CAGD and Scientific Visualization

Habilitationsschrift

zur Erlangung des akademischen Grades

Dr. Ing. habil.

an der Fakultät der Ingenieurwissenschaften
der Universität Rostock

vorgelegt von
Holger Theisel
geb. am 16.08.1969 in Jena

Rostock, 1. Juli 2001

Gutachter:
Prof. Dr. Heidrun Schumann (Universität Rostock)
Prof. Dr. Gerald Farin (Arizona State University, Tempe, USA)
Prof. Dr. Thomas Ertl (Universität Stuttgart)

Abstract

In this work we investigate the correlation between the two disciplines CAGD and Scientific Visualization. Both are correlated to Computer Graphics but became disciplines of their own and therefore developed rather independently of each other.

It is the task of this work to analyze the interapplicability of both disciplines to each other, to find where ideas and methods of one discipline can be applied to the other, and vice versa.

In the first part we analyze both disciplines concerning their histories, the present data, and their pipelines. Doing so we obtain concrete expectations on where exactly it makes sense to search for interapplications.

Based on these expectations, part two of this work systematically investigates applications of CAGD ideas and methods in Scientific Visualization. Doing so, we do not only collect and systematize existing approaches, we also develop a number of new techniques.

Part three of this work deals with the application of of Scientific Visualization in the design process of curves and surfaces. Here we also provide a number of new approaches in addition to the systematization of the existing ones.

Acknowledgements

The work described here has been carried out during my stay as a research and teaching assistant at the Institute of Computer Graphics of the University of Rostock (Germany) from 1996 to 2001. First of all, I would like to thank my advisor Prof. Heidrun Schumann for her constant support and encouragement. She was able to give me both the necessary scientific freedom and her sound advice when I needed it. Her style of scientific working and her style of leading a research group deeply impressed and influenced me. I have regarded it as a pleasure and an honor to work under her supervision.

I would also like to thank the members the Institute of Computer Graphics of the University of Rostock under the leadership of Prof. Dietmar Jackél. In this group I found a creative atmosphere, inspiration and help in the many little problems which appear every day. In particular I'd like to thank my former colleagues and students Tino Weinkauf, Uwe Rauschenbach, Carlos Pérez Risquet, Matthias Kreuseler and Martin Köller for their collaboration with me which results are partially reported in this work.

Ian Harknett deserves a special thank you for the English proofreading of this work.

Finally I would like to thank my wife Daily Bravo Rangel. Although most of the time of carrying out this work we were not able to stay with each other, I always felt you on my side. Your love gave me the strength and patience to finish this work.

Rostock, July 1^{st}, 2001
Holger Theisel

Contents

Chapter 1

Introduction

Modern Computer Graphics shows more and more a trend of interdisciplinary working methods. Today's real-life problems are of such complexity that a specialist in one discipline cannot solve them alone. Instead, teams of experts have to work on the problems. For example, modern approaches in Data Mining contain approaches of Data bases, Scientific Visualization, Modeling/Simulation, and Statistics. Also within the field of Computer Graphics the disciplines come closer to each other to solve complex and general problems. This collaboration with other disciplines can be fruitful for a particular discipline, because this way a discipline is considered in the light of other disciplines, can contribute to other disciplines, or can apply and improve their results.

In this context it may occasionally turn out that different disciplines have worked on similar problems and found similar solutions independently of each other. Moreover, the application of one discipline to another one might yield significant new results there. Thus the investigation of correlations, dependencies, and mutual influence between different disciplines is a promising approach to develop them further.

It is the purpose of this work to investigate the correlations and mutual influence of two disciplines which are related to Computer Graphics: CAGD (Computer Aided Geometric Design) and Scientific Visualization. We want to investigate where to apply the ideas and methods of one discipline to the other in order to find improvements for both disciplines.

