user chooses to descend to the right child of the node; the application grays out and minimizes the points of the scatter plot that are not on the right side of the node (subfigure 2). While descending to the child node, the application needs to change the axis mapping to (8, 5). This looks like a 3D rigid body rotation presented as snapshots from the animation (subfigures 2–7). At the child node, the user selects the left child (data dimension 4) and the other points of the scatter plot are grayed out by the system (subfigure 8).

The expert reported that she had gained insight related to the labeling of the trajectories that had significant errors and thereby degraded the decision tree obtained by the learning algorithm. She found the application and the decision tree navigation intuitive. Even without previous exposure to similar systems, there was no steep learning curve. In summary, the domain expert found decision tree navigation useful.

Understanding Multidimensional Data

Scatter plot navigation with decision trees is not only useful to understand decision trees, but also to get insight into multidimensional data. The key observation is that decision trees use the dimensions of the data that are most suited to separate differently labeled data points. This property can be utilized to guide the exploration of multidimensional data sets. The process consists of two stages: first, a decision tree is trained from the input data; second, the navigation technique is applied for interactive data analysis.

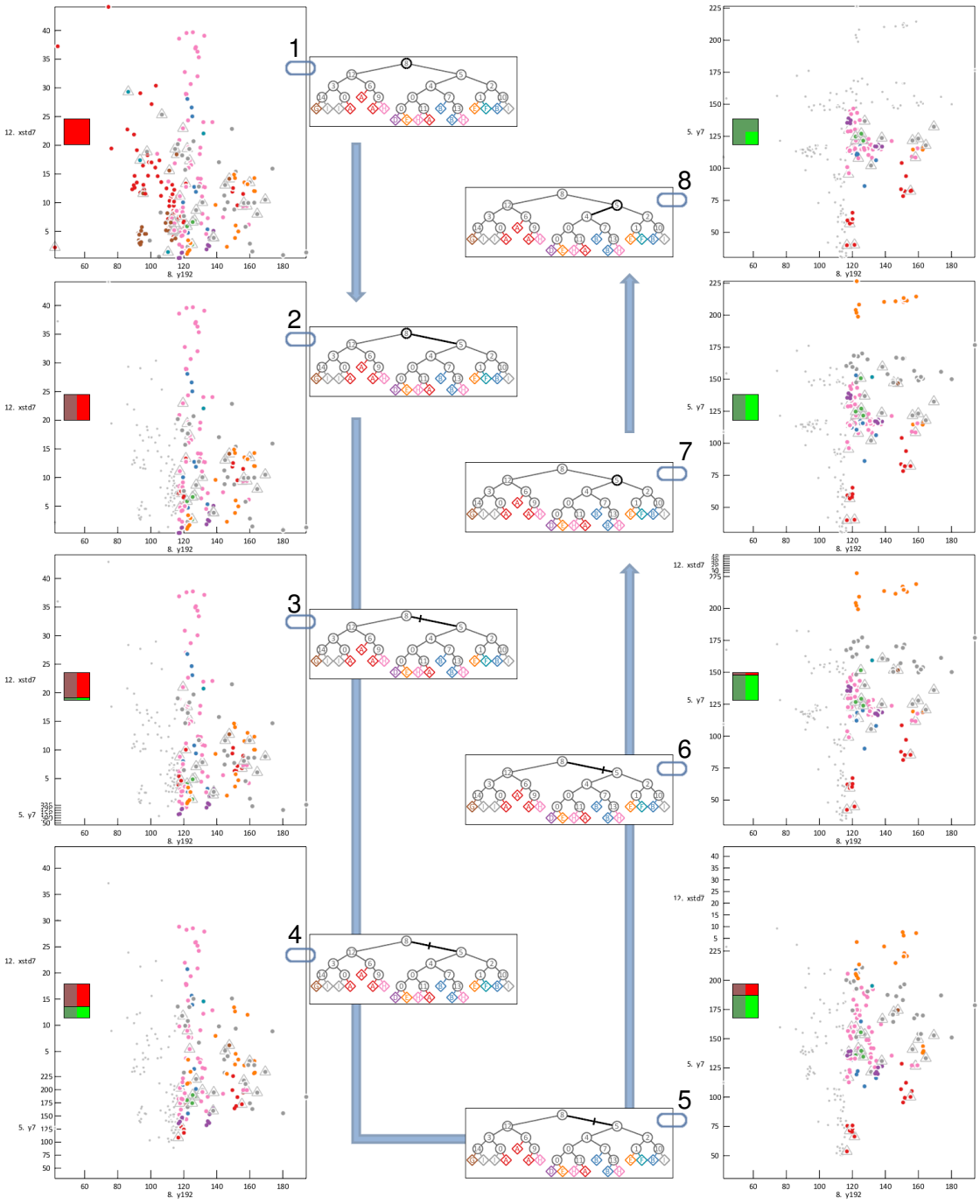


Figure 5.4: Navigation with a decision tree in a video trajectory data set. Subfigures are ordered counterclockwise. The red-green box indicates the rotation and the grayed out subset of points. See the accompanying video (`decTree.avi`) for an animated presentation.

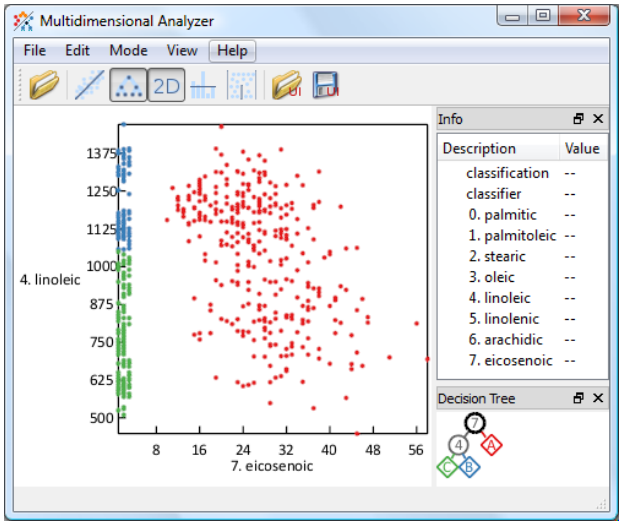


Figure 5.5: Scatter plot navigation using the regions of the "olive oil" data set. Regions: A: Region 1 (south), B: Region 2 (Sardinia), C: Region 3 (north).

Data analysis is illustrated for the well-known "olive oil" data set [FALT83], briefly used in Section 4.4.2. This benchmark data set consists of olive oils coming from nine different areas of Italy given their composition with respect to eight fatty acids. The areas are grouped in three regions. Using regions as class attribute results in a small tree with only three leaf nodes and two inner nodes. Therefore, a single 2D scatter plot is sufficient to separate the three regions as shown in Figure 5.5. Note that these two dimensions can be found automatically, and no user interaction is required.

A decision tree can also be trained using areas as class attribute. Then, the nine areas cannot be separated as easily, see Figure 5.6. The root node separates the areas North-Apulia, Calabria, South-Apulia, and Sicily from the other five areas. The left branch consisting of six inner nodes separates the other five areas from each other. The right branch, however, requires 15 inner nodes to separate the four regions. By ignoring Sicilian oils—similar to Caragea et al. [CCH01], who analyzed the "olive oil" data set using SVM classifiers—the remaining three regions can be separated by just four inner nodes.

5.2 Visual Exploration of Classifiers for Hybrid Textual and Geospatial Matching

The last section incorporated a decision tree as a means of scatter plot navigation. In contrast, here visualization techniques are presented which support natural language processing domain experts in solving a specific classification task using decision trees.

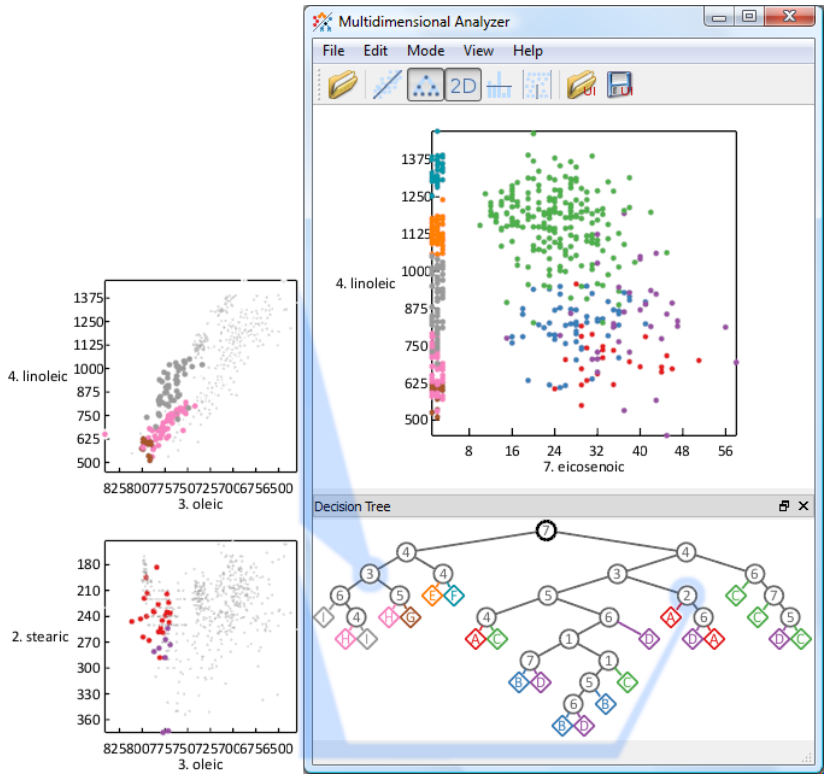


Figure 5.6: Scatter plot navigation using the areas of the “olive oil” data set. Areas: A: North-Apulia, B: Calabria, C: South-Apulia, D: Sicily, E: Inland-Sardinia, F: Coast-Sardinia, G: Umbria, H: East-Liguria, I: West-Liguria. The two scatter plots shown separately on the left-hand side are displayed during decision tree navigation to inner nodes “3” and “2”, respectively.

Additionally a visualization technique for the decision tree boundary is introduced.

The availability of large geospatial data from different sources has dramatically increased, but for the usage of such data in geo-mashup or context-aware systems, a data fusion component is necessary. To solve the integration issue classifiers are obtained by supervised training, with feature vectors derived from textual and geospatial attributes. In an application example, a coherent part of Germany was annotated by humans and used for supervised learning. Annotation by humans is not free of errors, which decreases the performance of the classifier. This section shows how visual exploration techniques can be used to efficiently detect such false annotations. Especially the textual features introduce high-dimensional feature vectors, where visual exploration becomes important and helps understand and improve the trained classifiers. Particular technical components used in the system presented here are scatter plots, multiple coordinated views, and interactive data drill-down.

This application is studying the usefulness of visual exploration for designing, debugging, and optimizing machine-learning techniques in geospatial text-based matching. This work was done together with André Blessing and Hinrich Schütze, who are machine-learning and natural language processing experts, and was realized in the SFB 627 (Nexus), an interdisciplinary research project that covers areas of computer science, electrical engineering, and philosophy related to context-aware information systems, results are reported in [SBSW09].

Context-aware systems adapt to changing environmental conditions, but they need information of the environment. The Nexus [DHN+04] platform provides data for context-aware systems by representing the real world through an internal *world model*. Data in the world model is heterogeneous and unstructured. Some data items are geo-referenced or geometric, others are often complex textual representations. For an open system like Nexus it is obligatory to handle data from several data providers, making automatic data integration an important task. In general, these data sets are not equally modeled and describe different—although largely overlapping—sets of objects. Approaches from the natural language processing and machine learning (ML) communities can be combined to address these new issues. However, the complex mechanisms in NLP and ML are hard to understand and error analysis and system optimization are not easy. Here visual exploration is used to facilitate these tasks.

The focus here is on the aggregation of geospatial data from different providers. In particular, each provider may have different measurements on which they base the model, resulting in data quality that varies from provider to provider. The goal is to create a homogeneous model out of underlying heterogeneous models. A central issue is to find coherent instances in the different data sets.

Data sets from two exemplary providers are considered, covering the same geospatial area. The data sets do not have the same schema and use different attributes. Interestingly, there are also intentional differences between the data sets; e.g., providers include

fictitious objects used as watermarks. Data sets also differ in quantity due to variations in density and types of geo-objects modeled. The approach is to first create small training sets—consisting of coherent and incoherent instances of the two providers—manually and then to train a model that delivers correct coherence pairs for the whole data sets. After this step it will be possible to perform an assessment of consistency and completeness of the two data sets and to derive a new quality model for the data.

There is a trade-off between domain optimized solutions for the given data and an approach that can be universally used on other domains and similar tasks. Minimizing manual labor was central issue considered during the development of the classifier. Therefore, the goal was to develop a classification technique that can be trained on a geographically small coherent region and yields proper results in geographically distant regions. Mostly established visualization components for analyzing and improving textual and geospatial matching are used. For example, 2D point and line plots, color coding, 3D scatter plots, and visual interaction and navigation techniques were employed. Where needed, specialized components were added, e.g., a hyperplane visualization of classifiers.

5.2.1 Related Work

Thomas and Cook [TC06] summarize the roots and the goals of visual analytics (VA). VA was defined as "the science of analytical reasoning facilitated by interactive visual interfaces". Visualization develops new *visual mapping* and *interaction techniques* and defines *insight* traditionally as aim, see van Wijk [vW05]. Besides developing new *visual mapping* and *interaction techniques*, VA should facilitate *analytical reasoning* and fast *decision making*. However, van Wijk proposes to use the *economic value of a taken decision* [vW05] instead of *insight* as an aim for visualization too, similar to VA. On the other side, Keim et al. [KMS+08] define *insight* as primary goal of VA and model the process as hypothesis testing. Additionally to visualization, automated analysis methods, like statistics and data mining techniques, are used in VA. Data mining uses ML methods like supervised and unsupervised learning. The system presented in this section visualizes data classified by decision trees, but it does not employ any ML methods for the data analysis. However, the system also employs a VA method: the human annotation is analyzed automatically and false annotations can be visualized in the map view.

Garg et al. [GNRM08] explain the successful combination of ML approaches and VA; they also use scatter plots, with focus on navigation in high-dimensional spaces, but do not deal with geospatial data. Robnik-Sikonja and Kononenko [RSK08] use scatter plots to compare the behavior of different classification methods on artificial data sets. Chen [Che08] uses VA to select features with high information gain to develop a classifier that is similar to the feature selection used here.

Previous work on matching of geospatial data includes the work of Sehgal et al. [SGV06]. They match data sets of Afghanistan from two different providers, using three

different string similarity measures and physical distance. First they consider each feature independently and compare different threshold values for the corresponding feature. In the last step they combine the features and learn a function to weight the different features. The main difference to the work presented here is that they do not use any visual analysis, and the feature set presented here is also a richer one. A follow-up publication [KSG07] describes a graphical tool for entity resolution. The tool provides many configuration options for user-driven interactive semi-automatic matching by filtering the matching candidates, but it does not directly visualize all matching candidates to evaluate the classification.

5.2.2 Application Background

Requirements

Visual analysis should assist the development of classifiers in several tasks. First, visual exploration of data is needed to get a better understanding of their characteristics. This leads to insights that can help design features. With the visualization of the high-dimensional feature space the impact of each feature can be analyzed and optimized. Finally, the performance of the classifier should be evaluated by visual tools.

Data Sources

The data integration is simulated with two data sets from different providers. One data set is commercial and provided by NAVTEQ (`http://www.navteq.com`), which is one of the two leading data providers for navigation systems. The other data set is freely available (Creative Commons License) and is managed by the Geonames (`http://www.geonames.org`) project following the idea of Wikipedia. Each user has the opportunity to add, modify, and delete data. The disadvantage of such a public collaborative resource is that it is not clear how to define a homogeneous quality model and to check the consistency of the data set. Another disadvantage of the Geonames data set is that it is based on free but inexact data sources. For example, the coordinates of many locations contain only hours, minutes and no seconds in the sexagesimal notation. This results in heavily rastered locations. Therefore, this is a challenge and stress test for the aim of finding the right corresponding data pairs. From both providers only the data layers are used that represent villages and quarters. This raises also the complexity of the matching problem because these areas constitute the lowest level of administrative area and are not well defined in many cases.

Classifier

Two village objects are a match if both refer to the same real world object. For making the matching decision, the geospatial positions and the strings of the names of the two

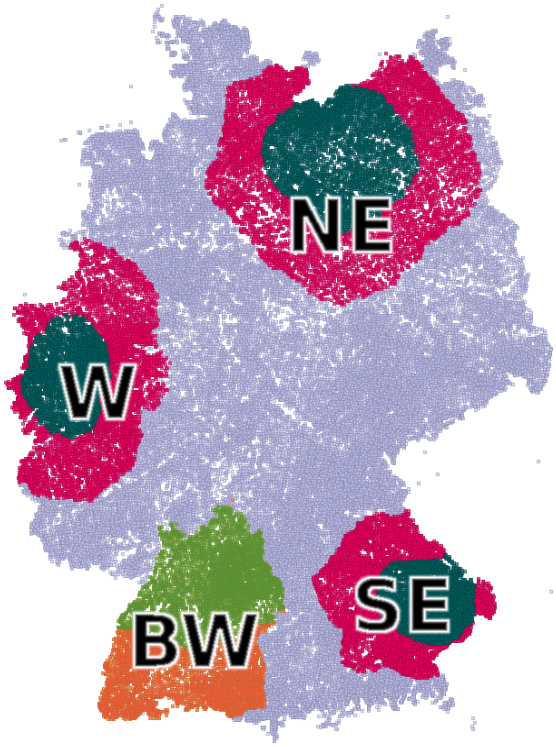


Figure 5.7: The area on the lower left side (corresponding to the German state of Baden-Württemberg) is used as development set. It is further split into a training (north BW) and a test set (south BW). The other three areas (W, NE, and SE) are only used for the final evaluation.

objects are used. Initial experiments showed that a *search space* of 10 kilometers in the environment is sufficient to find the match for any item. For the allowed deviation of the names and positions no commonly usable rules can be defined. But in most cases the decision can be made by a human judge with high inter-judge reliability. In some marginal cases additional sources, like the homepage of the village or an online encyclopedia, must be considered to make the right decision.

The matching component is implemented as a binary classifier that decides if a pair of village objects from the two data providers refer to the same location. A manually labeled training set is used to train a decision tree classifier.

Data Selection

Figure 5.7 shows all labeled sets for the experiments. Data sets from four distinct regions of Germany are used to compensate for differences in their regional properties. All regions are chosen by a snowball selection process, which is used in many machine learning tasks [SNB+08] as well. It makes sure that each of the four sample regions is

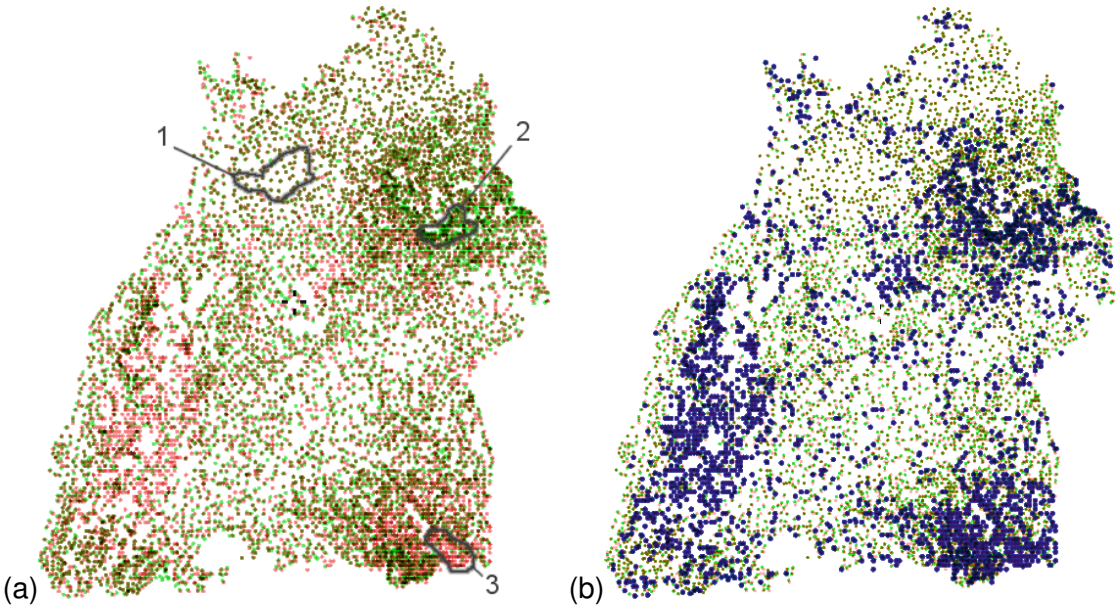


Figure 5.8: (a) Map showing the geo-position of data entries for Baden-Württemberg from the data providers NAVTEQ and Geonames, colored green and red respectively. (b) Highlighted data items (large blue points) have no correspondence.

well-connected as opposed to a set of disconnected points lacking the local information needed for matching. Further, each region is split into two parts: one for training/learning and the other for evaluation. In the development phase of the work only the south-west region (which is equal to the German state of Baden-Württemberg) was considered. The other three regions are only used for final evaluation tasks.

5.2.3 Classifier

Development Set

The goal of the classifier is to find a one-to-one correspondence between the NAVTEQ and Geonames data items. Figure 5.8(a) shows the distribution of the 6497 NAVTEQ (green) and the 7904 Geonames (red) data items in Baden-Württemberg. Each data item is drawn semitransparent with additive blending enabled. Based on the density and proximity of data points in region 1 (in top left part of Figure 5.8(a)) a good correspondence can be expected. In region 2 the number of NAVTEQ data items is larger whereas in region 3 the number of Geonames data items is larger. This results in many data items without correspondence in the respective other database.

Annotation Process

For the supervised approach, annotated data is needed for training and evaluation. The data sets were annotated by two annotators. Although inter-annotator agreement was good [EG04] (see Table 5.1, where $(\kappa > 0.8)$ is good and $(0.8 > \kappa > 0.6)$ is satisfactory) the subsequent visual exploration highlighted errors in the annotation. A matching candidate was defined as a match if both annotators annotated it as a match.

Table 5.1: Kappa values for annotated regions BW (Baden-Württemberg), SE (south-east Germany), W (west), and NE (north-east).

	BW	SE	W	NE
κ	0.92	0.89	0.77	0.84

The annotation process resulted in 5682 corresponding items. The remaining 815 NAVTEQ and 2222 Geonames items are highlighted in Figure 5.8(b): the non-corresponding items are concentrated in regions with high density differences (compare Figure 5.8(a)). The correspondences derived by the annotation can be represented as lines as shown in Figure 5.9. Corresponding items that are close together result in very short lines, not prominent in the image. Correspondences with large geospatial differences result in long lines, which are immediately visible and can be further examined. A long line does not automatically point to annotation errors. In some cases the quality of the Geonames data is low, because it can be edited by everyone.

Annotation by humans is not free of errors. Figure 5.9 also shows annotation errors, found automatically, since each item can have at most one corresponding item by definition.

Feature Design

The feature set was optimized on the development set of Baden-Württemberg in several iterations. In each iteration, features were developed that could distinguish matching candidates that were not handled correctly in the previous iteration.

The spatial distance between the source and destination objects is represented in the (*logDist*: $\log_{10}(\text{distance})$) feature.

The similarity between names started with the use of the feature *sim*: Trigram similarity, which is based on a trigram representation (Stuttgart $\rightarrow$ {__S, _St, Stu, ..., art, rt_, t__}). The similarity score of two names, a variant of the Jaccard index, is calculated by counting all equal trigrams and finally dividing them by the number of trigrams. Experience showed that it is not possible to store all information related to names in one feature. Therefore several additional features were developed in collaboration with the natural language processing experts which are presented below.

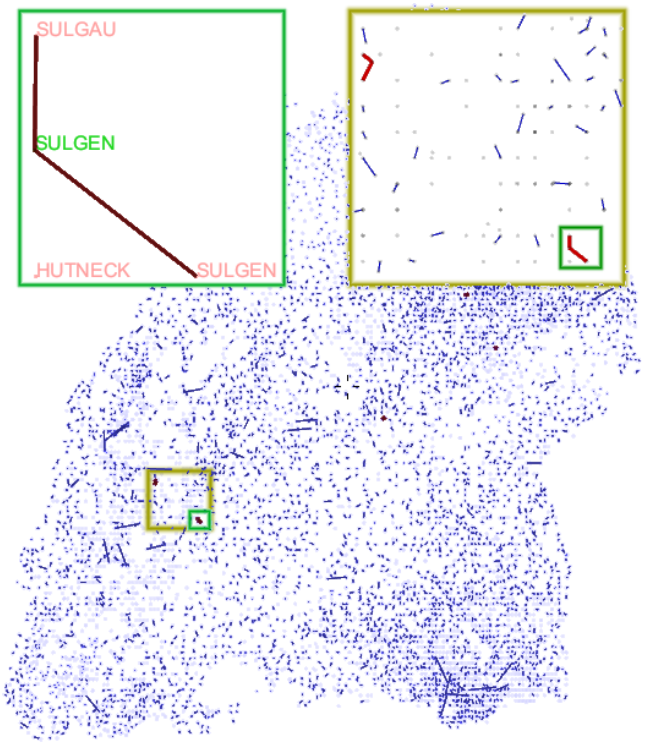


Figure 5.9: Corresponding NAVTEQ and Geonames data items connected with lines. Automatically found annotation errors are colored red.

Decision Tree Classifier

As already mentioned in the introduction to this chapter, decision trees use a white box model which makes it easier to analyze and understand classification decisions than with many other classifiers. For the decision tree learning J48, an open source Java implementation of Quinlan's C4.5 algorithm [Qui92] was used, included in the freely available data mining software Weka. For each matching candidate, which is a pair of an object from Geonames and NAVTEQ, a feature vector is calculated, consisting of the *logDist* and *sim* features.

To compare the progress of the development, some metric to measure the performance is obligatory. The classifier is trained to derive the same result as obtained by the annotation process. The classifier is not able to always derive correct results. The classifier results can be categorized by the following well known categories: true positive (TP)—the classifier finds *a correspondence* between two *corresponding* items; true negative (TN)—the classifier finds *no correspondence* between two *non-corresponding* items; false positive (FP)—the classifier finds *a correspondence* between two *non-corresponding* items; and false negative (FN)—the classifier finds *no correspondence* between two *corresponding* items.

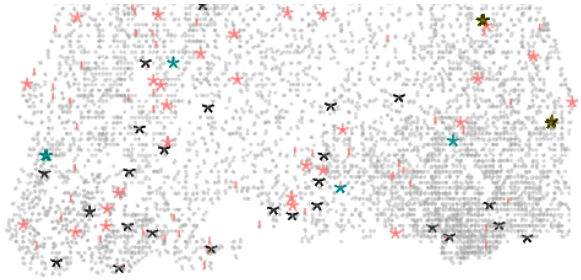


Figure 5.10: Star glyphs showing errors of the classifier. Each point of the star denotes an error in the respective iteration. Color coding of error types: red corresponds to false negative, black: the classifier finds a correspondence between two non-corresponding items that *both* have *a correspondence* to other items; cyan: the classifier finds a correspondence between two non-corresponding items that *both* have *no correspondences* to other items, and yellow: the classifier finds a correspondence between two non-corresponding items where the *Geonames* data item has *a correspondence* to another item but the NAVTEQ data item has no correspondence.

Visual-Aided Classifier Development

The map-based visualization of the first classification results showed that the basic algorithm is not sufficient because nearby classifications have an impact on each other. To overcome this issue an iterative algorithm [Bes86, LG03, JNG04] is applied. In each iteration the classification result of the previous iteration is taken as input. In the *bootstrap step* an initial classifier is trained on the training set. This classifier is then applied to the training set; the classifier results are appended to the feature vector, which is the input for the second classifier. This process can be applied iteratively. Two additional features model the previous assignments. The feature *preScore* values the score of the previous iteration. The more important new feature is the rank value *rank*. The ranking is built over the scores of matching candidates that include the same Geonames object.

To analyze the performance of a classifier, false classifications are visualized with the error category being color coded. To visualize the results of each iteration, a star-shaped glyph is drawn whose points denote errors in up to five consecutive iterations. Figure 5.10 shows a classifier with two iterations that uses the *sim* and *logDist* features in the first iteration and additionally the *preScore* and *rank* features in the collaborative second iteration. In the first iteration many correspondences are missing as can be seen by the amount of the red star glyphs where the first point of the star is present. In the following iteration many missing correspondences are found (red glyphs where *just* the first point of the star is present); but false positive correspondences are introduced (glyphs where the first point of the star is missing). This shows that the features are not descriptive enough to derive a proper classification.

Figure 5.11: Line connecting “EFRINGEN” and “EGRINGEN” showing a classifier error. On the right side of the star glyph, training set flag and classifier score are displayed.

When zooming in, a line connecting the two items is drawn with the classifier score and whether the match was part of the training set. This visualization presents all information necessary for diagnosing what went wrong in a small local region in an intuitive way. The design of additional features for improved accuracy has been greatly facilitated by this visualization. Certain names in the Geonames database fall exactly to the same position as can be seen in Figure 5.11 where the names of the regions are written next to the geo-position.

High-Dimensional Feature Space

After the first explorations the need for more similarity metrics became obvious.

Now 5 of the 8 string similarity metrics used in the system are presented. *levenshtein*: Levenshtein distance [Lev66] between the two names. The boolean feature *partof*: Part-of relation first splits names into more tokens if they contain separation characters like parentheses, hyphens, and slashes and then returns 1 if one of the tokens is a substring of the name in the other data set and 0 otherwise. Sometimes names are supplemented by additional expressions. In Germany, spa towns often start with the expression *Bad*. For some spa towns, a variant without *Bad* is used, e.g., “Urach” instead of “Bad Urach”. In the same way additional prepositional phrases containing spatial information about a river (“am Neckar”, compare to “upon Tyne”) can be added to names. As in the above case, these specifications are often used optionally. Therefore, two special similarity measurements, *fw* and *bw* were defined, that compute the length of the longest common prefix or suffix divided by the length of the shorter name. *hyph* is true iff one of the names includes a hyphen or a slash.

The density analyses and the classification errors of the previous features call for features that represent the geospatial surrounding of the matching candidates. 6 features were implemented belonging to this class. Just one example: the *sim_05* feature counts other possible candidates in the vicinity that have *sim* value higher than 0.5.

The errors introduced during the classification can be divided in two classes: *systematic errors*, which are likely to be learned by the classifier, and *non-systematic errors*, which are not learned. Non-systematic errors can be detected more easily than systematic

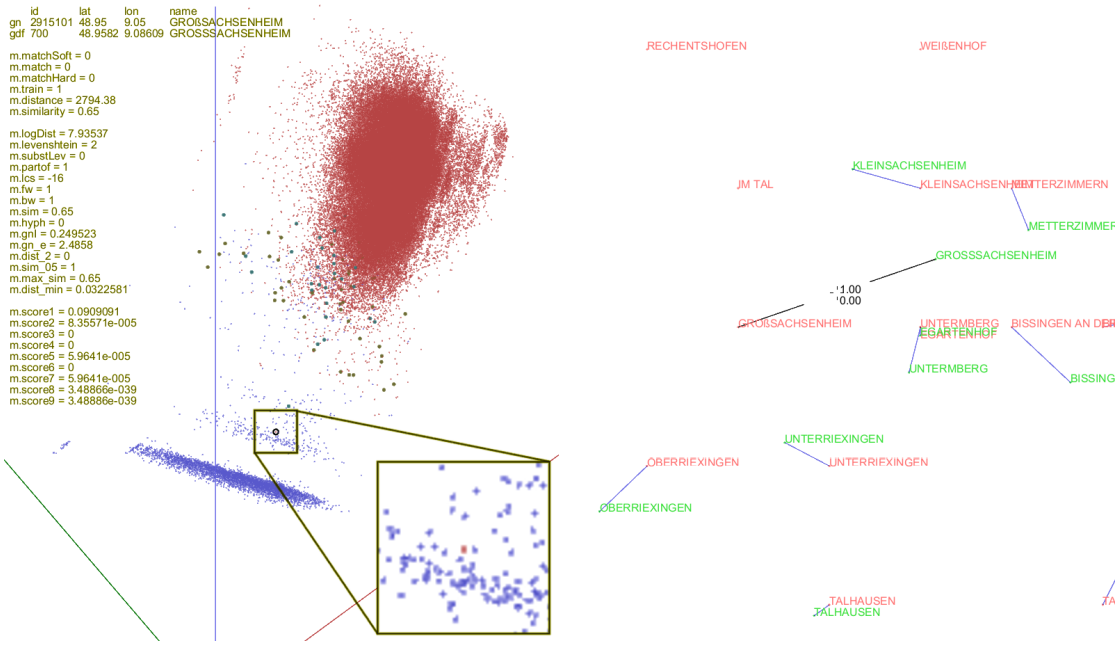


Figure 5.12: Linked views. The scatter plot on the left showing matching candidates. True positives, true negatives, false positives, and false negatives colored blue, red, green, and yellow respectively. The map on the right shows the position of the selected item.

ones e.g., by examining the "false" classified items with the technique presented above.

Each matching candidate is represented as a feature vector. To detect systematic errors, which are learned by the classifier, the feature vectors used by the classifier need to be examined. The feature vectors used by the classifier are high-dimensional, one dimension for the distance, 8 dimensions for "name distance", and 6 dimensions for the geospatial surrounding matching candidates. The feature space can be normalized to unit size and mapped to 3-space with a modified FastMap algorithm. FastMap [FL95] maps points from n-dimensional space to k-dimensional space ($n \leq k$) with the focus on preserving distances between points. The original FastMap algorithm was modified to take into account the user classification according to the supervised PCA technique [KC04]. The scatter plot showing the 159 973 feature vectors for Baden-Württemberg can be seen in Figure 5.12. Please note that the data items in the scatter plot can be recognized much better on a computer screen than on paper due to higher contrast and larger space. The selected negative match next to the positive matches was learned correctly by the classifier (true negative). However it can be easily detected in the scatter plot. By selecting a matching candidate in the scatter plot, the map on the right jumps to the selected position and allows examining the neighboring items. The selected matching candidate "GROßSACHSENEIM"–"GROSSSACHSENEIM" was annotated as no

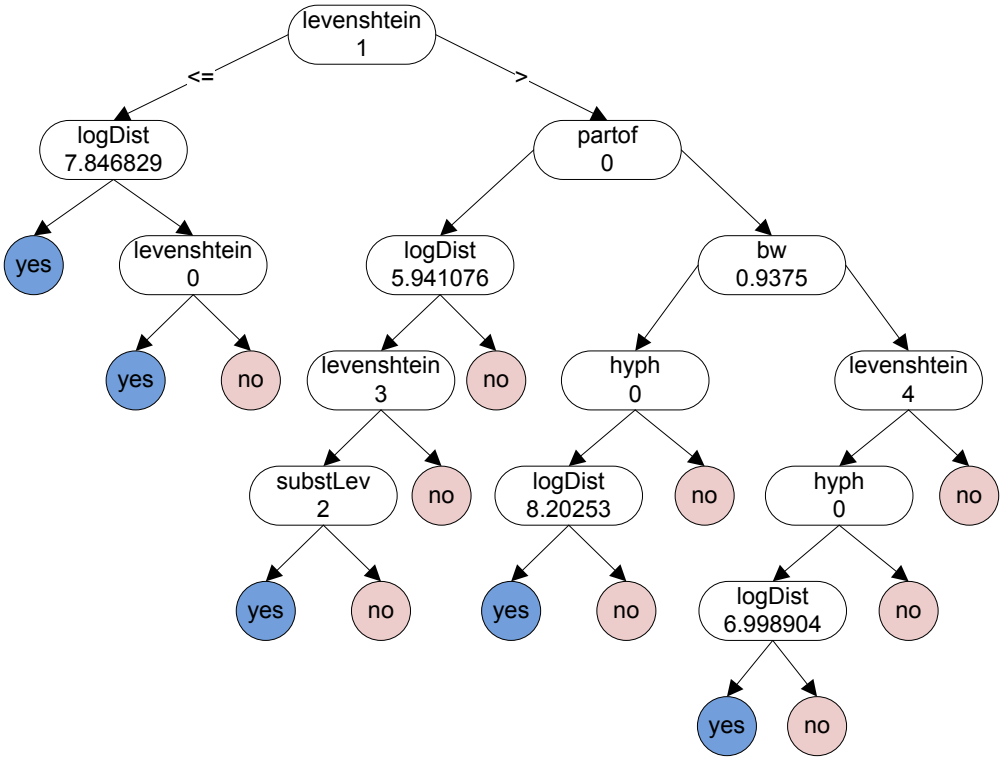


Figure 5.13: Decision tree used in a classifier iteration.

match, but since the lowercase German letter ß does not have a corresponding uppercase counterpart it is written as SS when writing uppercase. Therefore the matching candidate is a match. This is an example of the effectiveness of the visualization in identifying possible improvements to the underlying representation.

5.2.4 Decision Tree Hyperplanes

Figure 5.13 shows a decision tree that is used by a classifier iteration. A decision tree takes a subspace of the feature vector space dimensions to classify each vector as positive or negative. In this case 6 of the 15 dimensions are used by the decision tree. The decision tree separates the space into two regions. This section will present a visualization of the separating co-dimension 1 manifolds to get a better understanding of the classifier. Tibshirani and Hastie [TH07] illustrated individual hyperplanes as lines in a 2D scatter plot; Urbanek [Urb08] displayed some hyperplanes defined by decision trees similarly; both did not consider all hyperplanes defined in the decision tree. Cook et al. [CCH04] also visualized the hyperplane defined of a Support Vector Machine by points sampled on the hyperplane. H-BLOB [SBG00] represents clusters by semitransparent surfaces in 3D, which is a similar representation to the semitransparent rendering of hyperplanes

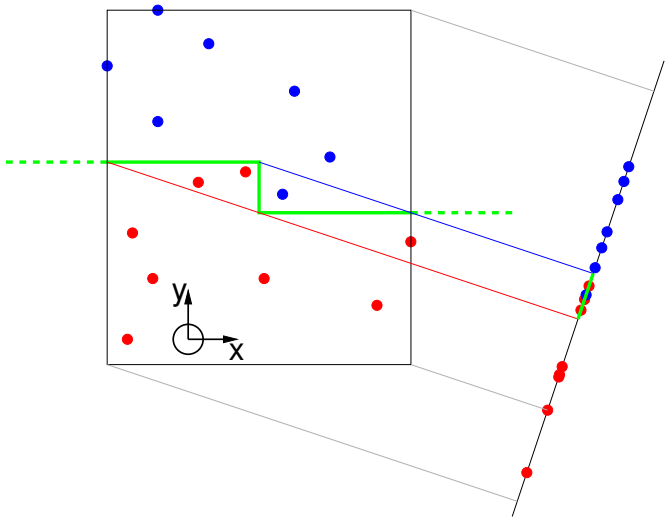


Figure 5.14: Two-dimensional feature vectors, classified as red and blue separated by 2-space hyperplanes (green line). In the right part of the image a projection to 1-space is shown; the separating hyperplanes within the domain are mapped to a line segment.

introduced here. Figure 5.14 shows how a decision tree separates two-dimensional feature vectors. The dots represent feature vectors, blue and red color represents the classification. The decision tree divides the space along the axes and separates the differently classified feature vectors. Since the feature vectors used by the decision tree classifier are high-dimensional, for visualization they need to be projected to a lower-dimensional space as indicated in Figure 5.14. The hyperplanes defined by the classifier are not bounded, therefore their projection would cover the whole domain. However, the hyperplanes can be clipped by the bounding volume of the feature vectors, so that the hyperplanes project to a finite volume, which is a line segment in the 1D case. Only blue points are projected to the one side of the line segment, to the other side only red points. Within the line segment there are blue and red points that cannot be visually separated in the projection.

Figure 5.15 visualizes the hyperplanes of the decision tree defined in Figure 5.13. The feature vectors were projected orthogonally to the features *levenshtein*, *logDist*, and *partof*. In 3-space the projected hyperplanes are illuminated and rendered opaque with the projected feature vectors. Hyperplanes that divide regions based on the selected features project to planes in 3-space. Hyperplanes dividing regions based on other features project to volumes containing all feature vectors that were divided by them.

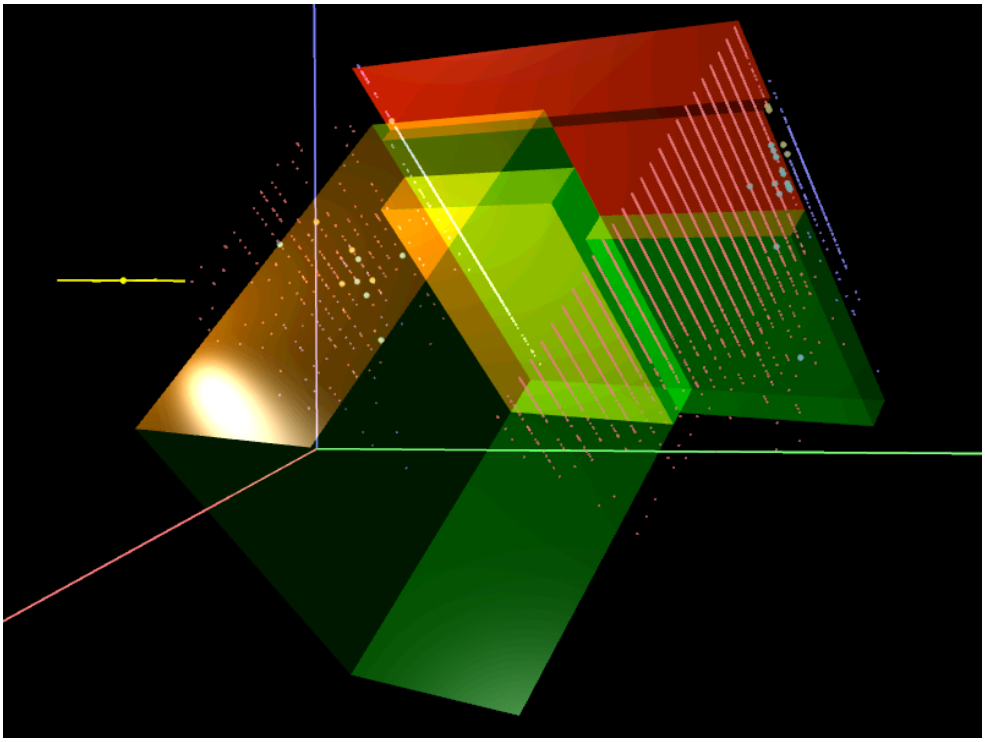


Figure 5.15: 6-space hyperplanes used by the classifier, lit in 3-space projection.

Decision Tree Hyperplane Rendering

In Figure 5.12, FastMap was used to obtain a quite good separation of the differently labeled items. This section presents how to apply the decision tree hyperplane rendering technique not only to projections where certain data dimensions have been mapped to certain axes, but to allow us to map linear combination of dimensions to the axes.

Each leaf node of the decision tree defines an orthotope. Always when differently labeled orthotopes are next to each other, a hyperplane separating the two regions needs to be drawn. A straightforward implementation checks each orthotope with each other, requiring $(h(h-1))/2$ checks (h being the number of leaf nodes). For decision tree presented in Figure 5.13 there are 14 orthotopes and 46 resulting hyperplanes.

It is known that an n-cube has 2^n nodes (vertices); and there are $2^{n-k}\binom{n}{k}$ k-cubes on its boundary (k<n). The triangle count is two times the number of 2-cubes (k=2). Therefore in an f-dimensional feature space each hyperplane of the decision tree is an $n = f-1$-dimensional orthotope. According to the formulas, each hyperplane has 2^n vertices and $2^{n-2}\binom{n}{2} = 2^n(n^2-n)/8$ quads are needed for rendering. For the 15-dimensional feature space considered here, therefore, $2^{(15-1)-2}\binom{15-1}{2}$=372 736 quads are required for each hyperplane. This leads to a total triangle count of over 34 M triangles, which is too large

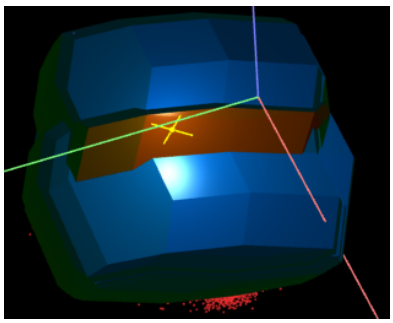


Figure 5.16: Hyperplanes used by the classifier, lit in 3-space projection where linear combination of dimensions have been mapped to the axes.

to be drawn interactively even on modern graphics hardware.