The main purpose of *Scientific Visualization* is to produce visual representations of large data sets. The task is to explore data and information in such a way as to gain understanding and inside into the data. Modern data sources like satellites or CT (Computer Tomography) devices produce daily a high amount of data which has to be analyzed. In fact, data sets of the size of Gigabytes or even Terabytes are nowadays common. To explore these large data sets, visual analysis is a promising approach. The general reason for this is the fact that the human eye is able to recognize a high amount information in a single moment. Scientific Visualization tries to make use of this ability by providing appropriate visual representations of the data and leaving the interpretations of the images to the intelligence of a human being. This way Scientific Visualization appears in a row with other data analysis tools which mostly come from the area of

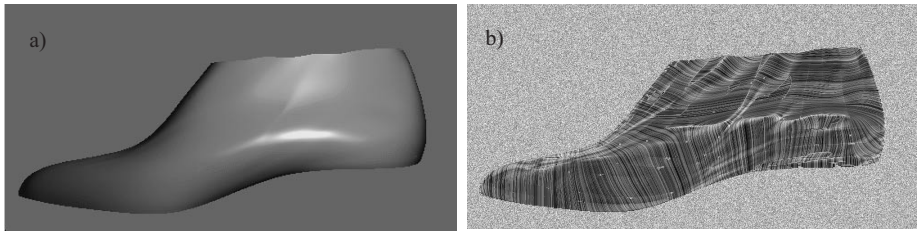


Figure 1.1: a) piecewise bicubic B-spline surface; b) visualizing one class of lines of curvature using methods of flow visualization.

statistics. For complex problems a rather common approach is to combine a number of analyzing tools. Since there are a lot of different kinds of data to be visualized, and since there are a number of different goals and motivations for applying visualization, a variety of approaches and techniques to Scientific Visualization exist, and it is crucial to choose an appropriate one.

While Scientific Visualization is an approach to analyzing data, *CAGD* can mainly be considered as an approach to creating data by design. Here the data to be designed are curves and surfaces. Curves/surfaces and their properties are well-known for a long time. It is the task of CAGD to find representations of curves/surfaces which are useful for design purpose. These representations should be

- intuitive. There should be an intuitive relation between the representation and the curve/surface.
- simple. The designer should be confronted with as few degrees of freedom as necessary to design the curve/surface.
- flexible. The designer should be able to design virtually every curve/ surface which he/she has in mind.

It turned out that the representations which fulfill this task best are rather simple networks of control points which can interactively moved by the designer. CAGD explores the theory behind these networks of control points and thus makes designed curves/surfaces applicable to a variety of areas.

Since there are a variety of different motivations and tasks to design curves/ surfaces, there are a number of different curve/surface schemes in CAGD which emphasize different aspects and properties of the curves/surfaces. Nevertheless, most applications focus on a particular class of curves/ surfaces which have thus become a quasi-standard: the class of Bézier- or B-spline curves/surfaces.

It is the purpose of this work to explore the correlations of both disciplines: Scientific Visualization and CAGD. We want to investigate where ideas, methods and concepts of one discipline can be applied to the other, and vice versa. Doing so we show that this approach gives new contributions to both disciplines. To illustrate the main idea of this work, we consider two examples.

Figure 1.1a shows a piecewise bicubic B-spline surface, a standard surface class in CAGD. To evaluate the quality of this designed surface, one way is

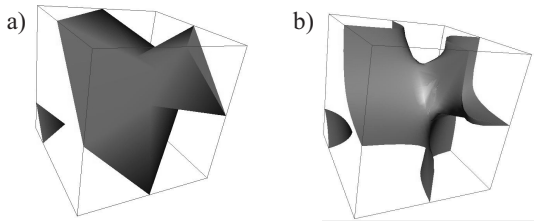


Figure 1.2: a) piecewise triangular approximation of an isosurface of a trilinear volume data set using Marching Cubes; b) computation of the exact isosurface as trimmed piecewise rational cubic surface.

to consider the lines of curvature[1] on the surface. Although lines of curvature on a surface give valuable information about the surface behavior, they cannot be described in a closed form as families of parametric curves on the surfaces. However, it can be shown that they can be interpreted as tangent curves of a certain vector field on the surface. Hence a variety of techniques from Scientific Visualization (in particular from flow visualization) exist and can be used to visualized the lines of curvature for the CAGD process. Figure 1.1b shows an example of applying a visualization technique[2] which makes the behavior of the lines of curvature visible. This is an example of how to apply a technique from Scientific Visualization to CAGD.