However, this huge triangle count can be greatly reduced when considering the visibility of quads for a given axes mapping. Only a small fraction of the quads are visible, namely the quads having all the vertices on the convex hull of the 3D projection. The number of vertices on the convex hull is $n^2 - n + 2$ resulting in $n^2 - n$ quads (without proof). Note that the number of hull-quads does not grow exponentially—like before the number of all quads—but only squared with the number of dimensions. For the given case resulting in only just under 17 k triangles, which can be drawn interactively. For the implementation it can be exploited that whether a vertex is part of the convex hull of the orthotopes or not, is independent of the location and size of the orthotope. Therefore, it is sufficient to project a single orthotope to 3-space, run a convex hull algorithm, and use the results for all hyperplanes.

Figure 5.16 shows such a decision tree hyperplane rendering, here PCA has been used to calculate the projection axes. Therefore, the projection axes do not point in the direction of a certain feature, but in a linear combination of features. Hence the hyperplanes have an accordingly complex geometry. The visualization technique has been integrated into the Multidimensional Analyzer as well, see Figure 5.17.

Unfortunately, for the given case the decision tree hyperplanes cover a large volume in 3D and most of the items are inside this volume, therefore a clear separation of the differently labeled items cannot be achieved this way. The results obtained suggest that the decision tree hyperplane rendering is best applied when just one data dimension is assigned to each axis and not a linear combination, because this way just one single data dimension contributes to the size of the hyperplane in each axis direction, and this tends to reduce the volume of the resulting decision tree hyperplanes.

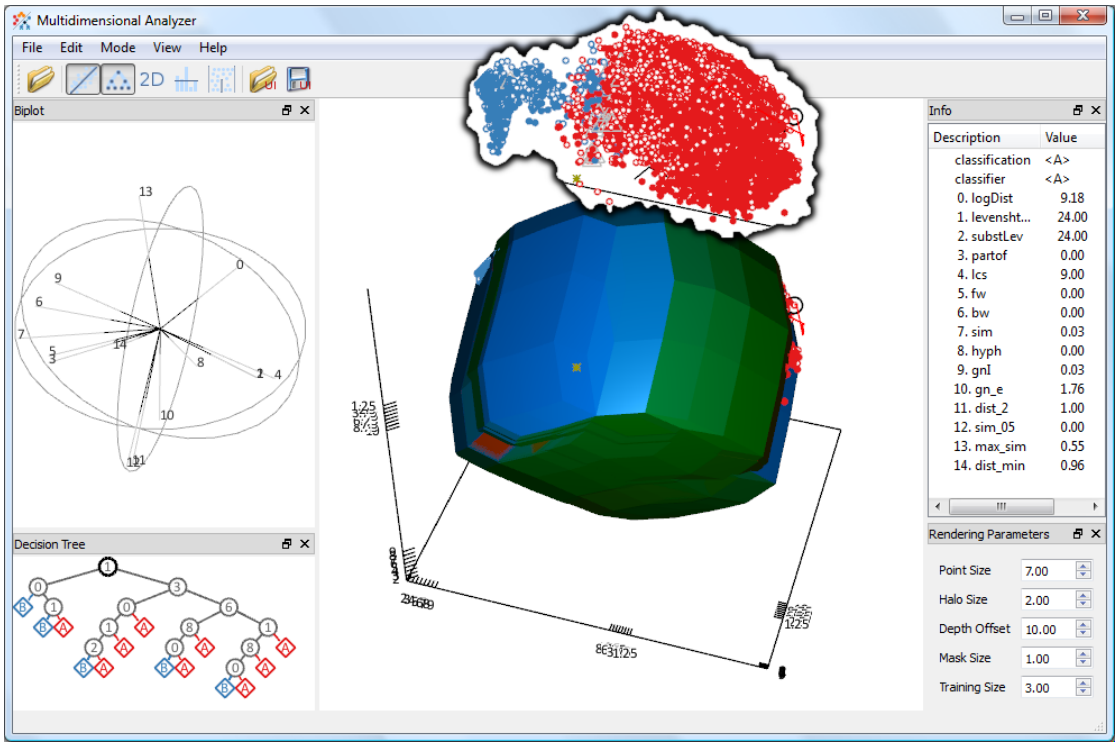


Figure 5.17: Data set visualized with Multidimensional Analyzer. Overlay added to illustrate the large number of data items inside the decision tree hyperplanes.

5.2.5 Evaluation

This section presents the evaluation of the presented approach. The introduced visual tools make the development of a classifier more convenient, due to the possibility of fast and simple data analysis, annotation assistance, and the aid in the feature design. The described visual analytics tool was developed to meet the requirements for the design of an effective classifier and greatly helped in achieving the following results, which represent a large improvement in the matching process.

Table 5.2: F-Scores ($\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision+recall}}$) for classifiers, with precision: $\frac{\text{TP}}{\text{TP+FP}}$ and recall: $\frac{\text{TP}}{\text{TP+FN}}$.

evaluation	training	SIMPLE	ALL	BEST	ITERATIVE
BW	BW	52.3%	91.0%	93.0%	91.8%
SE	BW	51.1%	86.0%	86.4%	86.4%
SE	SE	72.1%	88.3%	86.7%	90.9%
W	**BW**	**34.6%**	**77.8%**	**81.3%**	**82.4%**
W	W	62.0%	80.0%	85.4%	88.4%
NE	BW	55.0%	69.6%	66.7%	82.6%
NE	NE	48.0%	82.9%	82.0%	89.1%

Table 5.2 presents the result of the final evaluation of the developed classifier including different feature sets. The first row contains the results for the development set and unsurprisingly, methods which were optimized during the development get the highest results. The following rows are of more interest. Each region is evaluated twice, first for a classifier that is trained on the development training set and second with a classifier that is trained on a close by region that has probably the same characteristics. To go into more detail let us pick the bold marked row for the West (W) region that was evaluated by a classifier trained on the development set (BW). The first result column shows the result for the simple feature set that can be seen as a baseline result. The results in the other columns benefit from the previously described visual-aided methods to get more sophisticated feature sets. In the *ALL* case all revealed features are used in one feature set. The *ALL* feature set is further outperformed by the *BEST* feature set that was identified by selecting all possible subsets of the 15 features as feature sets and evaluating them on the development set (BW). The *BEST* combination contains 5 features: *logDist*, *levenshtein*, *substLev*, *partof*, and *hyph*. The last column lists the results for the *ITERATIVE* classifier, which performs best in all cases. A further fact can be derived from the results: the

advanced classifiers are less dependent on the similarity of the training and evaluation sets, compared to the *SIMPLE* classifier.

The analysis of large data sets, especially by the scatter plot shown in Figure 5.12, benefits from large high resolution displays. The linked view technique also requires large space, and the ability to extend the scatter plot to the whole screen was very helpful for the exploration of the data.

Usually the NLP community uses only lists and tables to analyze data. For the geospatial matching task these are not sufficient because the local interaction can hardly be represented without drawing the instances in 2D space. Also the visual density analysis assists the finding and definition of features. At last the consideration of 3D scatter plots showed obvious annotation errors that are not visible by standard NLP methods.

CHAPTER 6 ANAGLYPH STEREO WITHOUT GHOSTING

In recent years 3D technology has emerged around us very fast; now everybody can view the newest 3D movies in theaters, buy 3D televisions and 3D consumer cameras. Back in Chapter 2 several depth cues have been introduced, all of these contribute to depth perception or 3D shape perception. What is so special about the new 3D technology? All previews techniques were showing 2D motion pictures. These could capture all monocular static and dynamic depth cues. To present binocular cues it is necessary to present different images to the two eyes, this is what is new 3D technology.

However, technology for stereopsis is not new: in 1838 Wheatstone [Whe38] first described stereopsis together with the stereoscope, a device to view side-by-side images. Nowadays there are several ways to use stereoscopy to provide 3D depth perception for images, videos, or computer graphics: side-by-side viewing, anaglyphs, shutter glasses, polarized light, Infitec glasses, and autostereoscopic displays. Anaglyphs use colored filters to separate the left and the right image. Anaglyph stereo was first described by Rollmann [Rol53] in 1853. Nowadays red-cyan glasses are most common, which can be purchased for just a few cents. Therefore, anaglyph stereo provides a low-budget solution to view stereoscopic images. However, it may suffer from ghosting, bad color reproduction, and retinal rivalry. Even small amounts of ghosting noticeably degrade image quality [TWA11]. Here this issue is addressed. Ghosting results from crosstalk, this refers to the fact that a color channel of the image that should be filtered out for an eye passes the filter partially. Crosstalk comes from the imperfection of the anaglyph filters and the display device and cannot be eliminated by an algorithm. However, this chapter presents a technique to reduce ghosting substantially, so that it becomes negligible.

Although stereoscopic rendering of 3D scatter plots makes sense [Yan99], the presented technique is not limited to 3D scatter plot renderings and can even be used to remove ghosting from stereoscopic or anaglyph photos. Therefore this chapter first presents the technique itself and then the integration into the Multidimensional Analyzer.

The technique aims to perceptually calibrate an anaglyph stereoscopic system and to use the calibration to eliminate ghosting from the anaglyph image. The technique was

presented in a similar form in [SW11].

The idea is to match the luminance of images perceived through the glasses to the value perceived without the other image. One goal is to rely neither on data for the emission spectra of displays nor transmission spectra of filters, since these can seldom be obtained without measurements, nor on a properly calibrated monitor. Instead the target systems are low-cost environments and, therefore, the relevant data should be acquired through simple and easy-to-use perceptual measurements that come at no extra cost. First this chapter will present how to measure perceived luminance through anaglyph glasses and build a model based on luminance perception by the left and right eyes through the anaglyph glasses. Next it will show how to use the model to compensate for crosstalk and eliminate the aspect of ghosting caused by luminance. The technique can render anaglyphs, similar to the anaglyphs known as full color (Photoshop algorithm), half color (modified Photoshop method), and grayscale, but with removed ghosting artifacts.

After the discussion of related work, the model for luminance perception is presented and it is shown how to derive the parameters by just a few measurements. Section 6.3 shows how ghosting can be corrected given the model parameters. Section 6.4 generalizes the model to arbitrary filter colors. Section 6.5 illustrates the results of the correction process using two idealized models for the sources of crosstalk. Section 6.6 analyzes a combination of monitor and glasses based on transmission and emission spectra and compares the result to the idealized models. Section 6.7 describes how luminance ghosting can be eliminated even for high contrast input images. Finally Section 6.7 considers the application of anaglyph stereo rendering for 3D scatter plots.

6.1 Related Work

Dubois [Dub01] considers the spectral distribution of the display colors and the transmission of the filters to calculate anaglyphs based on minimizing projection error between the original stereo pair and the anaglyph seen. This is closely related to the method introduced in this chapter; however, the novel method presented here does not need the spectral distributions and the transmission curves but derives anaglyphs based on few measurements capturing the relevant interrelations. Dubois uses a weighted square error in CIE XYZ space, which is not perceptually uniform. A perceptually uniform space would be more appropriate, but these spaces are non-linear. Hence, a non-linear transformation would be required. In contrast to this approach, here luminance is in focus, which is most important to perception of high spatial frequencies [Mul85]. Additionally, the method presented here does not try to minimize the error between the original stereo image and the final anaglyph but the error between the stereo image seen through the filter and the final anaglyph.

Winkler et al. [WvdBLK01] give a concise introduction to the encoding of luminance and chromaticity in video processing. The theory of opponent colors states that humans

process the responses of the S, L, and M cones to a luminance channel, a red-green and a blue-yellow channel. The accuracy of the chromatic channels is significantly lower than of the luminance channel. Therefore, in video processing often the chromatic channels are compressed stronger than the luminance channel. This technique is called chroma subsampling. In JPEG encoding often a factor of 4 is used.

Sanders and McAllister [SM03] compare three stereo generation approaches: the Photoshop algorithm, the algorithm proposed by Dubois [Dub01], and the midpoint algorithm operating in CIE L*a*b* space. They put a threshold on the value of the red channel of the anaglyph and claim that an algorithm that produces a red channel value above the threshold will cause ghosting. This is a heuristic approach that has little validity without considering the colors in the two images.

Ideses and Yaroslavsky [IY05] point out that anaglyphs are one of the most economical methods for stereoscopic presentation and propose several methods to reduce ghosting in anaglyph stereo images. These rely on stereo image registration, defocusing, and a non-linear operation on depth maps. Ghosting can be eliminated by registering the images and bringing them to alignment. Since accommodation is related to convergence [OUW+06, Uka06], aligning the images is beneficial. However, only one depth plane can be aligned at once, which limits the utility of this technique. Ideses and Yaroslavsky [IY05] also show how blurring the image color components can reduce ghosting effects in anaglyphs. Lobel [Lob09] carries this to extremes by presenting magenta-cyan anaglyphs, for which both eyes receive the blue channel that is blurred to reduce ghosting. However, Lobel reports ghosting as a limitation of his approach; the ghosting elimination technique presented here could be used to solve this issue.

Some methods rely on explicit knowledge of the spectral distributions of the display device and the transmission functions of the filters, which is not required for the method presented here. Such a technique was introduced by McAllister et al. [MZS10] and compared to different other techniques. Similarly, Sorensen et al. [SHS04] designed a special amber-blue filter pair, known as ColorCode 3-D, with the goal to separate the color information and the depth information to the two eyes.

Woods et al. [WYK07] discuss crosstalk in anaglyph stereoscopic images and also explain crosstalk, the source of ghosting [WT02], for different stereoscopic displays [Woo10]. According to Woods et al., three factors play an important role when looking at crosstalk in anaglyph stereo: the spectral quality of the display, the spectral quality of the glasses, and the quality of the anaglyph image generation matrix. Woods and Harris [WH10] compare the crosstalk of different anaglyph glasses on different displays and recommend good matches for a given monitor to minimize crosstalk. Using the "right" anaglyphs for a given monitor is much less important with the novel technique introduced here. If it happens that you have the "right" glasses for your monitor, you will simply not notice any difference using the novel technique, since there is no need to compensate for non-existent ghosting. However, if you wear the "wrong" glasses, the

novel technique can significantly reduce ghosting. Bloos [Blo08] designed a test pattern for ghosting determination. In contrast to the calibration method presented here, he uses a static image to determine two measurement values for both eyes; these do not serve a calibration purpose but are a quality measure to enable the comparison of two stereo renderings.

There are several software applications for the generation of anaglyph stereo images. One typical example is StereoPhoto Maker [SS10], which allows the user to generate anaglyphs for different filters by using a variety of algorithms and also by providing a custom matrix.

6.2 Determining Filter Parameters

Tuples are used instead of column vectors throughout the chapter:

$$(a,b,c) = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

6.2.1 Model

The model should capture how to account the different color channels for the received luminance through a filter of some anaglyph glasses. With the model it should be possible to answer questions of the following kind: how much of the luminance coming from the green channel is leaking through the red filter?

First, let us consider the following model:

$$a_\mathrm{l} x_\mathrm{lr}^\gamma + b_\mathrm{l} x_\mathrm{lg}^\gamma + (1 - a_\mathrm{l} - b_\mathrm{l}) x_\mathrm{lb}^\gamma = Y$$

Here, x_lr, x_lg, and x_lb denote the non-linear RGB values, ranging from 0 to 1, Y is the (normalized) luminance and a_l, b_l, and γ are the model parameters. If measurements of Y for varying RGB values could be obtained, the model could be fitted by least squares fitting on Y, obtaining the model parameters. Unfortunately, luminance cannot be measured without additional hardware. However, the model can be modified instead:

$$a_\mathrm{l} x_\mathrm{lr}^\gamma + b_\mathrm{l} x_\mathrm{lg}^\gamma + (1 - a_\mathrm{l} - b_\mathrm{l}) x_\mathrm{lb}^\gamma = y^\gamma \tag{6.1}$$

Here y denotes a gray value (red, green, and blue have identical values); therefore, the model parameters can be obtained without requiring any additional hardware. So far, the model is designed for the luminance arriving at one eye. An identical model could be built for the other eye's filter. However, since γ is identical for the two eyes, the two can

also be combined into one model:

$$(a_{\mathrm{l}}x_{\mathrm{lr}}^{\gamma} + b_{\mathrm{l}}x_{\mathrm{lg}}^{\gamma} + (1 - a_{\mathrm{l}} - b_{\mathrm{l}})x_{\mathrm{lb}}^{\gamma} + a_{\mathrm{r}}x_{\mathrm{rr}}^{\gamma} + b_{\mathrm{r}}x_{\mathrm{rg}}^{\gamma} + (1 - a_{\mathrm{r}} - b_{\mathrm{r}})x_{\mathrm{rb}}^{\gamma})^{1/\gamma} = y$$

The (leading) subscripts l and r refer to quantities related to the left and right eyes, respectively. When measuring for the left eye, the following assignment is made $x_{\mathrm{rr}} = x_{\mathrm{rg}} = x_{\mathrm{rb}} = 0$, resulting in Eq. (6.1) for the single-eye model. When measuring for the right eye, the left eye's values are set to zero. y is always used for the measured gray value.

6.2.2 Measurement Process

To calculate the luminance parameters measurements were conducted on the left and right eyes while wearing the anaglyph glasses. The user had to adjust the luminance of red, green, blue, cyan, magenta, and yellow to match a gray value by face-based luminance matching [KRC02]. This process was performed not just for one but for 20 gray values, resulting in 120 measurement points for each eye.

6.2.3 Model Evaluation

The sRGB standard uses a slightly different model for gamma than presented here. The sRGB color system uses a piecewise function with a linear and a power function part. However, since modern LCD monitors often have a gamma substantially different from the one in sRGB, the model presented here allows for better fitting of measurements. Figure 6.1 shows the differences between the measured data values and the fitted model for the red-cyan glasses (the model parameters will be presented in Section 6.5.2). The estimated population standard deviation is 0.006 60=1.68/255. This low variance indicates that the model is valid. The model was tested with different persons, even with one red-green blind person; each of them had very similar calibration results. It was also tested with different LCD monitors and a CRT monitor and obtained low variances.

6.2.4 Reduced Calibration

Since it is time-consuming to take 240 samples, the number of samples was reduced to 3 for each eye for practical calibration purposes. The following heuristics for good sample positions has been developed. Since the intensity of colors is adjusted to match that of a gray sample, colors should be chosen that are far away from the main diagonal of the color cube, where the gray colors are. At the same time, colors should be far away from each other. Therefore, pure red, green, and blue are good sample positions for both eyes.

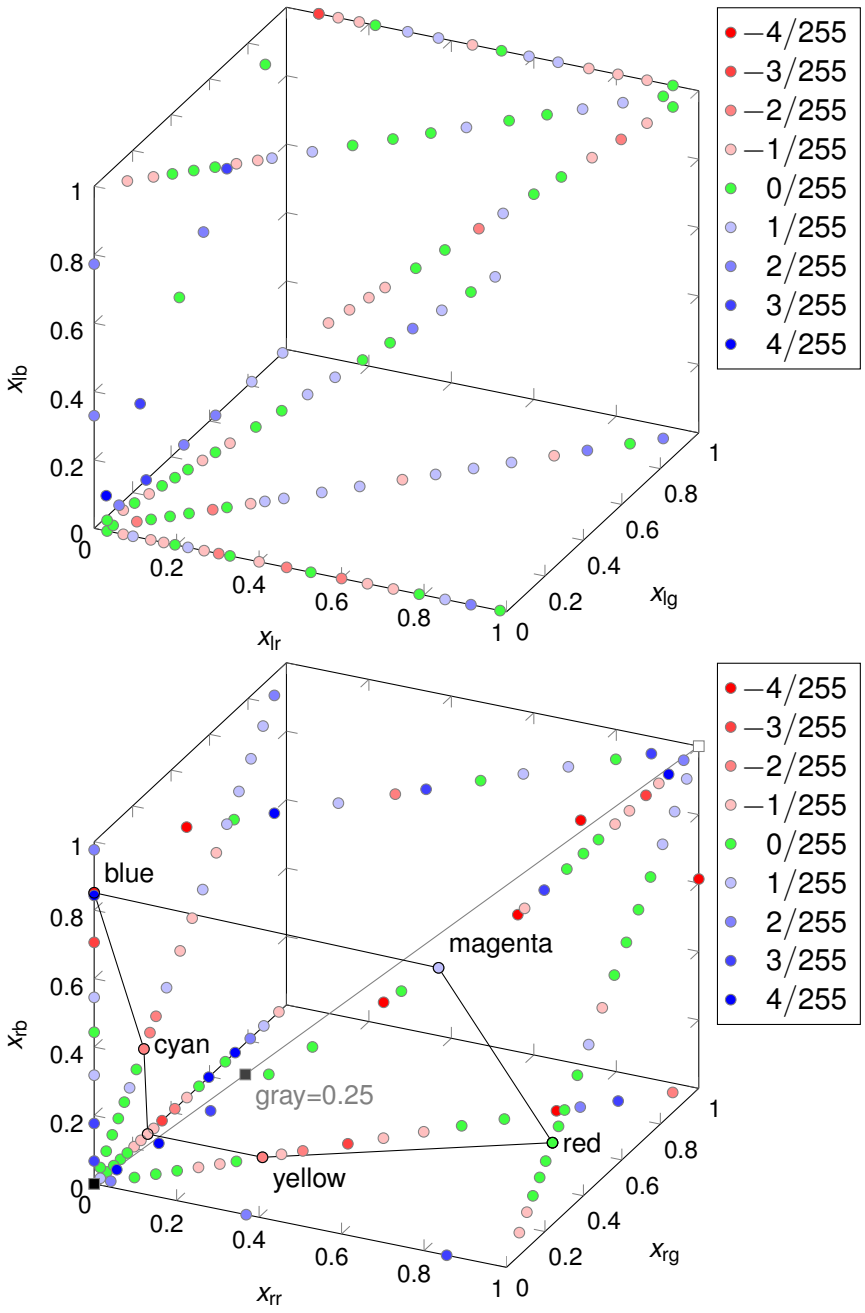


Figure 6.1: Difference between model and measured values for the left (red) and right (cyan) eyes, in the top and bottom image, respectively. The axes are scaled between 0 and 1. The bottom image additionally illustrates the color values whose luminance is perceived equal to gray=0.25.