To show an example of applying CAGD methods to Scientific Visualization, consider figure 1.2. Figure 1.2a shows a common approach for an isosurface extraction in volume visualization[3]. If this approximation is too rough, the correct isosurface can be described as trimmed piecewise rational cubic surface. Figure 1.2b illustrates this for the example in figure 1.2a. Hence, this is an example of applying CAGD methods to improve the results of Scientific Visualization.

Nowadays both disciplines Scientific Visualization and CAGD are rather complex and heterogeneous. In both disciplines a lot of different methods and approaches have been developed. Moreover, both disciplines have collaborations with other disciplines in science and technology. Thus it cannot be the purpose of this work to give a complete overview over the state of the art in both disciplines. Excellent surveys exist for both disciplines ([55], [95] for CAGD, [167] for Scientific Visualization). Instead, this work only deals with those aspects of both disciplines where a collaboration with the other discipline makes sense.

To be able to investigate the correlations between CAGD and Scientific Visualization in a systematic way, as a first step we have to choose the points of potential correlations between the disciplines inside themselves. To do so, we have to answer the following specific questions:

[1] Lines of curvature are surface curves which have the defining property of being tangential with one of the principal directions of the surface in every point of the curve. They form two families of surface curves which reflect geometric properties of the surface. See [55] for details.

[2] The visualization technique applied here is called Integrate & Draw and is explained in section 4.3.3.2 of this work.

[3] Volume visualization is a part of Scientific Visualization, as we will see later in this work. The technique to extract the triangular approximation of the isosurface is called Marching Cubes ([130]) which is treated in section 3.5.2.2 of this work.

1. Which concepts and methods of CAGD are good candidates to be applied in Scientific Visualization?

2. Which concepts and methods of Scientific Visualization can be applied to CAGD?

To answer question 1, we see the following applications:

- Since the human eye reacts rather sensitively to slight perturbations of curves or surfaces[4], these are promising candidates to encode a high amount of data. Scientific Visualization needs methods to encode large data sets intuitively. Hence curves/surfaces are promising geometric objects to be applied in Scientific Visualization.

- CAGD has gathered knowledge and experience about smooth interpolation and approximation, especially for curves and surfaces. This way, rather complicated interpolation conditions can be expressed in terms of simple geometric correlations of control points. In Scientific Visualization smooth interpolations and approximations which preserve certain properties play an important role. Hence we may use the knowledge of CAGD about simple interpolation schemes to find simple and robust interpolations in different parts of the visualization process.

- CAGD is basically concerned with designing data while it is the task of Scientific Visualization to find appropriate visual representations of given data sets. Although most of the data treated in Visualization comes from measurement or simulation, data designed by methods similar to CAGD are also of interest. Especially when this data has to be compared with measured data, or if certain techniques in Scientific Visualization have to be evaluated, the controlled design of input data – and thus the application of ideas and concepts of CAGD – may be of interest for Scientific Visualization.

To answer question 2, we see the following applications:

- At certain levels of the design process in CAGD, the designer has to deal with rather large amounts of data. Based on this data the designer has to make decisions concerning the further steps in the CAGD process. (For example, the designer might have to choose some parameters or decide if a complete redesign of parts of the curve/surface is necessary.) Hence it might be worth trying to find a visualization of the present data and current results of the design process in order to prepare and choose the next design steps.

In order to investigate the applicability of CAGD methods in Scientific Visualization and vice versa, the rest of this work is divided into three parts. Part one analyzes CAGD and Scientific Visualization in the light of possible applications to each other. This part consists of chapter 2. Based on the results of this analysis, part two treats the applications of CAGD methods in Scientific Visualization. This part consists of the chapters 3–6. Finally part three treats the

[4]This is a fact which can be confirmed by any owner of a car: he/she will detect a small bump in the surface of a car immediately, even if it is a very small one!

applications of Scientific Visualization in CAGD. This part consists of chapter 7 .

Exploring the interapplicability of CAGD and Scientific Visualization, this work intends both to collect and systematize existing methods and to present new approaches of applying one discipline to the other. Since it turned out that a rather high number of interapplications already exists, this work has to incorporate two styles of presentation. The collection and systematization of existing methods is done in a survey-like style where readers are referred to the original papers for details. On the other hand, detailed descriptions are given for new approaches which are suggested by the author, and some of which are previously unpublished.