Having few sample positions is more likely to cause problems with the fitting algorithm, which requires the first-order derivatives of the model function with respect to the model parameters. These derivatives have terms x^β with fractional β (γ and $1/\gamma$) and—due to inaccuracies—negative x. Such functions cannot be evaluated since x^β is undefined, causing the fitting algorithm to stuck. A feasible solution is to extend the function C^0 continuously at 0 to negative values by a linear function with the slope of $\frac{\mathrm{d}x^\beta}{\mathrm{d}x}(1) = \beta$.

6.3 Correction of Ghosting

The main idea behind the ghosting correction technique is that the anaglyph image should be identically perceived through the glasses as if the display showed the original images.

If we had ideal red-cyan glasses together with a corresponding display device, the red filter would entirely filter out the green and blue components for the left eye and the cyan filter would entirely filter out the red component for the right eye and the desiderata could be fulfilled by just taking the red component from the left eye image and the green and blue components from the right eye image.

Since neither anaglyph filters nor displays are ideal, there always is some leakage from the one image to the other one, which may cause ghosting. The color difference between the desired and the perceived image can be described by the luminance and the chromaticity difference. Luminance differences result in much more distracting ghosting than chromaticity differences. Therefore, the main goal is to eliminate luminance differences. There are three degrees of freedom when rendering the anaglyph image, two of them are fixed due to luminance. Thus, one degree of freedom remains that will be used to minimize the most significant chromaticity difference.

6.3.1 Anaglyphs Without Ghosting

This section presents the novel color anaglyph generation technique that does not produce luminance ghosting. Each pixel is processed independently. The inputs are RGB colors for the left $(R'_\mathrm{l}, G'_\mathrm{l}, B'_\mathrm{l})$ and right $(R'_\mathrm{r}, G'_\mathrm{r}, B'_\mathrm{r})$ eyes, and an RGB anaglyph color (R', G', B') that does not exhibit luminance ghosting is produced.

First the non-linear R′G′B′ colors are transformed to linear RGB space by using the γ value measured:

$$C = C'^{\gamma}$$

where $C \in \{R_\mathrm{l}, G_\mathrm{l}, B_\mathrm{l}, R_\mathrm{r}, G_\mathrm{r}, B_\mathrm{r}\}$ and $C' \in \{R'_\mathrm{l}, G'_\mathrm{l}, B'_\mathrm{l}, R'_\mathrm{r}, G'_\mathrm{r}, B'_\mathrm{r}\}$.

For both eyes, luminance is exactly reproduced. Additionally, one chromaticity channel is reproduced for one eye. Figure 6.2 shows how the general approach can be described by two transformations $\boldsymbol{P}$ and $\boldsymbol{R}$; this section will present the derivation of $\boldsymbol{R}$.

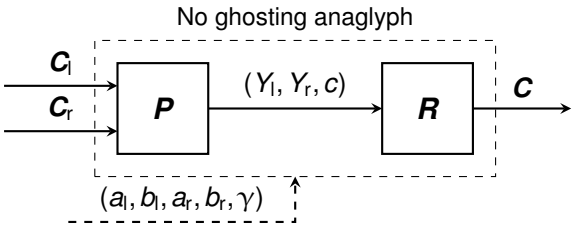


Figure 6.2: First the luminance values for the input images (Y_l and Y_r) and a chromaticity value (c) are calculated by the transformation $\boldsymbol{P}$. Next an output color is calculated that has the desired luminance values and the desired chromaticity value by the transformation $\boldsymbol{R}$.

Luminance can be calculated, using the calibrated values that exactly capture the luminance through the three color channels, by:

$$Y_{\mathrm{EYE}}((R,G,B)) = a_{\mathrm{EYE}}R + b_{\mathrm{EYE}}G + c_{\mathrm{EYE}}B$$

where $c_{\mathrm{EYE}} = 1 - a_{\mathrm{EYE}} - b_{\mathrm{EYE}}$.

Some of the following considerations are specific to red-cyan filter pairs; the obtained solution is generalized for arbitrary filters in Section 6.4. Because the light passing the red filter is largely monochromatic (red), the cyan filter on the right eye is responsible for the color impression. Light passing this filter is dominated by the green and blue channels. Neglecting the red component entirely, green-blue defines the most prominent chromaticity channel one can perceive. Therefore the chromaticity channel is set to $c = G - B$. Taking the red component into account, an even better estimate for c could be derived using chromatic measurements. However, green-blue is a very good approximation and the effort of additional measurements would not pay off in most situations. Hereby, the linear transform that maps a color vector to a (luminance left, luminance right, chromaticity) vector (YYc space) is obtained by:

$$\boldsymbol{R}^{-1} = \begin{bmatrix} z_\mathrm{r} & z_\mathrm{g} & z_\mathrm{b} \\ y_\mathrm{r} & y_\mathrm{g} & y_\mathrm{b} \\ 0 & 1 & -1 \end{bmatrix}$$

where $y_\mathrm{r} = a_\mathrm{r}$, $y_\mathrm{g} = b_\mathrm{r}$, $y_\mathrm{b} = 1 - a_\mathrm{r} - b_\mathrm{r}$, $z_\mathrm{r} = a_\mathrm{l}$, $z_\mathrm{g} = b_\mathrm{l}$, and $z_\mathrm{b} = 1 - a_\mathrm{l} - b_\mathrm{l}$. Thus the inverse of $\boldsymbol{R}^{-1}$, which is $\boldsymbol{R}$, should be taken to calculate no ghosting anaglyphs.

$$\boldsymbol{R} = \frac{1}{z_\mathrm{r} - y_\mathrm{r}} \begin{bmatrix} y_\mathrm{g} + y_\mathrm{b} & -z_\mathrm{g} - z_\mathrm{b} & y_\mathrm{g}z_\mathrm{b} - y_\mathrm{b}z_\mathrm{g} \\ -y_\mathrm{r} & z_\mathrm{r} & y_\mathrm{b}z_\mathrm{r} - y_\mathrm{r}z_\mathrm{b} \\ -y_\mathrm{r} & z_\mathrm{r} & y_\mathrm{r}z_\mathrm{g} - y_\mathrm{g}z_\mathrm{r} \end{bmatrix}$$

The transformation from YYc space to RGB space is illustrated in Figure 6.3. In the following, full color, half color, and gray variants will be presented which only differ in $\boldsymbol{P}$ but have the same $\boldsymbol{R}$.

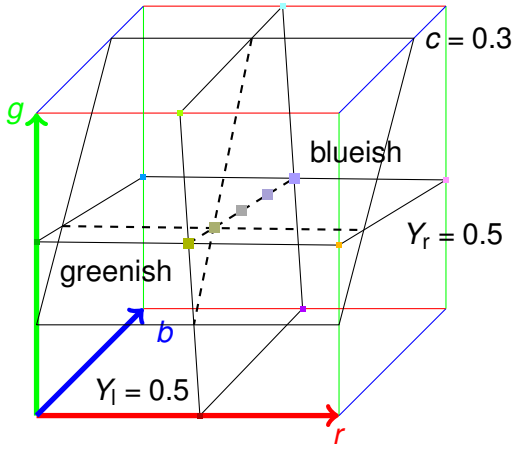


Figure 6.3: Points in linear RGB space, where luminance for the left eye (Y_l)=0.5 and luminance for the right eye (Y_r)=0.5 and chroma (c)=0.3 lie on planes. The plane normals are the row entries of $\boldsymbol{R}^{-1}$. The figure illustrates the red-cyan glasses that are measured in Section 6.5.2.

6.3.2 Full Color Anaglyphs

In the same manner as $\boldsymbol{R}^{-1}$ maps a color to YYc space, $\boldsymbol{P}$ needs to map two colors to YYc space. Y_l has to be calculated using the color of the left, Y_r and c using the color of the right images. The transformation that maps the input colors for the images can be described by a linear transformation. The input is the vector $(R_l, G_l, B_l, R_r, G_r, B_r)$. Hence $\boldsymbol{P}$ can be represented by the following 3×6 matrix:

$$\boldsymbol{P} = \begin{bmatrix} z_r & z_g & z_b & 0 & 0 & 0 \\ 0 & 0 & 0 & y_r & y_g & y_b \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Nice Properties

The so defined full color anaglyphs have some nice properties. First, in case the input for the left and right eyes is identical ($\boldsymbol{C}_l = \boldsymbol{C}_r$), the anaglyph will just reproduce the input color ($\boldsymbol{C} = \boldsymbol{C}_l$). Second, the luminance for both eyes is preserved. The last property is described for red-cyan glasses: in case the green and blue components are identical in the input image for the right eye (cyan filter), they will be identical in the anaglyph image too. From these properties, one could derive some equations and obtain the same overall transformation ($\boldsymbol{RP}$) like presented above. However, the introduction of the YYc space helps obtain an intuitive understanding of the transformation, which also holds for the half color and gray anaglyphs presented below.

6.3.3 Half Color Anaglyphs

Full color anaglyphs cause binocular rivalry (aka retinal rivalry) for some people. Half color anaglyphs are designed to reduce binocular rivalry by taking the gray value instead of the red component of the left image as red component for the anaglyph image. Considering the technique, the individual luminance calculation for the left and right images can be replaced by a calculation that treats the left and right image channels identically. A reasonable calculation can be based on the luminance from the RGB calculation of the sRGB model, given by $Y((R,G,B)) = \boldsymbol{t}_{\mathrm{Y}}^{\mathrm{sRGB}}(R,G,B)$ with the row vector $\boldsymbol{t}_{\mathrm{Y}}^{\mathrm{sRGB}} = [x_{\mathrm{r}}\ \ x_{\mathrm{g}}\ \ x_{\mathrm{b}}] \approx [0.2126\ \ 0.7152\ \ 0.0722]$. Hence $\boldsymbol{P}$ can be represented by:

$$\boldsymbol{P} = \begin{bmatrix} x_{\mathrm{r}} & x_{\mathrm{g}} & x_{\mathrm{b}} & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{\mathrm{r}} & x_{\mathrm{g}} & x_{\mathrm{b}} \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

6.3.4 Gray Anaglyphs

For gray anaglyphs, the green and blue components need to be identical, which is the case when c is zero. Hence $\boldsymbol{P}$ is given by:

$$\boldsymbol{P} = \begin{bmatrix} x_{\mathrm{r}} & x_{\mathrm{g}} & x_{\mathrm{b}} & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{\mathrm{r}} & x_{\mathrm{g}} & x_{\mathrm{b}} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

6.3.5 Removing Ghosting from Anaglyphs Generated by Naïve Methods

Now a method will be presented to correct ghosting in anaglyph images generated by naïve methods, see Figure 6.4. The naïve methods use the red component (full color) or the luma (half color and gray) of the left image and the green and blue components (full and half color) of the right image, which are assigned to the red, green, and blue components of the anaglyph image. The green and blue components are replaced by the luma of the right image in the gray anaglyph case. The red component of the anaglyph image is interpreted as luminance of the left image, since this is the only information available from the left image; therefore, it is the best estimation of luminance. To estimate the luminance of the right image we use the green and blue components weighted by the filter values, which need to be normalized to account for the unknown red component. Hence $\boldsymbol{P}$ is given by:

$$\boldsymbol{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{y_{\mathrm{g}}}{y_{\mathrm{g}}+y_{\mathrm{b}}} & \frac{y_{\mathrm{b}}}{y_{\mathrm{g}}+y_{\mathrm{b}}} \\ 0 & 1 & -1 \end{bmatrix}$$

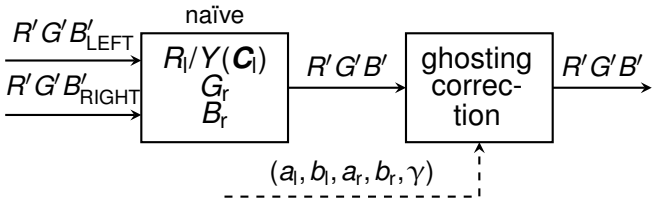


Figure 6.4: First the images for the left and right eyes are composed by the naïve approach to an anaglyph stereo image, by selecting the red channel or the luma of the left and the green and blue channels of the right image. The naïve anaglyph image is taken and ghosting is eliminated by using the filter values obtained in Section 6.2.

6.4 Filter Color Estimation

The novel technique is not limited to red-cyan glasses. The only thing that needs to be changed to make the technique work for other filters is to decide which chromaticity channel should be preserved. This is the *green-blue channel* of the *right eye* in the red-cyan case. In the general case, the decision can be made based on the filters.

The vector $(a_{\text{EYE}}, b_{\text{EYE}}, 1-(a_{\text{EYE}}+b_{\text{EYE}}))$ actually is the luminance from the red, green, and blue color channels, respectively, as perceived through the filters. From the sRGB model, the linear RGB to luminance transformation $Y((R,G,B)) = \boldsymbol{t}_{\text{Y}}^{\text{sRGB}}(R,G,B)$ is used. Since there is a filter between the display and the eyes of the user, not all of the luminance arrives: $(a_{\text{EYE}}, b_{\text{EYE}}, 1-(a_{\text{EYE}}+b_{\text{EYE}})) = f_{\text{EYE}}\boldsymbol{f}_{\text{EYE}} \circ [0.2126 \;\; 0.7152 \;\; 0.0722]^T$, where $\circ$ denotes the entry-wise vector product and f_{EYE} is used to normalize $\boldsymbol{f}_{\text{EYE}}$ in the way that the largest component is 1. Given this relationship, the filter values $\boldsymbol{f}_{\text{EYE}}$ can be calculated.

Anaglyph filters are designed in a way that one from the filter pair filters out all but one color channel (this is the red filter for the red-cyan filter pair, which filters out all but the red channel); the other two color channels of the other eye are chosen to be preserved. Algorithmically, the decision which eye is chosen is made based on the second largest filter component. There is no need to derive new equations but simply interchange the input channels when calculating the correction matrix.

6.5 Results

An application was developed using C++, OpenGL, and Qt, which can be obtained from the website `www.vis.uni-stuttgart.de` (in "Research"). GLSL is used for rendering the anaglyph image from the left and right images. The 3×6 transformation matrix ($\boldsymbol{RP}$) can be decomposed into a dot product of a 3-component vector from one image and a 3×4 matrix multiplication taking the result of the dot product and the other source image, which can be efficiently evaluated in the shader.

Next measured ColorCode 3-D and red-cyan paper-frame glasses are presented for a Dell U2410 monitor, with the Standard Preset Mode.

6.5.1 ColorCode 3-D (Amber-Blue)

For the ColorCode 3-D glasses, the following results were obtained: $a_\mathrm{l} = 0.4581$, $b_\mathrm{l} = 0.5183$, $a_\mathrm{r} = 0.0076$, $b_\mathrm{r} = 0.0552$, $\gamma = 1.856$. Using this, one obtains $\boldsymbol{f}_\mathrm{l} = (1, 0.336, 0.152)$ and $\boldsymbol{f}_\mathrm{r} = (0.003, 0.006, 1)$. This means, the left eye glasses let pass the entire red component, 34% of the green, and 15% of the blue component; and the right eye filter lets pass the blue component, 0.3% of the red, and 0.6% of the green component. This is remarkable, since on the left eye one can potentially distinguish all colors and only have a limited luminance leaking consisting of 2.4% $(= 1 - (a_\mathrm{l} + b_\mathrm{l}))$, which causes ghosting. On the right eye, there is 6.3% luminance leaking. With the presented technique, the luminance leakages can be eliminated, but it is not possible to eliminate the color leakage of the blue component to the left eye.

6.5.2 Red-Cyan Filter

For the red-cyan glasses, the following results were obtained: $a_\mathrm{l} = 0.9260$, $b_\mathrm{l} = 0.0612$, $a_\mathrm{r} = 0.0094$, $b_\mathrm{r} = 0.8705$, $\gamma = 1.668$; $\boldsymbol{f}_\mathrm{l} = (1, 0.020, 0.041)$; $\boldsymbol{f}_\mathrm{r} = (0.027, 0.732, 1)$. 2% of the green and 4% of the blue channel are not enough to see green or blue on the left eye. The 3% red on the right eye is also not apparent. Considering the ghosting, there is 7% for the left eye's luminance but just 1% for the right eye's luminance, which means without correction there is strong ghosting on the left eye.

6.5.3 Illustration

The previous section presented measurements of how much luminance is perceived from the primaries through the filters, but the chromaticity perceived (i.e., as which color this luminance is perceived) was not considered. Given the filter values, the color of an image seen through the glasses can be estimated. However, without having further measurement data capturing color perception, but just luminance measurements, some assumptions need to be made.

When assuming that the filter only affects the luminance but not the color hue and saturation, one obtains the images shown in the left columns of Figure 6.5, which are calculated by simply multiplying the RGB values with the filter transmission values. This behavior would be obtained by monochromatic primaries.

Monitor primaries usually are not monochromatic but extended and overlap partially [BDM04]. Let us assume that having anaglyph filters that clearly separate. In such a case, light from the red, green, and blue channels passing the red filter appears red, since the blue and green frequencies are totally removed by the ideal filter. Light

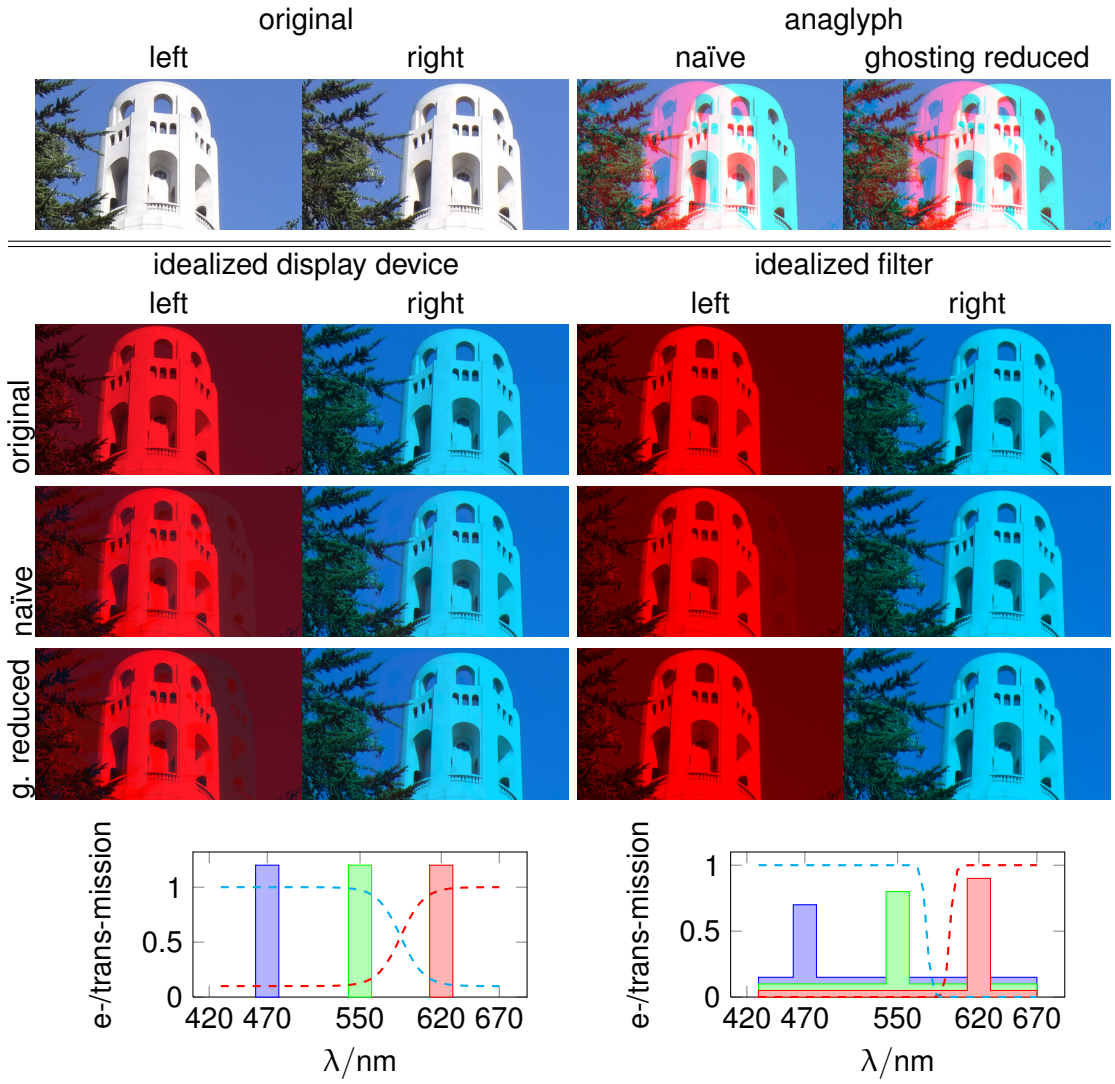


Figure 6.5: The diagrams illustrate the transmission spectra of the red-cyan filters as dashed lines, and the emission spectra of the monitor primaries as shaded areas. The left diagram uses an idealization with respect to the display device in the sense that the spectra are narrow; however the red-cyan filters do not separate well. The right diagram uses an idealization with respect to the filter, in the sense that the filter has a clear separation; however the display spectra overlap. From top to bottom, there are the original left and right images and the naïve and ghosting reduced anaglyphs in the first row; the (non-anaglyph) images as they look through red-cyan glasses (second row), the naïve anaglyph images as they look through the glasses (third row), the ghosting reduced anaglyph images as they look through the glasses (forth row). (Colors may be strongly distorted in printouts.)

from the red, green, and blue channels passing the cyan filter does not appear red. This means, the luminance can be collected from the green and blue color channels passing through the red filter and replace it with a luminance that is added to the red channel; and the same way the luminance from the red channel passing through the cyan filter is distributed to the green and blue channels. Contrary to the previous assumptions, which represent the worst case for the ghosting removal technique, this model leads to the best case. The resulting images are shown in the right columns of Figure 6.5.