Chapter 2

Analysis and Comparison of Visualization and CAGD

In recent years both Scientific Visualization and CAGD have become rather complex and consist of a variety of different aspects and facets. In order to apply one discipline to the other, this complexity of the disciplines has to be reduced to those parts where an exchange and an application of the ideas of one discipline to the other gives hope for fruitful results. In other words: we have to analyze both disciplines concerning to their applicability to the other discipline. Doing this we formulate expectations about where the applications may succeed. These expectations yield the guide for the particular investigations of the applicability of the two discipline to each other.

In order to analyze CAGD and Scientific Visualization, and their correlations to each other, we compare both disciplines from three points of view. We compare the data which is considered in both disciplines (section 2.1), we compare the historical development of both disciplines (section 2.2), and we compare the pipelines of both the Visualization and CAGD processes (section 2.3).

2.1 Data Comparison of Visualization and CAGD

Scientific Visualization is concerned with a lot of different kinds of data. Hence there are a large number of approaches to describing and classifying this data. In CAGD, an explicit classification of the present data does not have the high importance it does in Scientific Visualization. So we use the following strategy in that section: among the existing approaches for a data description and classification in Scientific Visualization, we choose an appropriate one and try to describe the data which is present in CAGD in terms of this data classification.

To describe the data in Scientific Visualization, a number of schemes exist which emphasize different aspects of the data. Most of them agree in the assumption that the data consists of a number of values which are measured in a certain space. Hence the data in Scientific Visualization is mainly characterized by describing and classifying the values and the space where the values were obtained.

[17] considers the data as m-dimensional data on a k-dimensional grid using the notation L_m^k. This way k gives the dimensionality of the grid where the data is measured while m counts the number of values which are measured at a grid point. This approach to describing the data mainly focuses on an exact description of the underlying grid but does not allow us to give a further characterization of the present data types in a grid point.

In [210] a specification of scientific data is introduced which has been kept as simple as possible while still describing the most important properties of the data set. In fact, [210] only counts the dimensionality of the dimensions of the observation space and the number of measured values in a point of this space. This specification of scientific data was originally done to describe a particular subclass of scientific data (called multiparameter data, see chapter 5), but its ideas can generalized to describe general scientific data sets as well.

[70] introduces a data specification which focuses on an exhaustive and complete description of the data. There too, the underlying space and the data at the grid points of this space are specified, but a lot of additional and detailed information about the data set is given as well. The result is a quite complete and exact description of a data set. Unfortunately, the specification tends to become too complex and cluttered.

The specification we want to use here for our purposes comes from [23] and is a useful compromise between a compact and a complete description of the data. [23] considers scientific data to be a map from a certain domain into a certain range of values using the notation E_D^R. Both the domain D and the range of values R are then specified in detail. To describe D, [23] describes the dimensionality and the regions of validity for a measured data point. Three cases are distinguished: the data values are only valid at the observation points (written E_n^R), the data values are valid in certain areas around the observation point (written $E_{[n]}^R$), or the domain is an enumerated set (written $E_{\{n\}}^R$). In these notations, n describes the dimensionality of the domain.

To describe the range of values R, [23] distinguishes between the data types point (P), scalar (S), vector (V), and tensor (T). It describes $E_D^{2S\,1V_3}$ a data set where 2 scalar values and a 3D vector are measured at the points of the domain.

The description of [23] gives a compact description of both domain and range of values. The main disadvantage is the fact that the structure and the connectivity of the domain are not considered. Usually the data lies on a certain grid in the domain. To take this into consideration, we give additional verbal descriptions of the grid characteristics to the data specification of [23] when necessary.

Since its very beginning, Scientific Visualization has focused on a number of different data classes where the research and application of these data classes became quite independent of each other. The most important data classes in Scientific Visualization are:

- volume data: $E_{[3]}^S$ – (treated in chapter 3 of this work);
- 2D flow data: $E_{[2]}^{V_2}$ – (treated in chapter 4);
- 3D flow data: $E_{[3]}^{V_3}$ – (treated in chapter 4);

- multiparameter data: E_n^{mS} with $m \geq 2$ – (treated in chapter 5);
- 2D scattered data: $E_{[2]}^{S}$ with no connectivity of the sample points in the domain – (treated in section 6.1.1);
- 3D scattered data: $E_{[3]}^{P}$ with no connectivity of the sample points in the domain – (treated in section 6.1.2);
- 3D second order tensor data: $E_{[3]}^{T_{3,3}}$ – (treated in section 6.2).