The first thing to note is that there are no visible differences between the cyan images; this could be expected, since just 1% of the luminance from the red channel leaks through the cyan filter. However, there is visible ghosting in the anaglyph image generated by the naïve method as seen through the red filter in both cases. This ghosting is mainly caused by the luminance differences to the original image as seen through the glasses. With the novel technique, the luminance differences are completely eliminated (if the dynamic range is sufficient see Section 6.7). In the first case, there may remain differences in color hue and saturation—depending on your display the compensation shown in the figure may be too large or to small, resulting in remaining luminance differences—but in the second case, ghosting can be entirely eliminated. The chromatic differences are direct consequences of the model assumptions. If you take the first model which assumes that the filter is imperfect, colors are still distinct (span a 3D vector space) after passing the filter; therefore, mixing primaries with different proportions, to obtain a desired luminance, results in different hues and/or saturations. With the second model, colors span a 1D vector space on the red eye and a 2D vector space on the cyan eye resulting in complete ghosting elimination.

6.6 Analysis

In the previous section, some assumptions were made on the filter transmission and the monitor primaries spectra leading to two different estimates. The second one (idealized filter) obtained very good results, while the first one (idealized device) suffered from quite strong color leakage. Here, an analysis of a real system will be presented to identify which of the two models comes closer to the real situation.

To simulate the colors the observer sees through the filters, the power spectrum of a monitor together with a pair of red-cyan anaglyph glasses is analyzed. CRT monitors have quite similar spectra [WT02]—which differ from LCD spectra [Sha02]. Here the monitor spectra of the right Sony Trinitron Multiscan GDM-F500 monitor presented by Boyaci et al. [BDM04] and the transmission spectra of the paper-frame glasses (perspektrum.de) presented by Wieser [Wie06] were used for the calculations. Using the CIE 1931 2° Standard Observer color matching functions, the gamut and white point of the GDM-F500 can be calculated, see Figure 6.6. The cyan and red gamuts are calculated the same way, but with multiplying the monitor spectra with the filters

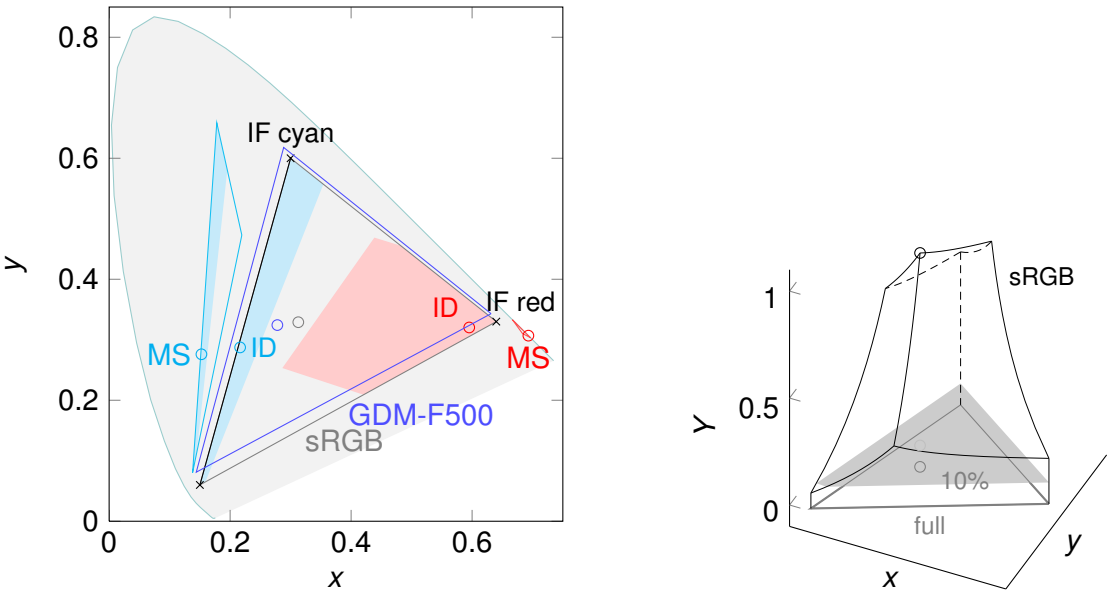


Figure 6.6: Gamuts (as triangles) and white points (as circles) of the idealized display model (ID), the idealized filter model (IF), and the spectral measurement (MS) in the CIExy coordinate system. Gamut parts reaching 10% of the maximal possible luminance are drawn as shaded areas, illustrated in the right figure. Red and cyan colored gamuts correspond to red and cyan filters, respectively. In blue, the gamut and white point of the Sony Trinitron Multiscan GDM-F500 are shown; the ID gamuts are identical to the sRGB gamut (in gray) at $Y \approx 0$.

transmission spectra beforehand. Both the red and the cyan gamuts lie completely outside the GDM-F500 and the sRGB gamuts, and can therefore not be reproduced on these monitors. Color gamuts are 3D structures, which can be presented in xyY space. At $Y \approx 0$ they are defined by the primaries. As luminance increases, the gamut shrinks and collapses to the white point at $Y = 1$. Gamuts are usually presented as projection to the xy plane, which corresponds to the $Y \approx 0$ gamut. At $Y \approx 0$, the gamuts used by the idealized display device model correspond to the full monitor gamut. As additional information gamut parts reaching 10% of the maximal luminance are shown as shaded areas in Figure 6.6.

Notice that for the red filter the spectral measurement's gamut significantly differs from the idealized display device model's gamut but comes close to the idealized filter model's gamut. This means, the idealized filter model is more adequate and low color leakage should be expected in the real world.

6.7 Dynamic Range

With imperfect filters, luminance intended for one eye reaches the other eye and vice versa. This implies that not each combination of luminance for the two eyes can be achieved, as illustrated in Figure 6.7. The gamut directly corresponds to the measured filter values for the red-cyan glasses (see Section 6.5.2).

After applying the ghosting correction matrix, resulting colors do not necessarily lie within $[0,1]$. Concrete example: the left input image is dark $\boldsymbol{C}_\mathrm{l} = (0.02, 0.02, 0.02)$ and the right one is bright $\boldsymbol{C}_\mathrm{r} = (0.9, 0.9, 0.9)$, resulting in point $\boldsymbol{A}$. The anaglyph color after ghosting correction would be $\boldsymbol{C} = (-0.051, 0.909, 0.909)$. Unfortunately, the red component needs to be reduced to negative values to compensate for the luminance coming from the green and blue channels of the output image shining through the red filter. Next two solutions are presented that bring the colors to the $[0,1]$ interval.

6.7.1 Clamp to Range

The easiest way to deal with the problem is to clamp the output color values to $[0,1]$, resulting in (0, 0.909, 0.909) for $\boldsymbol{A}$. Actually, this is a projection to the closest point in RGB-space inside the gamut. This way, luminance for inside gamut points is corrected for leakage and reproduced exactly and luminance for outside gamut points is largely maintained. Therefore, ghosting can appear for outside gamut points.

6.7.2 Complete Ghosting Elimination

To be able to completely eliminate ghosting points that get mapped outside the gamut need to be removed. This can be done by applying a transformation to the input color values. The luminance values are restricted to lie within given intervals in a way that the resulting combinations of luminance values (forming an axis-aligned rectangle) lie completely inside the gamut. By mapping $\boldsymbol{C}_\mathrm{l}^\mathrm{new} = 1 - z_\mathrm{r} + (2z_\mathrm{r} - 1)\boldsymbol{C}_\mathrm{l}^\mathrm{old}$ and $\boldsymbol{C}_\mathrm{r}^\mathrm{new} = 1 - y_\mathrm{r} + (2y_\mathrm{r} - 1)\boldsymbol{C}_\mathrm{r}^\mathrm{old}$, one obtains a range having the luminance of pure red and pure cyan in its corners for the red-cyan glasses, shown as dashed line in Figure 6.7. The point $\boldsymbol{A}$ is transformed to $\boldsymbol{A}^\mathrm{new}$, one obtains $Y_\mathrm{l} = 0.091$, $Y_\mathrm{r} = 0.892$, and (0.026, 0.901, 0.901) as resulting anaglyph color. The dynamic range with respect to the luminance on the right eye still is large. Unfortunately, it significantly reduces the dynamic range with respect to the left eye (Y_l). One cannot produce as bright colors as before, neither can one produce dark color on the red eye. Nevertheless, one can guarantee that no ghosting will appear in the output image. As noted above the intervals are not predefined; usually, dynamic range for one eye can be traded for dynamic range for the other one. The dotted line in Figure 6.7 shows intervals that slightly improve the range for the left but significantly reduce the range for the right eye.

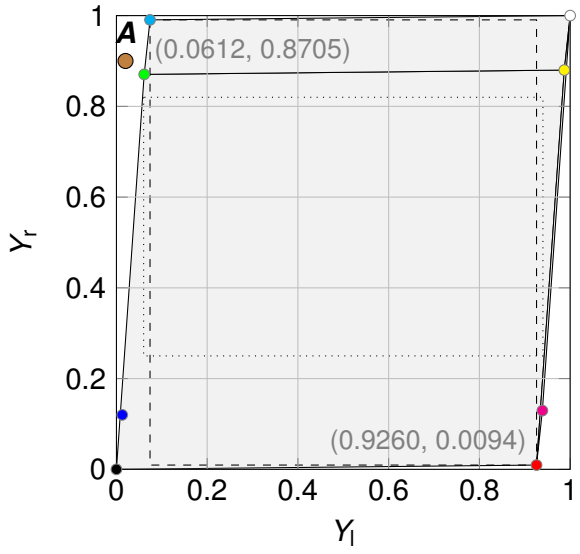


Figure 6.7: This figure illustrates the perceived luminance through the left and right glasses of the red-cyan filter (presented in Section 6.5.2) for different colors displayed on the screen. Luminance combinations that cannot be obtained through the glasses are shown outside the gray area. In particular, low luminance on the left eye cannot be obtained when there is high luminance on the right eye (point ***A***). Restricted working spaces are illustrated by dashed and dotted lines.

6.8 Application to 3D Scatter Plots

The anaglyph rendering technique has been integrated into MDA, revealing important issues when applying anaglyph rendering to 3D scatter plots.

The first issue is related to stereopsis as such. Section 4.1.2 explained that perspective projection should be avoided for 3D scatter plots, because it alters the x/y screen positions of the points and makes exact judgments impossible, limiting the utility of 3D scatter plots compared to their 2D counterparts. Unfortunately, for stereopsis to have an effect it is necessary to use perspective projection.

The second issue is related to anaglyph stereo. The half color anaglyph technique can be applied to minimize binocular rivalry. Color coding is often used in information visualization. About eight to nine colors can be differentiated [War04]. However when using red-cyan glasses the only perceivable colors are a yellowish-green, a purple-blue, and their gray mixture. However this issue can be alleviated by using anachrome, ColorCode 3D, or INFICOLOR anaglyphs, which have better color reproduction.

When looking at Figure 6.8 a third issue becomes obvious, since there are just points in 3D space and no closed surface, it is hard to fuse the left and right image. Kinetic depth effect helps fuse the right points to benefit from stereopsis. Texture could also

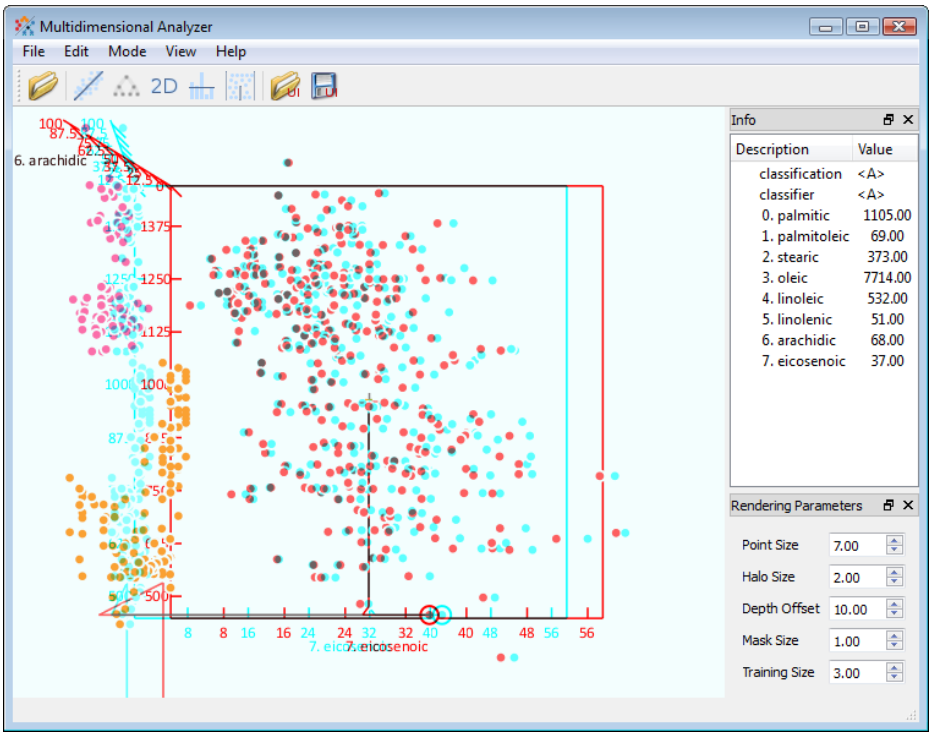


Figure 6.8: Anaglyph 3D Scatter plot showing the same data set as Figure 5.5. This half color anaglyph is ghosting corrected to be viewed on the setting reported in Section 6.5.2 (Dell U2410 monitor with red-cyan glasses) according to the complete ghosting elimination method.

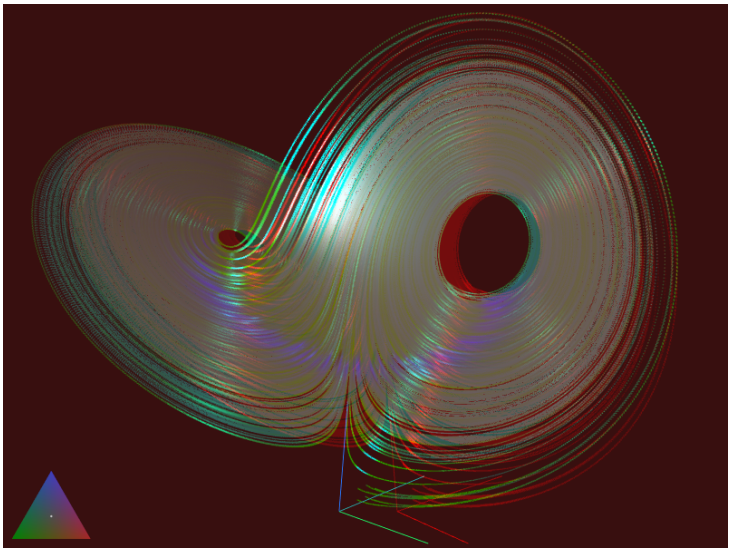


Figure 6.9: Anaglyph 3D Scatter plot showing the illuminated Lorenz attractor, compare to Figure 3.1.

improve the situation for static views by making the discrimination of points possible.

Overall the utility of anaglyph 3D rendering of sparse scatter plots is limited and the techniques presented in Chapter 3 should be considered instead. However, when considering dense scatter plots the illumination technique provides enough texture to make the scatter plot structure well perceivable even from static images with the 3D anaglyph technique, see Figure 6.9.

CHAPTER 7

DISTRIBUTED VISUALIZATION

During the last decade rendering hardware for the consumer market has emerged and run through a speedy development driven mainly by the game industry. Visualization has strongly benefited from this development making a large amount of interactive visualization techniques possible.

The visualization process can be decomposed into stages forming the visualization pipeline [HM90], see Figure 7.1. The immediate beneficiary of rendering hardware is the rendering stage of visualization. Generally the user wants to manipulate each stage of the visualization process to be able to get an understanding of the data. The results should be provided at interactive rates. GPUs consist of many parallel processing units, which individually are slower and less flexible—single instruction, multiple data (SIMD) [Fly72] architectures—than CPU cores but due to their number have an enormous computational power. To allow exploiting this power not only for rendering, programming languages and paradigms have been developed for GPU-computing. Converting CPU-code to GPU-code will often not result in a faster running algorithm, since GPUs are parallel and less flexible than CPUs. Many problems have large non-parallel parts—see Amdahl's law [Amd67]—others cannot be vectorized efficiently to benefit from SIMD hardware and overall it is harder to develop code for parallel processors especially for GPUs. However, growing flexibility allows transferring more and more of the visualization process to the GPU. The Fourier transform is a good example which can be used in the

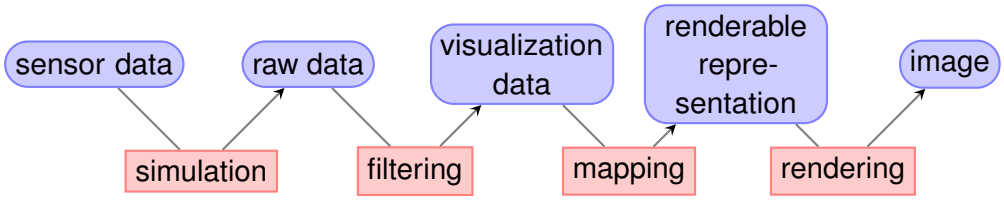


Figure 7.1: The visualization pipeline, extended with a simulation stage.

filtering stage. Such a method for the Fourier transformation of large image data on GPUs was developed together with Kauker, Frey, and Ertl [KSFE10]. However, there are techniques which require even more computation time and do not permit interactive visualizations on a single PC.

The illuminated scatter plots technique introduced in Chapter 3 is such an example. Users want to explore various kernel sizes, but with growing kernel size the computation time grows, too. A combination with the interactive 3D scatter plot navigation technique introduced in Chapter 4 would be interesting, but would not be interactive on a single PC.

To address such issues, a system using various personal computers distributed in a network was developed together with Nazario Cipriani and will be presented in this chapter. The system was published in [SCW11].

The visualization framework presented here is integrated in the context-aware Nexus system that uses a common underlying stream processing middleware for tight integration of data accessing, processing, and visualization. Context-aware systems [SAW94, Dey01] have been emerging with the trend toward ubiquitous and mobile computing. They use context information (especially the user's current positions and situation) to react to changes of the environment. Context information can be acquired and used by individual applications separately or managed by a dedicated platform for context data. Such platforms can collect data from different data providers and provide an interface to query that data. Context-aware systems are often realized on mobile devices such as mobile phones, smartphones, PDAs, or laptop computers that do not have the computational power to perform complex tasks. Therefore, a dedicated hardware infrastructure might be required for data processing. Moreover, context data changes over time and thus continuous data processing is needed.

In this case, stream processing is used, supporting parallelism on distributed and shared memory multiprocessors. The integration of visualization modules into a Java-based stream processing framework for context-aware systems is presented, with focus on efficient communication and parallelization. The approach is demonstrated for the example of a flow visualization scenario.

Stream processing systems, developed in the database community, are able to answer such continuous queries by mapping them to a network of operators managed and executed by a processing middleware. The NexusDS system by Cipriani et al. [CEB+09, LBCS10] was the first stream processing system from the database community targeting context data processing and considering visualization as promising application. NexusDS is part of the Nexus [DHN+04] platform, which provides a flexible middleware for context-aware applications. Nexus is designed as an open platform where everybody can contribute context information that is federated within the platform. NexusDS supports structured and unstructured data, which is necessary to produce useful image output. NexusDS uses the JXTA peer-to-peer protocol to enable communication over

different network protocols on different architectures supporting TCP/IP and Bluetooth communication, discovery, and NAT-traversal. NexusDS operators are executed only when new input data arrives and new output needs to be generated supporting power-aware computation. These properties make NexusDS well suited for a wide range of pervasive computing applications, for example integration of geo-spatial simulation results such as wind flow, contamination, or disaster recovery.

Here the extension of the Java-based NexusDS framework is presented which allows the development and integration of C/C++ modules, which are most commonly used in the scientific visualization community. The module architecture is quite common from the visualization point of view, but a tight integration in the operator concept is performed. NexusDS supports parallelism on distributed and shared memory multiprocessors. Therefore, here the focus lies on efficient communication and an extension of the framework by a technique which helps easily parallelize operators exploiting data parallelism.

The computational resources needed for a visualization may not be constant: they can depend on the input data (different data may require more computational work), the visualization accuracy needs (some applications may require higher accuracy than others), or the targeted level of interactivity (some applications like for augmented reality require low latency, while a delay of a few seconds may be acceptable for Web-based monitoring applications). Through stream processing a highly adaptive and scalable solution is obtained: the same visualization pipeline structure can be used to fulfill all these requirements. For example, in case of low hardware requirements, the visualization pipeline can be deployed to a single PC running multiple visualizations. In case of higher resource needs, multiple PCs can be used to execute the same visualization pipeline, leading to low latencies and high data and image throughput.

Given the large variety of data sources provided by a context-aware platform with the system various new applications can be realized which benefit from scalable visualization capabilities.

7.1 Related Work

This section reviews the main fields of previous research related to this work: context-aware systems and parallel visualization. Both fields are traditionally covered by mostly disjoint research communities. However, the trend toward ubiquitous information systems with heterogeneous data sources has recently brought these fields closer together. In particular, the massive increase in available data from sensors and computer networks in context-aware systems has emphasized the need for appropriate visual data analysis and presentation.

A typical example of a platform for context-aware mobile applications is given by Raento et al. [ROPT05], where information of mobile phones such as position, phone

profile, and last phone usage are collected and made available to the persons in the contact list. An alternative operator-based context-aggregation platform was presented by Chen and Kotz [CK02]. PLACE* [XECA07] is a distributed spatiotemporal data stream management system for moving objects. PLACE* supports continuous spatiotemporal queries that are evaluated by a network of regional servers. A query is continuously answered by a querying server, a tracking server, and a set of additional participating servers. While any of these context-aware systems supports accessing context data in some way or another, they do not facilitate efficient, low-latency, and generic visualization techniques.

Augmented reality systems also provide narrow context information like position and pose estimates of mobile objects, but offer only a limited scalability. Shibata et al. [SHF+06] present a scalable architecture, however scalability is limited to the ability to serve clients with different capabilities by one server, without considering parallelization of the rendering or visualization on the server side.

Computer graphics and visualization considered efficiency issues early on. Parallelization in particular has been a popular means of increasing processing speed. Early work on parallelization in the area of visualization concentrated on parallel rendering [MCEF94]. In contrast, here parallelization of the visualization process as a whole and not just its rendering stage is considered. A good description of the generic concepts of task, data, and pipeline parallelism in visualization is presented by Ahrens et al. [ALS+00]. The data-flow paradigm is the common basis for typical visualization processing, as used in scientific visualization environments like AVS [UFK+89], SCIRun [MHJ99], The Visualization Toolkit (VTK) [SML03], and COVISE [Cov]. The data-flow approach can be combined with parallelization to support fast processing of large data sets. However, beyond the common data-flow model and parallelization strategies, visualization systems differ in the details of data communication and workload distribution.

AVS [UFK+89] and COVISE [Cov] use a demand-driven execution model with a centralized executive. SCIRun [MHJ99] as well has a centralized executive. VTK [SML03] is an open source visualization toolkit with a demand-driven update semantics. Ahrens et al. [ALS+00, ABM+01] saw that designing an efficient mechanism for controlling many processes from one centralized executive is difficult and developed a parallel extension to VTK that has no centralized executive. In another extension for VTK, Moreland et al. [MT03] included parallel rendering components. Finally, Dutra et al. use VTK for distributed visualization [DRGS07]. Similar to this work, they use a Java-based toolkit to allocate resources and to communicate data. They propose a master-slave architecture where the master splits the data and merges the final results. This differs from the approach presented here where the deployment of operators is more flexible. ParaView [Par, CGM+06] uses VTK as data processing and rendering engine and avoids the use of a centralized executive to obtain a more scalable solution than AVS Express or SCIRun. In contrary to the system presented here, it has a demand driven update

semantics and only focuses on exploiting data parallelism but not task and pipeline parallelism.

Information visualization toolkits like the InfoVis Toolkit [Fek04], Prefuse [HCL05] (`http://prefuse.org`), or the JavaScript InfoVis Toolkit (`http://thejit.org/`) provide various data visualization methods but do not focus on distributed parallelization.

VisTrails [BCC+05] aims at speeding up multiple-view visualizations by eliminating multiple identical processing steps, but does not exploit parallelism. DeVIDE [BP08] employs a hybrid demand- and event-driven scheduling but uses a central scheduler which executes the modules in sequence.

NMM by Lohse et al. [LWRS08] is a stream processing framework running on distributed and shared memory multiprocessors targeting multimedia applications. In contrast to the framework presented here it has no visualization modules. FlowVR [AGL+04] is a middleware for distributed and shared memory parallel systems targeting virtual reality (VR) and scientific visualization applications dealing with large displays. FlowVR encourages the usage of MPI for high performance applications which indicates that FlowVR's own abstraction for the distribution of modules over a network does not offer a performant solution. In contrast to the system presented here it does also not focus on context data processing.

NexusDS [CEB+09] is a flexible and scalable middleware that is highly customizable. The complete system is implemented using the Java programming environment. While different elements of NexusDS appear in the above prior systems at various places, NexusDS is unique in its combination of event-driven communication (simplifying parallelization), decentralized execution (better scalability), and support for distributed and shared memory systems. This new combination of methods allows us to integrate parallel processing and visualization with high throughput and low latency in general streaming environments.

7.2 Conceptual Model and Structure

7.2.1 NexusDS Operator and Streaming Model

Now a brief overview of the functionality of NexusDS is given which is essential in order to understand the extensions to NexusDS and some configuration choices made.

The distributed stream processing framework consists of a set of stream processing nodes that are capable of finding each other in the network, accepting processing requests, and delivering results. A processing request is represented by a set of operators that form a processing graph. Operators have several input and output ports that are interconnected, thus forming a network of operators. An execution environment is available to execute the operators.

NexusDS allows parallelization on distributed and shared memory systems. Modules

running on the same machine are executed in one process, the common address space allows sharing data efficiently. To communicate data between modules running on different machines, data is serialized, sent over the network, and deserialized by the receiver.

The stream processing system is suitable to handle time varying data. To hide latencies introduced by the network transfer, processing and receiving of data is asynchronous. Each input port has a separate input queue. When data arrives it is processed; the results are pushed to the input queues of all operators connected to its output port. Therefore no centralized execution module or scheduler is needed to drive the stream processing system.

NexusDS provides abstract operators, so-called platform sinks and sources, which can handle sending and receiving data over the network between fragments during runtime. These operators allow communicating data over the network asynchronously.

7.2.2 Graphical Editor

After the development of the operators they have to be interconnected. Therefore a graphical editor was developed, which is similar to other dataflow-based visual programming environments. Available operators are stored in a repository which is read by the editor. Operators can be dragged and dropped to a GUI drawing area where the interconnections between the operators can be defined.

Once the visualization pipeline is designed it can be separated, within the graphical editor, into one or more so-called fragments. Each fragment groups operators that should run on the same stream processing node. During this step necessary platform sinks and sources are created automatically.

7.2.3 Deployment and Execution

After design time each visualization pipeline fragment needs to be assigned to a stream processing node. Several fragments may be assigned to a single node. The assignment is specified by the user taking into account the computational capabilities of the nodes and the interconnection between them.

Once all mapping information is available, the visualization pipeline fragments are deployed on the corresponding stream processing node. Each node is responsible for establishing the proper connection to its neighbor and for receiving, processing, and sending data in a push-based manner.

As a final result, the output of the processing pipeline can be displayed on the stream processing node running the rendering operator, or images can be sent to an image client operator for display on mobile devices.

Distributed-memory and shared-memory task decomposition and pipelining are supported by the NexusDS framework by placing operators onto different computational

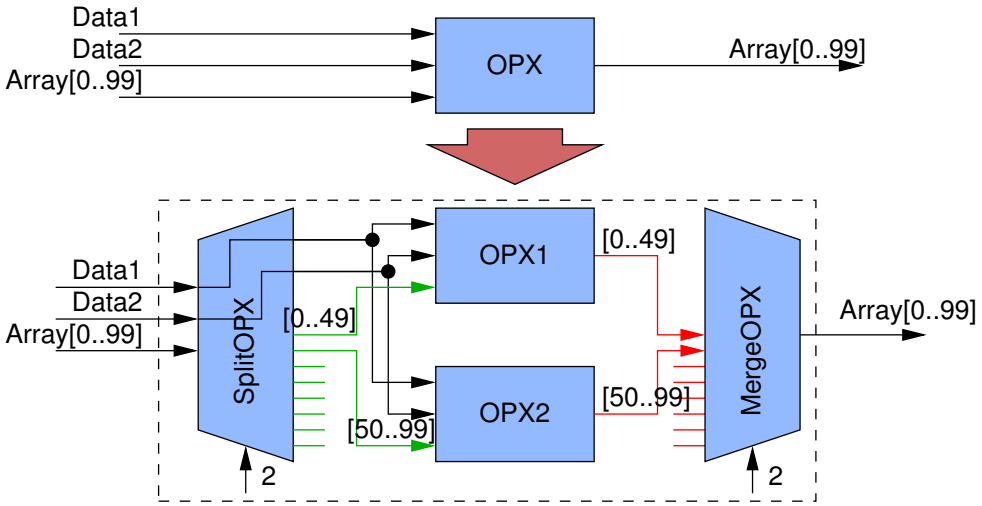


Figure 7.2: Domain decomposition of operator X. The operator receives an array of variable length (100 elements in this example). These elements are processed depending on two additional data elements, Data1 and Data2. Domain decomposition is achieved by creating multiple instances of the operator (two in this example). A split operator distributes the work equally to all operator instances (green lines). It also sends the incoming additional data elements to all operator instances. Finally a merge operator concatenates the partial results (red lines).

nodes or to one node respectively.

7.2.4 Domain Decomposition

Figure 7.2 illustrates how domain decomposition was realized with NexusDS for an example where an operator is decomposed into two independently working operators. The number of array elements may change for each data packet. Moreover, the number of operators can vary, as specified by the module parameters during module initialization.

7.3 Data Transfer and Execution Model

7.3.1 Efficient Communication

NexusDS is implemented in Java. For the development of efficient visualization operators the C++ language was chosen. The Java Native Interface (JNI) is used to call C/C++ code from Java. To implement efficient communication between the C++ visualization operators over the Java NexusDS framework `java.nio.ByteBuffer` objects were used. These objects are wrappers for C++ memory blocks, so that C++ objects can be handed

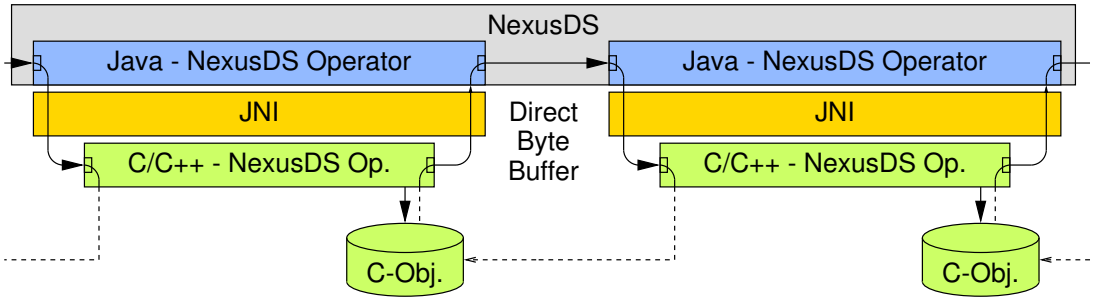


Figure 7.3: Operators send direct byte buffers over the JNI interface.

over to Java without copying the data itself. The Java NexusDS system can transport references to data between operators if the operators are on the same node, see Figure 7.3. A design decision was made not to use linked but flat data structures. If the operators run on different nodes, flat data structures also prove beneficial. Since the data is already in a linear memory block, it can be sent over the network and stored in a local memory block on the receiver side. For this task NexusDS was extended with platform sinks and sources which can handle direct byte buffer objects, using the `java.nio.ByteBuffer` class.

7.3.2 Memory Management

The operators allocate memory for their outputs on the native (C++) side. They just hand over constant references to other operators, i.e., operators are not allowed to modify input data, but they can output more than one reference to the same memory to different operators. The communication channels are unidirectional and thus there is no mechanism to explicitly tell an operator that a given input is not needed any more. Therefore, a different mechanism was implemented: the input becomes invalid after the next data packet has been received on the same input port. Analogously, the memory of an operator output has to be kept until another output element is sent via the same port. This implies that the receiving operator should block if data is coming through an input port when the previous element that came through the same input port is still needed for computations. In contrast to the solution presented by Ahrens et al. [ALS+00], local copies are entirely avoided using this paradigm. As NexusDS uses internal queues for the input ports, the number of outputs an operator has to keep increases with the size of these queues.

NexusDS was configured to use first-in, first-out (FIFO) queues in the system. The size of the input queues needs to be small to keep latencies low. In case of uneven concentration of incoming events the queues should be increased to allow high utilization of nodes. (This is not an issue in the example scenario of Section 7.4.)

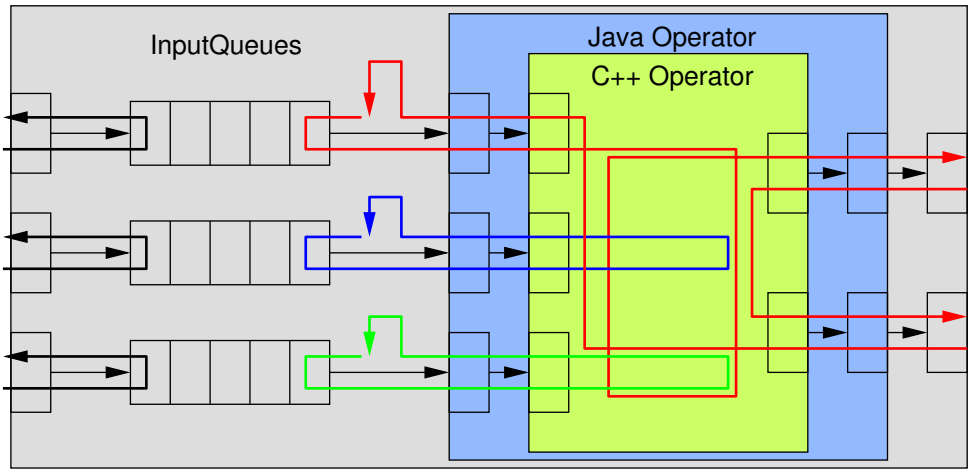


Figure 7.4: This operator has three input and two output ports. Threads running inside the operator—denoted as red, blue, and green arrows—are associated with the input queues.

7.3.3 Thread Model

NexusDS is responsible for the data transfer between operators and also provides an execution model. An operator can have several input ports, see Figure 7.4. Each port has an input queue for incoming data. NexusDS was configured to associate one thread to each input queue, this thread is waiting until the input queue is not empty. Then the respective input method of the Java operator is called, which calls the native input method of the C++ operator. The operator processes the data and eventually outputs results to some output ports. Since each input queue delivers input data to the operator concurrently, synchronization has to be performed by the C++ operator.

7.3.4 Communication over the JNI

For the communication between the Java and the C++ side the Call-Invoke pattern [WK00] was used. To output data from the C++ operators each operator has one registered listener that sends the output of all output ports back to the Java part of the operators.

Resolving a C++ Object over the JNI

It is not possible to call a C++ object method over the JNI, but only C functions. In case of multiple operator instantiations, it is therefore necessary to call the right instantiation. This problem is solved by storing a C-pointer inside the Java classes that is passed to the C function as an additional argument, which is then used to resolve the right C++ object.

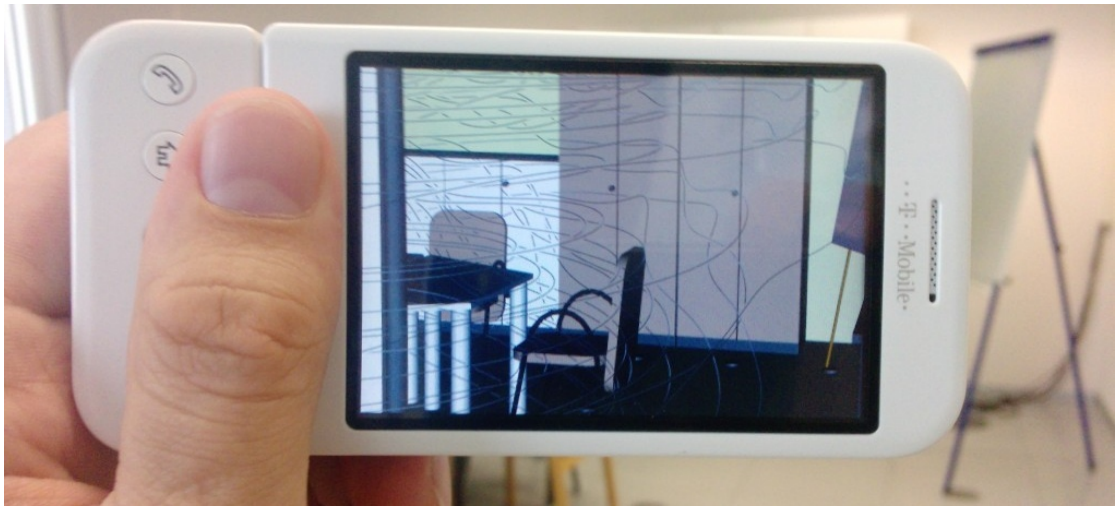


Figure 7.5: Air flow visualized on a mobile client.

Multi-threaded Execution

Since a separate thread is associated with each input port, the operators receive data in different execution threads. To resolve the Java object and the method for sending back the operator output, a JNI interface pointer and the jobject pointer are needed. These are valid only for the thread associated with it [Lia99]. By saving these pointers in thread local storage it is possible to resolve the right pointers and allow input queue threads to run concurrently on the C++ side.

7.4 Context-Aware Mobile Visualization

The presented framework can be used to facilitate ubiquitous computing by implementing demanding visualizations based on context data for mobile clients.

In this example, air flow in a room is simulated and visualized (see Figure 7.5). However, due to the generality of NexusDS and the tight integration of visualization modules, any other visualization example should be workable within the environment. For general background information on flow visualization, see overview presentations like [WE05].

The operators for the whole data generation and visualization process are shown in Figure 7.6. Dynamic changes in the scene are input by tracking operators. The status of the windows and doors are obtained by computer vision techniques and are already stored in a database, called the context server. Similar object recognition techniques for the Nexus system have been presented together with Eissele and Ertl. [ESE09]. Data which does not change too frequently and is not time-critical can be stored in the context server.

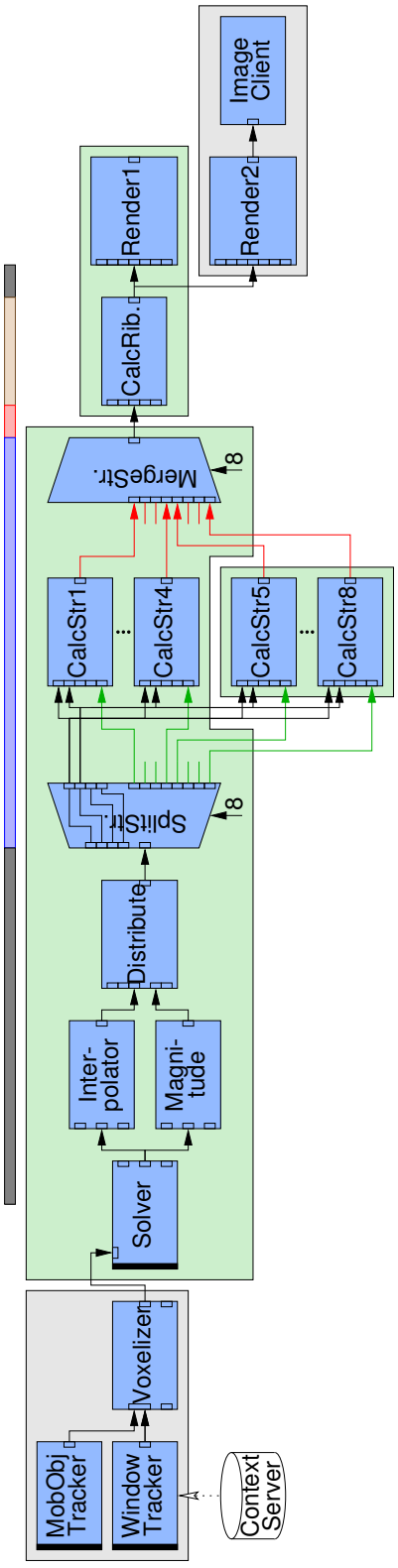


Figure 7.6: Schematic view of the F8S operator-network, containing 8 streamline-calculation operators. Green and gray areas represent different computational nodes. The gray nodes were not part of the measurement setup. The colored bar refers to the measurements: Streamline calculation time (blue) incorporates the split- and merge-operations. The transfer time (red) denotes the time for transferring the streamlines to the node where the stream-ribbon calculation and the rendering is executed. Stream ribbon calculation time (brown) refers to the execution of the stream ribbon calculation. The time to execute the whole pipeline minus the former three time measurements is referred as "other" time (gray).

The Window-Tracker module streams this data in by regularly querying the database and generating output when the status changes. The MobObj-Tracker module is responsible for tracking moving object in the room, which is an example of a permanently updated context-data source. Eissele et al. [EKE08] considered the tracking quality in a similar scenario as additional context information.

The changing geometry is voxelized and sent to a solver operator that calculates the velocity in the simulation volume (using a finite-volume numerical solver for computational fluid dynamics). The output of the solver operator is the air flow velocity field discretized on a regular grid. Then, the velocity field is sent to interpolation and velocity magnitude calculation operators. Their results are used to automatically place seed points for streamlines at appropriate positions to capture the important features of the flow. The most time-consuming task is the subsequent computation of streamlines (CalcStr) by solving the ordinary differential equation for particle tracing (with 4th-order Runge-Kutta integration and tricubic interpolation). Based on the streamlines stream ribbons are calculated (CalcRib), which are visualized by one or more render clients. Render operators can also output an image stream which can be sent to remote targets (e.g., mobile graphics devices) by image client operators. Constant generation of new data is triggered by the tracking and solver operators, leading to data streaming through the complete process of Figure 7.6.

The links between the operators in Figure 7.6 are just partially shown to avoid clutter in the diagram. The render operator, for example, can also display streamlines or LIC (line integral convolution) and needs the tracked positions of mobile objects connected to the input ports, which are left empty in Figure 7.6.

The render operator uses the OpenSceneGraph (OSG) application programming interface (API). The test platform consisted of PCs with Core 2 Quad Q6600 CPUs, 4 GB DDR2 main memory, and gigabit Ethernet interconnection.

Different configurations were run to assess the efficiency of the framework (see Figure 7.7). For the measurements the tracking operators are neglected, which are asynchronously feeding new positions of the tracked objects to the solver operator, and a single render operator was considered. A single-threaded version was first tested, with one single streamline calculation operator (VC), where the operators run on one node and were linked directly by listeners, without the overhead of the communication over the JNI. Next a single-threaded configuration was created where the operators were instantiated in Java and were connected in the Java classes (VJ). So the overhead of the communication over the JNI could be measured. Then NexusDS was used to run the operator-network on a single node (F1). As the framework creates one thread for each input port of an operator this was a multi-threaded version. In this way the Interpolation operator and the Magnitude operator were running concurrently, exploiting task parallelism. With this configuration the overhead of the framework was tested. The measurements show that approximately 95% of the time is spent in the streamline calculation (see Figure 7.7 VC).