Although virtually all combinations of the specification of domain and range of value are thinkable, the combinations collected above are the most relevant for Scientific Visualization.

In order to compare the data of Scientific Visualization and CAGD, we now try to insert the data which CAGD deals with into the data specification of [23]. We obtain:

- 2D parametric curves: $E_{[1]}^{V_2}$. The points of a 1-dimensional domain are mapped to 2D vectors which are interpreted as locus vectors.
- 3D parametric curves: $E_{[1]}^{V_3}$. The points of a 1-dimensional domain are mapped to 3D locus vectors.
- Parametric surfaces: $E_{[2]}^{V_3}$. The points of a 2-dimensional domain are mapped to 3D locus vectors.
- Control polygons of 2D curves: E_2^P and $E_{\{1\}}^S$. E_2^P describes a set of points in 2D where a linear connectivity is additionally assumed. $E_{\{1\}}^S$ describes the parameterization.
- Control polygons of 3D curves: E_3^P and $E_{\{1\}}^{2S}$. E_3^P describes a set of points in 3D where a certain regular connectivity (linear, triangular, rectangular) is additionally assumed. $E_{\{1\}}^{2S}$ describes the parameterization for a rectangular control point scheme. (If the scheme of control points has a liner connectivity, the parameterization is described by $E_{\{1\}}^S$. For a triangular connectivity of the control points, no parameterization may be necessary.)

Comparing this classification of CAGD data with the classification of the most relevant data in Scientific Visualization collected above, we can see that the CAGD data is not explicitly treated as a data class in Scientific Visualization. This leads us to

Expectation 1 *In CAGD there are currently only a few applications of Scientific Visualization because the data which is dealt with in CAGD is not explicitly considered by Scientific Visualization.*

Comparing the data specifications of CAGD and Scientific Visualization again, we can see that the data description of curves and surfaces come close to the data description of flow data (at least closer than to the other Scientific Visualization data classes mentioned above). For example, 3D flow data $E_{[3]}^{V_3}$ and surface data $E_{[2]}^{V_3}$ only differ in one dimension of the domain. So we can formulate

Expectation 2 *Due to similarities of the treated data, most of the applications of CAGD in Scientific Visualization can be expected in the visualization of flow data.*

2.2 History of Visualization and CAGD

In this section we want to analyze the correlation of CAGD and Scientific Visualization from a historical point of view. By analyzing and comparing both disciplines we derive more expectations on where to apply one discipline to the other. Section 2.2.1 outlines the history of Scientific Visualization, section 2.2.2 does so for CAGD. Section 2.2.3 compares the histories to give conclusions and expectations on the goal of this work.

2.2.1 History of Scientific Visualization

Visualization is not a new issue. Instead, centuries ago people tried to find appropriate visual representations of certain information. Euclid's "Elements" ([53]) uses drawings to represent and illustrate properties in geometry. In the Middle Ages astronomical maps appeared which used arrow plots[1] to visualize prevailing winds over the oceans. Height lines were used in topographical maps in the 18^{th} century. Early applications of isolines include representations of magnetic declinations on the earth surface (Halley, 1701) or investigations of temperature gradients on the northern hemisphere (von Humboldt, 1817). Other developments of early visualization approaches were motivated by the arts. Artists (especially painters) were always interested in finding appropriate visual representations of certain ideas or information. Already in 1637 Descartes formulated ([33]): "*Imagination or visualization, and in particular the use of diagrams, has a crucial part to play in scientific investigations*". This statement holds until today.

With the development of computer technology in the last century, visualization was confronted with new challenges and new possibilities which shifted visualization into a completely new quality. On the one hand the presence of computers caused a rapid growing of the data to be processed, making it impossible to find visual representations "by hand". On the other hand, the development of computers (and especially Computer Graphics) gave opportunities to create visual representations of larger data sets automatically by the computer. Hence early developments in visualization using computers were strongly connected to the development of Computer Graphics.