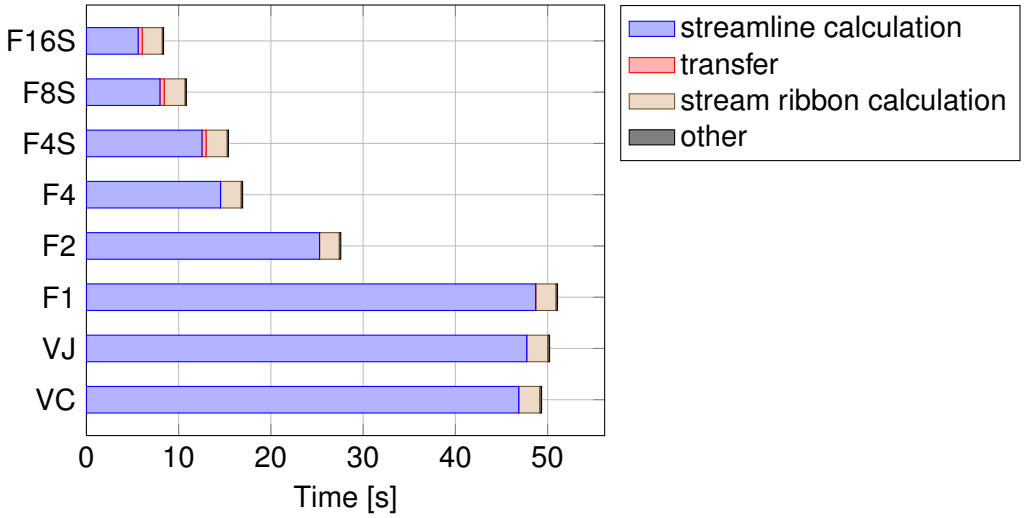


Figure 7.7: Measurements showing the breakup of timing results for different test configurations.

To show the scalability of the framework, several configurations were created where the streamline calculation was parallelized with the domain decomposition presented in Section 7.2.4. Configurations were created with 2 and 4 CalcStreamline operators running in parallel (F2, F4) on one node together with all other operators. Since OSG, which was used for final rendering, consumes a full CPU core the rendering and the stream ribbon calculation parts were moved to a separate computing node (F4S). Finally, configurations with 8 and 16 parallel streamline calculation operators were created, with up to 4 streamline calculation operators running on one node and with the stream ribbon and rendering operators running on a separate node (F8S, F16S).

From the measurements one can conclude that there is no significant overhead – neither for the thin Java communication interface nor for the streaming operator framework. By moving the stream ribbon calculation and the rendering to a separate node, an improvement regarding the streamline calculation time was obtained (see Figure 7.7 F4 and F4S). Since the results of the streamline calculation now have to be transferred to a different node, the transfer time is growing but the configuration pays off since all four cores are available for streamline computation. The setup is even more beneficial when several streamline calculations need to be performed and pipeline parallelism is exploited.

CHAPTER 8 CONCLUSION AND OUTLOOK

This thesis focused on techniques which help explore multivariate data using 3D visualizations. It showed that in many cases 3D visualizations can provide more information than 2D and even improve task performance. This benefit was obtained by the careful usage of depth cues to support depth perception without giving up the benefits of 2D visualizations, see Table 8.1. This thesis also presented a significant improvement in the usage of the stereoscopic depth cue on consumer hardware by substantially reducing ghosting artifacts in anaglyph stereo presentations.

To show the relation of three attributes, 3D scatter plots can be used instead of a more traditional 3x3 scatter plot matrix. Depth perception of 3D scatter plots can be improved by the extraction and illumination of linear, planar, and volumetric structures. The introduced technique is suitable for highlighting structures in dense scattered data. It relies on the presence of structures and on a sufficiently high density of samples to display smoothly shaded structures, which support the user in inferring shape from shading. The novel technique can be combined with existing ones, like blending, color mapping, halo rendering, linked views, or scatter plot matrix navigation. Additionally, a new halo rendering technique was presented which draws halos only at significant depth discontinuities reducing clutter and effectively supporting the occlusion depth cue.

When dealing with multivariate data a single 3D scatter plot is not enough. This thesis presented an interpolation scheme and projection technique for navigation between scatter plots. In particular, an animation method has been introduced that supports transitions between 3D scatter plots that are perceived as 3D rigid body rotations. The novel transition scheme nicely fits to navigation with 3D scatter plot matrices. The technique has the benefit of preserving context by navigating between different 3D scatter plots instead of just switching between them. It provides a way for the user to easily follow the rigid body motion of the points of the scatter plot, heavily reducing the cognitive load that would be associated with the unconstrained animation of data points between scatter plots. A controlled user study has shown that rigid body rotational transitions are more effective than interpolating transitions. The usefulness of the 3D scatter plot navigation

Table 8.1: This table shows the various depth cues playing central roles in the presented techniques. X means the depth cue has been applied, O means the depth cue has been intentionally left away in the technique, and an empty field means the depth cue is not the focus of the technique. The table also shows the subset of the techniques which are used by MDA.

		depth cues					
technique	Chapter/Section	shape-from-shading	occlusion	kinetic depth effect	linear perspective	stereoscopic cues	MDA integration
Illuminated 3D Scatter Plots	3.1	X					O
Halo Rendering at Depth Discontinuities	3.5		X				X
3D Scatter Plot Navigation	4			X	O		X
Decision Tree Navigation	5.1			X	O		X
Decision Tree Hyperplane Rendering	5.2.4						X
Anaglyph Stereo without Ghosting	6					X	X
Distributed Visualization	7						O

approach is largely subject to the degree of usefulness and appropriateness of single 3D scatter plots. The utility of 3D scatter plots, compared to 2D scatter plots, can suffer badly from inaccuracies due to perspective projection and from ill-designed rendering that would fail to support depth perception from static views and rely only on structure from motion. Since 3D scatter plots without additional depth cues look like 2D scatter plots when viewed along the major axes, they only provide additional information when they are extended with depth cues but have no drawbacks from such static viewpoints. For application areas where interactive 3D scatter plots are useful, the enhancement by 3D scatter plot navigation will be useful as well. This usefulness has been illustrated for an application example from natural language processing. Additionally, a qualitative study has been conducted to evaluate the usefulness of the 3D scatter plot matrix navigation.

Scatter plot navigation with decision trees is a novel technique for exploring multidimensional data sets. It facilitates the understanding of the decision tree and the multivariate data likewise. Two application examples were presented. One is from the computer-vision domain concerning the classification of trajectories with decision tree classifiers. The other example deals with a multivariate data set for which machine learning can be employed to train a decision tree classifier that can be examined with the novel technique to get an understanding of the data. During feature engineering, domain experts usually try to develop new features that allow them to separate the false positives

from the true positives and the false negatives from the true negatives. The number of mutual comparisons an expert has to examine can become quite large. By navigating towards the leaf nodes of the decision tree, the analyst can focus on similar data items, greatly reducing the number of combinations but still focusing on the most interesting ones. Therefore, the introduced technique has great potential for feature engineering.

Sometimes domain knowledge is essential to analyze data. In such cases it is beneficial to design special visualizations which help the analysts in the task they want to perform and not simply treat data as multivariate. This was the case for the geospatial matching scenario. From the visual exploration of the data several properties have emerged, which helped improve the classifiers. First, annotation, although well done, can be improved by finding errors with high impact for the later classification in the scatter plot. Second, the visual analysis of the classification errors helped design new features to improve the classifier. Especially false positives that have corresponding items motivated the use of an iterative classifier. In most cases, these classifiers perform better than non-iterative classifiers and they are robust to new domains since the choice of the training set is not as striking as for the non-iterative classifiers. When considering a high-dimensional feature space, the scatter plot visualization helps look at the right spot but does not separate the matches from the non-matches entirely. On the other hand if a separation in a 3-space projection would exist, the solution to the classification problem would be easy to find, making the problem less interesting. In general, visual exploration tools have helped the natural language processing domain experts develop and improve a classifier, as demonstrated by substantial improvements of the F-score results. Mostly, well known visualization components could be used, such as 2D plots, glyph plots, scatter plots, navigation, and interaction techniques. However, some specific visual mappings had to be developed, in particular, for visualizing high-dimensional feature space and decision tree hyperplanes.

When dealing with 3D scatter plots, 3D structure has to be presented to the user through depth cues. Stereopsis is probably the most compelling depth cue. This thesis has presented a model to capture the luminance perceived through anaglyph glasses that can be used to substantially remove ghosting artifacts from anaglyph stereo images. The model parameters can be estimated quickly with just six measurements. The method can be used to render full color, half color, and gray anaglyphs without luminance ghosting, as well as to remove luminance ghosting artifacts from anaglyphs created with a naïve method. The novel method is not limited to red-cyan glasses. As it determines which glasses are in use during calibration, no further information is required. The results of the method have been illustrated for different display devices and filters and the results have been related to a real monitor and filter combination. Ghosting can be eliminated independently of the luminance combinations in the source images by sacrificing dynamic range. The novel method could be of interest since it greatly enhances anaglyph stereo rendering. It is versatile and can also be combined with current approaches. Especially

the combination with the magenta-cyan technique targeting reduction of retinal rivalry could be interesting.

Several visualization methods require computationally demanding processing. This thesis has shown how to integrate common visualization modules in a stream processing framework supporting parallelization on shared and distributed memory systems. Special attention was put on efficient communication and a scalable adaption mechanism for data parallel processing was developed. In addition to parallelized visualization, the framework is designed to transparently fit into large, heterogeneous systems—both in terms of hardware infrastructure and software architecture.

Most of the visualization techniques presented in this thesis can be combined and are supported by the Multidimensional Analyzer. An integration of the illuminated 3D scatter plot visualization technique would also be valuable. This could be achieved by realizing the pre-processing as a visualization module which could be executed by the stream processing framework. The algorithm processes each point individually and would highly benefit from domain decomposition.

A significant result of this thesis and with relevance to the information visualization community is the result obtained in the user study showing that 3D rigid body rotations (3D scatter plot rotation) lead to better task performance than interpolation of positions (2D movement), presented in Section 4.3. Another main contribution is the improvement of the anaglyph stereo rendering, which has very general applicability.

8.1 Outlook

A crucial issue with novel visualizations is the evaluation of their usefulness. Certain aspects of the presented approaches have been evaluated in this thesis. However, "real-world" applications may have different requirements and the novel approaches need to prove useful for them. Unfortunately the presented methods require a substantial effort for implementation and integration into existing tools. A first step could be to conduct further user studies to gain additional insight into the usefulness of the presented approaches.

While shape perception plays the central role for 3D scatter plots, the comparison of depth to other attributes like color, size, glyphs, or even motion could be an important piece of future work. Additionally to the evaluation issues, the illumination method has high computational demands, which currently prevents an interactive calculation. Further research could address this issue through a hierarchical data structure for the PCA calculation, maybe by approximating the correct solution. Another area of future work could investigate techniques for the automatic neighborhood size estimation, which is strongly dependent on which structures the users want to be highlighted.

The 3D scatter plot interpolation and projection technique is quite general and besides the navigation with 3D scatter plot matrices or with decision trees additional application

domains could be investigated.

The reproduction quality of anaglyph stereo in print suffers even more from ghosting than on display devices. The direct application of the anaglyph stereo without ghosting method is not applicable since the color reproduction is different. Nevertheless, the main idea of the method, to reproduce luminance, is still a valid criterion which could lead to new techniques for the anaglyph reproduction in print media.

As general outlook for the future I suppose that now that the computational power is seldom the bottleneck, especially for information visualization applications, the evaluation of visualization techniques plays a more central role. This would also strongly benefit from further interdisciplinary work since the people concerned with visualizations often are searching for the right application of their visualization techniques.

BIBLIOGRAPHY

[Abb84] E. A. Abbott. *Flatland: A Romance of Many Dimensions*. second, revised edition, 1884. 12

[ABM+01] J. Ahrens, K. Brislawn, K. Martin, B. Geveci, C. C. Law, and M. Papka. Large-scale data visualization using parallel data streaming. *IEEE Computer Graphics and Applications*, 21(4):34–41, 2001. 108

[AdO04] A. Artero and M. de Oliveira. Viz3D: Effective exploratory visualization of large multidimensional data sets. In *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*, pages 340–347, 2004. 12

[AEK00] M. Ankerst, M. Ester, and H.-P. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pages 179–188, 2000. 59

[AGL+04] J. Allard, V. Gouranton, L. Lecointre, S. Limet, E. Melin, B. Raffin, and S. Robert. FlowVR: a middleware for large scale virtual reality applications. In *Proceedings of the International Euro-Par Conference*, volume 3149 of *Lecture Notes in Computer Science*, pages 497–505. Springer Berlin / Heidelberg, 2004. 109

[ALS+00] J. Ahrens, C. Law, W. Schroeder, K. Martin, and M. Papka. A parallel approach for efficiently visualizing extremely large, time-varying datasets. Technical Report LAUR-00-1620, Los Alamos National Laboratory, 2000. 108, 112

[Amd67] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the Spring Joint Computer Conference*, pages 483–485, 1967. 105

[Ans73] F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973. 14

[Asi85] D. Asimov. The grand tour: A tool for viewing multidimensional data. *SIAM Journal on Scientific and Statistical Computing*, 6(1):128–143, 1985. 40

[BA86] A. Buja and D. Asimov. Grand tour methods: An outline. In *Proceedings of the Symposium on The Interface*, pages 63–67, 1986. 40

[Ban94] D. C. Banks. Illumination in diverse codimensions. In *Proceedings of the ACM SIGGRAPH Conference*, pages 327–334, 1994. 21

[BBS+08] R. P. Botchen, S. Bachthaler, F. Schick, M. Chen, G. Mori, D. Weiskopf, and T. Ertl. Action-based multifield video visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):885–899, 2008. 12

[BC87] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987. 18

[BCC+05] L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. VisTrails: Enabling interactive multiple-view visualizations. In *Proceedings of the IEEE Conference on Visualization*, pages 135–142, 2005. 109

[BDM04] H. Boyaci, K. Doerschner, and L. T. Maloney. Perceived surface color in binocularly viewed scenes with two light sources differing in chromaticity. *Journal of Vision*, 4(9):664–679, 2004. 96, 98

[Bec97] B. G. Becker. Volume rendering for relational data. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 87–90, 1997. 17

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. 29

[Bes86] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986. 75

[BFSO84] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1st edition, 1984. 62

[Bli77] J. F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the ACM SIGGRAPH Conference*, pages 192–198, 1977. 23

[Blo08] W. Bloos. Ghosting test. `http://www.stereoforum.org/viewtopic.php?f=16&t=53`, 2008. Last accessed on 8 April 2011. 88

[BLW88] S. V. Bemis, J. L. Leeds, and E. A. Winer. Operator performance as a function of type of display: Conventional versus perspective. *Human Factors*, 30(2):163–169, 1988. 11

[BN01] T. Barlow and P. Neville. Case study: Visualization for decision tree analysis in data mining. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 149–152, 2001. 59

[BP08] C. P. Botha and F. H. Post. Hybrid scheduling in the DeVIDE dataflow visualisation environment. In *Proceedings of the Simulation and Visualization Conference (SimVis)*, pages 309–322, 2008. 109

[BW08a] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1428–1435, 2008. 5

[BW08b] L. Byron and M. Wattenberg. Stacked graphs – geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252, 2008. 60

[CA05] P. Cavanagh and G. A. Alvarez. Tracking multiple targets with multifocal attention. *Trends in Cognitive Sciences*, 9(7):349–354, 2005. 41, 46

[CB97] D. Cook and A. Buja. Manual controls for high-dimensional data projections. *Journal of Computational and Graphical Statistics*, 6(4):464–480, 1997. 39, 40

[CBCH95] D. Cook, A. Buja, J. Cabrera, and C. Hurley. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 4(3):155–172, 1995. 40

[CCH01] D. Caragea, D. Cook, and V. G. Honavar. Gaining insights into support vector machine pattern classifiers using projection-based tour methods. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pages 251–256, 2001. 59, 66

[CCH04] D. Cook, D. Caragea, and V. Honavar. Visualization in classification problems, with examples using support vector machines. In *Proceedings in Computational Statistics*, pages 799–806, 2004. 55, 78

[CCTK83] J. M. Chambers, W. S. Cleveland, P. A. Tukey, and B. Kleiner. *Graphical Methods for Data Analysis*. Chapman & Hall, 1983. 14

[CEB+09] N. Cipriani, M. Eissele, A. Brodt, M. Großmann, and B. Mitschang. NexusDS: A flexible and extensible middleware for distributed stream processing. In *Proceedings of the International Symposium on Database Engineering & Applications (IDEAS)*, pages 152–161, 2009. 5, 106, 109

[CGM+06] A. Cedilnik, B. Geveci, K. Morel, J. Ahrens, and J. Favre. Remote large data visualization in the ParaView framework. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization*, pages 163–170, 2006. 108

[Che08] C. Chen. An information-theoretic view of visual analytics. *IEEE Computer Graphics and Applications*, 28(1):18–23, 2008. 69

[CJ02] M. Cammarano and H. W. Jensen. Time dependent photon mapping. In *Proceedings of the Eurographics Workshop on Rendering*, pages 135–144, 2002. 29

[CK02] G. Chen and D. Kotz. Solar: An open platform for context-aware mobile applications. In *Proceedings of the International Conference on Pervasive Computing*, pages 41–47, 2002. 108

[CLKP10] J. Choo, H. Lee, J. Kihm, and H. Park. iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 27–34, 2010. 59

[CM00] A. Cockburn and B. McKenzie. An evaluation of cone trees. In S. McDonald, Y. Waern, and G. Cockton, editors, *People and Computers XIV - Usability or Else!: Proceedings of HCI 2000*, pages 425–436. Springer-Verlag London Berlin Heidelberg, 2000. 11

[CMS99] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999. 2

[Cov] COVISE. `http://www.hlrs.de/organization/vis/covise/`. Last accessed on 3 November 2011. 108

[CRS+96] E. H.-h. Chi, J. Riedl, E. Shoop, J. V. Carlis, E. Retzel, and P. Barry. Flexible information visualization of multivariate data from biological sequence similarity searches. In *Proceedings of the IEEE Conference on Visualization*, pages 133–140, 1996. 17

[CS07] D. Cook and D. F. Swayne. *Interactive and Dynamic Graphics for Data Analysis: with R and GGobi*. Springer, 2007. 40

[Cut97] J. Cutting. How the eye measures reality and virtual reality. *Behavior Research Methods*, 29:27–36, 1997. 7

[DDG88] A. W. Donoho, D. L. Donoho, and M. Gasko. MacSpin: Dynamic graphics on a desktop computer. *IEEE Computer Graphics and Applications*, 8(4):51–58, 1988. 17

[Dey01] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001. 106

[DHN+04] F. Dürr, N. Hönle, D. Nicklas, C. Becker, and K. Rothermel. Nexus–a platform for context-aware applications. In *1. Fachgespräch Ortsbezogene Anwendungen und Dienste der GI-Fachgruppe KuVS*, pages 15–18, 2004. 68, 106

[DRGS07] M. Dutra, P. Rodrigues, G. Giraldi, and B. Schulze. Distributed visualization using VTK in grid environments. In *IEEE Symposium on Cluster Computing and the Grid*, pages 381–388, 2007. 108

[DSW86] B. A. Dosher, G. Sperling, and S. A. Wurst. Tradeoffs between stereopsis and proximity luminance covariance as determinants of perceived 3D structure. *Vision Research*, 26(6):973–990, 1986. 7

[Dub01] E. Dubois. A projection method to generate anaglyph stereo images. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1661–1664, 2001. 86, 87

[DWA10] T. N. Dang, L. Wilkinson, and A. Anand. Stacking graphic elements to avoid over-plotting. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1044–1052, 2010. 11

[DWE03] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive cutaway illustrations. *Computer Graphics Forum*, 22:523–532, 2003. 27

[EDF08] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1141–1148, 2008. 36, 37, 40, 49, 55

[EG04] B. D. Eugenio and M. Glass. The kappa statistic: A second look. *Computational Linguistics*, 30(1):95–101, 2004. 73

[EHK+06] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-Time Volume Graphics*. A K Peters, Ltd., 2006. 27

[EKE08] M. Eissele, M. Kreiser, and T. Ertl. Context-controlled flow visualization in augmented reality. In *Proceedings of Graphics Interface*, pages 89–96, 2008. 116

[ESE09] M. Eissele, H. Sanftmann, and T. Ertl. Interactively refining object-recognition system. *Journal of WSCG*, 17(1-3):1–8, 2009. 114

[FALT83] M. Forina, C. Armanino, S. Lanteri, and E. Tiscornia. Classification of olive oils from their fatty acid composition. In H. Martens and H. Russwurm, jr., editors, *Food Research and Data Analysis*, pages 189–214. Applied Science Publishers, London, 1983. 53, 66

[Fek04] J.-D. Fekete. The infovis toolkit. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 167–174, 2004. 109

[FFT74] M. A. Fisherkeller, J. H. Friedman, and J. W. Tukey. PRIM-9: An interactive multidimensional data display and analysis system. In *Proceedings of the Pacific ACM Regional Conference*, 1974. 37, 38, 39

[FL95] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the ACM SIGMOD Conference*, pages 163–174, 1995. 33, 77

[Fly72] M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21(9):948–960, 1972. 105

[Gab71] K. R. Gabriel. The biplot graphic display of matrices with application to principal component analysis. *Biometrika*, 58(3):453–467, 1971. 40, 51

[GE03] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003. 57, 58

[GNRM08] S. Garg, J. Nam, I. Ramakrishnan, and K. Mueller. Model-driven visual analytics. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 19–26, 2008. 69

[GPK+09] S. Georgieva, R. Peeters, H. Kolster, J. T. Todd, and G. A. Orban. The processing of three-dimensional shape from disparity in the human brain. *The Journal of Neuroscience*, 29(3):727–742, 2009. 8

[HAC01] C. G. Healey, R. S. Amant, and J. Chang. Assisted visualization of e-commerce auction agents. In *Proceedings of Graphics Interface*, pages 201–208, 2001. 18

[Häg70] T. Hägerstrand. What about people in regional science? *Papers in Regional Science*, 24:6–21, 1970. 12

[Ham06] L. Hamel. Visualization of support vector machines with unsupervised learning. In *Proceedings of the IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, pages 1–8, 2006. 59

[HCL05] J. Heer, S. K. Card, and J. A. Landay. Prefuse: A toolkit for interactive information visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 421–430, 2005. 109

[HGM+97] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley. DNA visual and analytic data mining. In *Proceedings of the IEEE Conference on Visualization*, pages 437–441, 1997. 12, 40

[HLE04] M. Hopf, M. Luttenberger, and T. Ertl. Hierarchical splatting of scattered 4D data. *IEEE Computer Graphics and Applications*, 24(4):64–72, 2004. 21

[HM90] R. B. Haber and D. A. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In G. M. Nielson, B. Shriver, and L. J. Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–93. IEEE Computer Society Press, Silver Spring, 1990. 105

[HR07] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, 2007. 3, 40

[IY05] I. Ideses and L. Yaroslavsky. Three methods that improve the visual quality of colour anaglyphs. *Journal of Optics A: Pure and Applied Optics*, 7:755–762, 2005. 87

[JNG04] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pages 593–598, 2004. 75

[JS91] B. Johnson and B. Shneiderman. Tree-Maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the IEEE Conference on Visualization*, pages 284–291, 1991. 11

[Kan01] E. Kandogan. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pages 107–116, 2001. 40

[KB04] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28:801–814, 2004. 21

[KC04] Y. Koren and L. Carmel. Robust linear dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):459–470, 2004. 77

[KDA+09] P. Kristensson, N. Dahlback, D. Anundi, M. Bjornstad, H. Gillberg, J. Haraldsson, I. Martensson, M. Nordvall, and J. Stahl. An evaluation of space time cube representation of spatiotemporal patterns. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):696–702, 2009. 12

[KEGB03] Z. Kourtzi, M. Erb, W. Grodd, and H. H. Bülthoff. Representation of the perceived 3-D object shape in the human lateral occipital complex. *Cerebral Cortex*, 13(9):911–920, 2003. 10

[KKH02] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002. 33

[KMS+08] D. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. In S. Simoff, M. Böhlen, and A. Mazeika, editors, *Visual Data Mining*, volume 4404 of *Lecture Notes in Computer Science*, pages 76–90. Springer Berlin / Heidelberg, 2008. 69

[KRC02] G. Kindlmann, E. Reinhard, and S. Creem. Face-based luminance matching for perceptual colormap generation. In *Proceedings of the IEEE Conference on Visualization*, pages 299–306, 2002. 29, 89

[KSFE10] D. Kauker, H. Sanftmann, S. Frey, and T. Ertl. Memory saving discrete Fourier transform on GPUs. In *Proceedings of the IEEE International Conference on Computer and Information Technology*, pages 1152–1157, 2010. 106

[KSG07] H. Kang, V. Sehgal, and L. Getoor. GeoDDupe: A novel interface for interactive entity resolution in geospatial data. In *Proceedings of the Conference on Information Visualisation*, pages 489–496, 2007. 70

[KSH04] R. Kosara, G. N. Sahling, and H. Hauser. Linking scientific and information visualization with interactive 3D scatterplots. In *Proceedings of the Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 133–140, 2004. 18, 20, 30

[KSJ95] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Essentials of Neural Science and Behavior*. Appleton & Lange, 1995. 29

[KSJ00] E. Kandel, J. Schwartz, and T. Jessell. *Principles of Neural Science*. McGraw-Hill Medical, fourth edition, 2000. 7, 9, 10, 41

[KW99] G. Kindlmann and D. Weinstein. Hue-balls and lit-tensors for direct volume rendering of diffusion tensor fields. In *Proceedings of the IEEE Conference on Visualization*, pages 183–189, 1999. 21

[LBCS10] C. Lübbe, A. Brodt, N. Cipriani, and H. Sanftmann. NexusVIS: A distributed visualization toolkit for mobile applications. In *Proceedings of the IEEE Pervasive Computing and Communications Workshops*, pages 841–843, 2010. Demonstration. 106

[Lev66] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. 76

[LG03] Q. Lu and L. Getoor. Link-based classification. In *Proceedings of the International Conference on Machine Learning*, pages 496–503, 2003. 75

[Lia99] S. Liang. *Java Native Interface: Programmer's Guide and Reference*. Addison-Wesley Longman Publishing Co., Inc., 1999. 114

[Lob09] R. Lobel. Magenta-cyan anaglyphs. `http://www.divideconcept.net/papers/MCA-RL09.pdf`, 2009. Last accessed on 8 April 2011. 87

[LVLRR08] L. Linsen, T. Van Long, P. Rosenthal, and S. Rosswog. Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1483–1490, 2008. 18

[LWRS08] M. Lohse, F. Winter, M. Repplinger, and P. Slusallek. Network-integrated multimedia middleware (NMM). In *Proceedings of ACM Multimedia*, pages 1081–1084, 2008. 109

[Mac86] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986. 18

[May] M. Mayer. A world in motion. `http://googleblog.blogspot.com/2007/03/world-in-motion.html`. Last accessed on 7 October 2011. 3

[MCEF94] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32, 1994. 108

[ME04] B. Mora and D. S. Ebert. Instant volumetric understanding with order-independent volume rendering. *Computer Graphics Forum*, 23(3):489–497, 2004. 27

[MHJ99] M. Miller, C. Hansen, and C. Johnson. The SCIRun problem solving environment: Implementation within a distributed environment. In *Proceedings of the SIAM Conference on Parallel Processing for Scientific Computing*, 1999. extended abstract. 108

[Min69] C. J. Minard. Carte figurative des pertes successives en hommes de l'armée Française dans la campagne de Russie, 1812–1813, 1869. 12, 60

[MKO+02] S. O. Murray, D. Kersten, B. A. Olshausen, P. Schrater, and D. L. Woods. Shape perception reduces activity in human primary visual cortex. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 99, pages 15164–15169, 2002. 10

[MS99] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The M.I.T. Press, 1999. 4, 57

[MT03] K. Moreland and D. Thompson. From cluster to wall with VTK. In *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, pages 25–31, 2003. 108

[Mul85] K. T. Mullen. The contrast sensitivity of human colour vision to red-green and blue-yellow chromatic gratings. *The Journal of Physiology*, 359:381–400, 1985. 86

[MW10] M. Migut and M. Worring. Visual exploration of classification models for risk assessment. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pages 11–18, 2010. 59

[MZS10] D. F. McAllister, Y. Zhou, and S. Sullivan. Methods for computing color anaglyphs. In *Proceedings of SPIE 7524*, pages 75240S–75240S–12, 2010. 87

[Nas95] G. Nason. Three-dimensional projection pursuit. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 44(4):pp. 411–430, 1995. 40

[NW72] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384, 1972. 49

[OHJS10] P. Oesterling, C. Heine, H. Jänicke, and G. Scheuermann. Visual analysis of high dimensional point clouds using topological landscapes. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 113–120, 2010. 12

[OUW+06] Y. Okada, K. Ukai, J. S. Wolffsohn, B. Gilmartin, A. Iijima, and T. Bando. Target spatial frequency determines the response to conflicting defocus- and convergence-driven accommodative stimuli. *Vision Research*, 46(4):475–484, 2006. 87

[Par] ParaView. `http://www.paraview.org`. Last accessed on 7 October 2011. 108

[PD08] F. Poulet and T.-N. Do. Interactive decision tree construction for interval and taxonomical data. In S. Simoff, M. Böhlen, and A. Mazeika, editors, *Visual Data Mining*, volume 4404 of *Lecture Notes in Computer Science*, pages 123–135. Springer Berlin / Heidelberg, 2008. 59

[PDT+10] J. Porrill, P. A. Duke, N. A. Taroyan, J. P. Frisby, and D. Buckley. The accuracy of metric judgements: Perception of surface normal. *Vision Research*, 50(12):1140–1157, 2010. 9

[PEP+11] J. Poco, R. Etemadpour, F. Paulovich, T. Long, P. Rosenthal, M. Oliveira, L. Linsen, and R. Minghim. A framework for exploring multidimensional data with 3D projections. *Computer Graphics Forum*, 30(3):1111–1120, 2011. 17, 18

[PKH04] H. Piringer, R. Kosara, and H. Hauser. Interactive focus+context visualization with linked 2D/3D scatterplots. In *Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 49–60, 2004. 18, 20, 30

[PNML08] F. Paulovich, L. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):564–575, 2008. 18

[Pou04] F. Poulet. SVM and graphical algorithms: a cooperative approach. In *Proceedings of the IEEE International Conference on Data Mining*, pages 499–502, 2004. 59

[PS88] Z. W. Pylyshyn and R. W. Storm. Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial Vision*, 3(3):179–197, 1988. 41

[PXYH05] D. Phan, L. Xiao, R. Yeh, and P. Hanrahan. Flow map layout. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 219–224, 2005. 60

[Qui92] J. R. Quinlan. Learning with continuous classes. In *Proceedings of the Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992. 4, 74

[RE04] G. Reina and T. Ertl. Volume visualization and visual queries for large high-dimensional datasets. In *Proceedings of the Eurographics/IEEE VGTC Symposium on Visualization*, pages 255–260, 2004. 17

[RFF+08] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of animation in trend visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332, 2008. 3

[RM05] L. Rokach and O. Maimon. Top-down induction of decision trees classifiers - a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(4):476–487, 2005. 57

[RMC91] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone Trees: Animated 3D visualizations of hierarchical information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology*, pages 189–194, 1991. 11

[Rol53] W. Rollmann. Zwei neue stereoskopische Methoden. *Annalen der Physik und Chemie*, 90:186–187, 1853. 85

[ROPT05] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen. ContextPhone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, 4(2):51–59, 2005. 107

[RSK08] M. Robnik-Sikonja and I. Kononenko. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):589–600, 2008. 69

[SAW94] B. N. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994. 106

[SBG00] T. Sprenger, R. Brunella, and M. Gross. H-BLOB: A hierarchical visual clustering method using implicit surfaces. In *Proceedings of the IEEE Conference on Visualization*, pages 61–68, 2000. 78

[SBS05] O. Schall, A. Belyaev, and H.-P. Seidel. Robust filtering of noisy scattered point data. In *Proceedings of the Eurographics/IEEE VGTC Symposium of Point-Based Graphics*, pages 71–144, 2005. 21

[SBSW09] H. Sanftmann, A. Blessing, H. Schütze, and D. Weiskopf. Visual exploration of classifiers for hybrid textual and geospatial matching. In *Proceedings of the Vision, Modeling, and Visualization Conference*, pages 245–253, 2009. 5, 68

[SCB98] D. F. Swayne, D. Cook, and A. Buja. XGobi: Interactive dynamic data visualization in the X Window System. *Journal of Computational and Graphical Statistics*, 7(1):113–130, 1998. 38

[SCL+99] M. M. Sebrechts, J. V. Cugini, S. J. Laskowski, J. Vasilakis, and M. S. Miller. Visualization of search results: a comparative evaluation of text, 2D, and 3D interfaces. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval*, pages 3–10, 1999. 11

[SCW11] H. Sanftmann, N. Cipriani, and D. Weiskopf. Distributed context-aware visualization. In *Proceedings of the IEEE Pervasive Computing and Communications Workshops*, pages 251–256, 2011. 5, 106

[SGV06] V. Sehgal, L. Getoor, and P. Viechnicki. Entity resolution in geospatial data integration. In *Proceedings of the ACM Symposium on Advances in Geographic Information Systems*, pages 83–90, 2006. 69

[Sha02] G. Sharma. LCDs versus CRTs-color-calibration and gamut considerations. *Proceedings of the IEEE*, 90(4):605–622, 2002. 98

[SHF+06] F. Shibata, T. Hashimoto, K. Furuno, A. Kimura, and H. Tamura. Scalable architecture and content description language for mobile mixed reality systems. In *Proceedings of the International Conference on Artificial Reality and Telexistence (ICAT)*, volume 4282 of *Lecture Notes in Computer Science*, pages 122–131. Springer Berlin / Heidelberg, 2006. 108

[SHS04] S. E. B. Sorensen, P. S. Hansen, and N. L. Sorensen. Method for recording and viewing stereoscopic images in color using multichrome filters, United States Patent US 6,687,003, issued April 26, 2004. 87

[Sie96] M. Siegrist. The use or misuse of three-dimensional graphs to represent lower-dimensional data. *Behaviour & Information Technology*, 15(2):96–100, 1996. 11

[SJCSO01] M. St. John, M. B. Cowen, H. S. Smallman, and H. M. Oonk. The use of 2D and 3D displays for shape-understanding versus relative-position tasks. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 43(1):79–98, 2001. 13

[SL91] S. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):660–674, 1991. 57

[SLBC03] D. F. Swayne, D. T. Lang, A. Buja, and D. Cook. GGobi: Evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43(4):423–444, 2003. 38, 39

[SLC90] P. H. Schiller, N. K. Logothetis, and E. R. Charles. Role of the color-opponent and broad-band channels in vision. *Visual Neuroscience*, 5(4):321–346, 1990. 9

[SM99] S. Scott and S. Matwin. Feature engineering for text classification. In *Proceedings of International Conference on Machine Learning*, pages 379–388, 1999. 58

[SM03] W. Sanders and D. F. McAllister. Producing anaglyphs from synthetic images. In *Proceedings of SPIE 5006*, pages 348–358, 2003. 87

[SML03] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object Oriented Approach to 3D Graphics*. Kitware, Inc., 3rd edition, 2003. 108

[SNB+08] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. Technical Report CS-TR-4905, University of Maryland, College Park, 2008. 71

[SS10] M. Suto and D. Sykes. StereoPhoto Maker. `http://stereo.jpn.org/eng/stphmkr/`, 2010. Last accessed on 3 November 2011. 88

[SSJOC01] H. Smallman, M. St. John, H. Oonk, and M. Cowen. Information availability in 2D and 3D displays. *IEEE Computer Graphics and Applications*, 21(5):51–57, 2001. 11

[Ste46] S. S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, 1946. 38

[STH02] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE*

Transactions on Visualization and Computer Graphics, 8(1):52–65, 2002. 38

[SW09] H. Sanftmann and D. Weiskopf. Illuminated 3D scatterplots. *Computer Graphics Forum*, 28(3):751–758, 2009. 4, 5, 19

[SW11] H. Sanftmann and D. Weiskopf. Anaglyph stereo without ghosting. *Computer Graphics Forum*, 30(4):1251–1259, 2011. 5, 86

[SW12] H. Sanftmann and D. Weiskopf. 3D scatterplot navigation. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1969–1978, 2012. 4, 5, 19, 37

[TC06] J. Thomas and K. Cook. A visual analytics agenda. *IEEE Computer Graphics and Applications*, 26(1):10–13, 2006. 69

[TCM06] M. Tarini, P. Cignoni, and C. Montani. Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1237–1244, 2006. 30

[TFO09] S. Takahashi, I. Fujishiro, and M. Okada. Applying manifold learning to plotting approximate contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1185–1192, 2009. 17

[TH07] R. Tibshirani and T. Hastie. Margin trees for high-dimensional classification. *The Journal of Machine Learning Research*, 8:637–652, 2007. 78

[TKAM06] M. Tory, A. Kirkpatrick, M. Atkins, and T. Möller. Visualization task performance with 2D, 3D, and combination displays. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):2–13, 2006. 13

[TM03] S. T. Teoh and K.-L. Ma. PaintingClass: Interactive construction, visualization and exploration of decision trees. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pages 667–672, 2003. 59

[TM04] M. Tory and T. Möller. Rethinking visualization: A high-level taxonomy. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 151–158, 2004. 1

[TMB02] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, 2002. 40

[TMWK99] J. R. Tresilian, M. Mon-Williams, and B. M. Kelly. Increasing confidence in vergence as a cue to distance. *Proceedings of Biological Sciences*, 266(1414):39–44, 1999. 8

[TN03] J. Todd and J. Norman. The visual perception of 3-D shape from multiple cues: Are observers capable of perceiving metric structure? *Attention, Perception, & Psychophysics*, 65:31–47, 2003. 9

[Tod04] J. Todd. The visual perception of 3D shape. *Trends in Cognitive Sciences*, 8(3):115–121, 2004. 9, 41

[Tor03] M. Tory. Mental registration of 2D and 3D visualizations (an empirical study). In *Proceedings of the IEEE Conference on Visualization*, pages 371–378, 2003. 13, 50

[Tre] Trendanalyzer. `http://www.gapminder.org`. Last accessed on 7 October 2011. 3

[TWA11] I. Tsirlin, L. Wilcox, and R. Allison. The effect of crosstalk on the perceived depth from disparity and monocular occlusions. *IEEE Transactions on Broadcasting*, 57(2):445–453, 2011. 85

[UFK+89] C. Upson, J. Faulhaber, T.A., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The Application Visualization System: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, 1989. 108

[Uka06] K. Ukai. Human factors for stereoscopic images. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1697–1700, 2006. 87

[Ull79] S. Ullman. The interpretation of structure from motion. In *Proceedings of the Royal Society of London*, volume 203 of *B. Biological Sciences*, pages 405–426, 1979. 41

[Urb08] S. Urbanek. Visualizing trees and forests. In C.-h. Chen, W. Härdle, and A. Unwin, editors, *Handbook of data visualization*, chapter 10, pages 243–264. Springer Berlin / Heidelberg, 2008. 60, 78

[vdEvW11] S. van den Elzen and J. J. van Wijk. Baobab view: Interactive construction and analysis of decision trees. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, 2011. 60

[VFP+02] W. Vanduffel, D. Fize, H. Peuskens, K. Denys, S. Sunaert, J. T. Todd, and G. A. Orban. Extracting 3D from motion: Differences in human and monkey intraparietal cortex. *Science*, 298(5592):413–415, 2002. 10

[vW05] J. van Wijk. The value of visualization. In *Proceedings of the IEEE Conference on Visualization*, pages 79–86, 2005. 17, 69

[vWvdW99] J. van Wijk and H. van de Wetering. Cushion treemaps: visualization of hierarchical information. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 73–78, 147, 1999. 11

[War94] M. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. In *Proceedings of the IEEE Conference on Visualization*, pages 326–333, 1994. 38

[War04] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, second edition, 2004. 7, 10, 29, 39, 101

[War08] C. Ware. *Visual Thinking for Design*. Morgan Kaufmann, 2008. 8, 41

[WB97] P. C. Wong and R. D. Bergeron. 30 years of multidimensional multivariate visualization. In G. M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization: Overviews, Methodologies, and Techniques*, pages 3–33. IEEE Computer Society, Los Alamitos, 1997. 1, 3

[WB08] C. Weigle and D. Banks. A comparison of the perceptual benefits of linear perspective and physically-based illumination for display of dense 3D streamtubes. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1723–1730, 2008. 41

[WBP07] G. Weber, P.-T. Bremer, and V. Pascucci. Topological landscapes: A terrain metaphor for scientific data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1416–1423, 2007. 12

[WE05] D. Weiskopf and G. Erlebacher. Overview of flow visualization. In C. D. Hansen and C. R. Johnson, editors, *The Visualization Handbook*, pages 261–278. Elsevier, 2005. 114

[WF71] H. Wallach and L. Floor. The use of size matching to demonstrate the effectiveness of accommodation and convergence as cues for distance. *Attention, Perception, & Psychophysics*, 10:423–428, 1971. 8

[WFG92] L. Wanger, J. Ferwerda, and D. Greenberg. Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics and Applications*, 12(3):44–58, 1992. 7

[WGP98] C. Ware, C. Gobrecht, and M. Paton. Dynamic adjustment of stereo display parameters. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 28(1):56–65, 1998. 8

[WH10] A. J. Woods and C. R. Harris. Comparing levels of crosstalk with red/cyan, blue/yellow, and green/magenta anaglyph 3D glasses. In *Proceedings of SPIE Stereoscopic Displays and Applications*, volume 7524, page 75240Q, 2010. 87

[Whe38] C. Wheatstone. Contributions to the physiology of vision.–Part the first. On some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London*, 128:371–394, 1838. 85

[Wie06] W. Wieser. Anaglyph glasses transmission spectra. `http://www.triplespark.net/render/stereo/anaglyph/glasses/`, 2006. Last accessed on 3 November 2011. 98

[Wil99] L. Wilkinson. Dot plots. *The American Statistician*, 53:276–281, 1999. 12

[WK00] S. Wilson and J. Kesselman. *Java(TM) Platform Performance: Strategies and Tactics*. Prentice Hall PTR, 2000. 113

[Woo10] A. Woods. Understanding crosstalk in stereoscopic displays. Keynote Presentation at the Conference on Three Dimensional Systems and Applications, 2010. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.165.5355`. 87

[WPG+97] C.-F. Westin, S. Peled, H. Gudbjartsson, R. Kikinis, and F. A. Jolesz. Geometrical diffusion measures for MRI from tensor basis analysis. In *Proceedings of ISMRM*, page 1742, 1997. 21, 22

[WT02] A. J. Woods and S. S. L. Tan. Characterising sources of ghosting in time-sequential stereoscopic video displays. In *Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems*, pages 66–77, 2002. 87, 98

[WvdBLK01] S. Winkler, C. J. van den Branden Lambrecht, and M. Kunt. Vision and video: Models and applications. In C. J. van den Branden Lambrecht, editor, *Vision Models and Applications to Image and Video Processing*, chapter 10, pages 201–231. Springer, 2001. 86

[WYK07] A. J. Woods, K. L. Yuen, and K. S. Karvinen. Characterizing crosstalk in anaglyphic stereoscopic images on LCD monitors and plasma displays. *Journal of the Society for Information Display*, 15(11):889–898, 2007. 87

[XECA07] X. Xiong, H. G. Elmongui, X. Chai, and W. G. Aref. PLACE*: A distributed spatio-temporal data stream management system for moving objects. In *Proceedings of the Conference on Mobile Data Management*, pages 44–51, 2007. 108

[Yan99] L. Yang. 3D grand tour for multidimensional data and clusters. In *Proceedings of the Symposium on Advances in Intelligent Data Analysis*, volume 1642 of *Lecture Notes in Computer Science*, pages 173–184. Springer, Berlin / Heidelberg, 1999. 5, 18, 40, 85

[ZSH96] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive visualization of 3D-vector fields using illuminated stream lines. In *Proceedings of the IEEE Conference on Visualization*, pages 107–113, 1996. 21, 24