The time of birth of Scientific Visualization as a discipline of its own can be seen in 1987. In [35] the term "Visualization in Scientific Computing", now generally shortened to "Scientific Visualization", appeared for the first time. The developments of the following years were characterized by "the creation of an industry concerned with advanced scientific workstation hardware, software and networking. Conferences, journals, trade shows, videotapes, books, CD ROMs and networked communication to online digital libraries now abound, and indicate a healthy, growing research, development and technology-transfer environment" ([45]). In these times most of the algorithms and approaches were

[1] Arrow plots are still a frequently applied standard technique for flow visualization - see [167].

developed which became standard algorithms for the visualization of particular data classes. Examples are the Marching Cubes algorithm ([130]) for volume data in 1987, parallel coordinates ([100]) for multiparameter data, in 1987 as well, and Line Integral Convolution ([26]) for flow data in 1993[2].

Since 1990, the annual IEEE Visualization Conference has been held, which has managed to be the currently most important conference on Scientific Visualization. In Europe, the annual Eurographics Workshop on Scientific Visualization has been established since 1990. A variety of other journals, conferences, and workshops exist as well. Since 1987, a number of surveys and introductions to Scientific Visualization have been published. Most of them focus on the application of Visualization to particular data classes (for instance [210] for multiparameter data, [107] for volume data, or [152] for flow data). The reasons for this may be twofold:

- The different data classes in Scientific Visualization require quite a different treatment. In fact, inside the discipline "Scientific Visualization" the developments for the different data classes are rather independent of each other.

- A new discipline needs some years to gather the most important approaches, to find agreements in the community about important concepts and classifications. The discipline needs to collect material to be systematized. Only once this is done, do surveys or textbooks about the whole discipline have the chance to be widely accepted.

However, the first textbook about the whole discipline of Scientific Visualization seems to be [167] which appeared 13 years after Scientific Visualization was born as a discipline of its own. It gives a broad introduction into into all relevant aspects and data classes of visualization.

In recent years Scientific Visualization focuses on the application on very large data sets as well as on the steering and controlling of the visualization process. To do so, quality criteria[3] of a visualization technique have to be introduced and the particular visualization techniques have to be evaluated according to these criteria. The results are rule-based visualization systems which not only enable the scientist to visualize large data sets but also help him/her to produce appropriate and high quality visualizations.

Another trend of recent Scientific Visualization is the connection with other research areas. Already [35] formulated: "Visualization and Science go hand in hand as partners". For instance, the results of the visualization may be used to directly control the parameters of the modeling and simulation process which produced the data. This visualization scenario (already introduced under the name "interactive steering" in [85]) requires a level of computing power which is still too high for today's computing resources. Instead a focusing on particular issues has to been done ([120], [200]).

One particular approach of connecting Scientific Visualization with another discipline is treated in the work - the connection with CAGD.

[2] All these techniques are treated in detail later in this work.

[3] [167] introduces and explains the quality criteria expressivity, effectiveness, and suitability.

2.2.2 History of CAGD

Similar to Scientific Visualization, the roots of CAGD go back to Euclid and Descartes. Most of the bases of CAGD came from differential geometry and approximation theory. In fact, curves and surfaces have been a well-researched issue for a long time. With the appearance of computers, "only" their applicability to the design had to be discovered.

In the 50's computers were used to drive numerical controlled milling machines in automotive and aircraft industry. Also in shipbuilding people got interested in finding curve/surface schemes for the design. For this, a number of schemes have been tried.

In 1959 and 1963, de Casteljau ([40], [41]) and Bézier developed independently of each other the concept of Bézier curves and surfaces (which is strongly related to the de Casteljau algorithm). This scheme might be the most important development of the whole discipline. Nowadays the overwhelming majority of applications in CAGD are based on these concepts. Another approach that gained a rather high popularity were Coons patches ([34]) and Gordon surfaces.

B-spline curves and surfaces were introduced as design tools by de Boor ([39]) and Cox ([36]). Their extension to NURBS (non-uniform rational B-splines) builds nowadays the quasi-standard for free-form curves and surfaces. Apart of this, a variety of other spline curve schemes have been tried which were mainly motivated by approximation-theoretical approaches, i.e. they were the solutions of certain minimization problems.

The year 1974 can be considered to be the date of birth of CAGD as a discipline of its own. In a conference at the University of Utah the concept "CAGD" was used for the first time ([10]). In the following years the classic concepts of CAGD were applied in a variety of areas, such as CAD, geoscience, molecule design, pharmacy, architecture or simple word-processing. Of course, car and aircraft design was still a main application area of CAGD.

The first textbooks which cover the most relevant issues of the whole discipline appeared in 1988 ([55]) and 1989 ([95]), hence 14 and 15 years after CAGD was considered as a discipline of its own. Both textbooks have been updated and still set the standard in CAGD textbooks. Other textbooks which focus on certain aspects of CAGD are [150], [54], [158], [56] and [1] . A variety of conferences was held on CAGD. CAGD has its own journal at Elsevier.

In recent years CAGD has sought new application areas in science and industry. In addition, research into a number of unsolved problems is being carried out. Current research in CAGD includes the investigation of new curve/surface classes with special properties as well as research on still unsolved theoretical problems.

2.2.3 Historical comparison

Comparing the historical development of CAGD and Scientific Visualization, we can find a number of similarities:

- Both disciplines have roots which go back centuries; their development was mainly driven by the challenges of applying these roots to computer technology.

- The development of both disciplines is highly correlated to the development of Computer Graphics. In fact, in their early stages both disciplines were treated as parts of Computer Graphics before becoming disciplines of their own.
- In both disciplines the development of the most important algorithms was done quite soon after their "time of birth". On the other hand, in both disciplines it took more than 10 years until the first comprehensive textbooks appeared.
- Both disciplines are currently in a stage of systematization, search for new applications and for connections with other disciplines.

These points mentioned above give some of the motivations for the research described in this work.

However, the historical development of CAGD and Scientific Visualization also has a significant difference:

- CAGD is the older discipline. It was founded more than 10 years before Scientific Visualization. When Visualization started its fast-growing developments in the 80's, most of the relevant concepts and methods of CAGD were already present and could be included. Reversely, most of the CAGD algorithms had to be developed without the tools of Scientific Visualization in mind.

The reason for this difference might be the fact that Scientific Visualization in general makes higher demands on the resources of a computer. In fact, many of the classical CAGD algorithms work on rather slow computers while visualization problems deal with large data sets and thus need high-end machines. Hence the development of Visualization had to wait until computers reached a suitable level of power.

From the historical difference between CAGD and Visualization we can obtain the following

Expectation 3 *Since CAGD was already well-researched when Scientific Visualization came up, we can expect a broad application of CAGD methods in Visualization. Also new applications of CAGD in Visualization can be expected.*

Considering the application of Visualization in CAGD, we get the following

Expectation 4 *Since most of the relevant CAGD algorithms were developed in a time when Scientific Visualization was not present yet, we expect only a small number of applications of Scientific Visualization in CAGD. On the other hand, this gives the chance that a redevelopment of classical CAGD problems with Visualization in mind may give improvements of the classical CAGD algorithms.*

2.3 Pipelines in Visualization and CAGD

In Computer Graphics it is a common approach to describe the processes in terms of pipelines. This way a systematic treatment of the data and processes

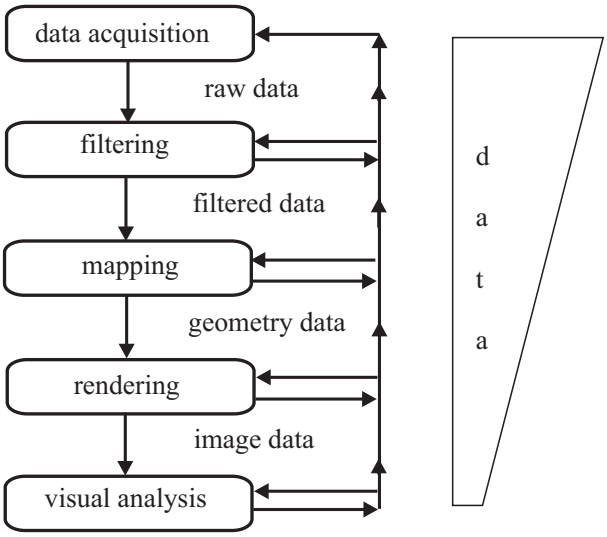


Figure 2.1: The visualization pipeline.

is possible (despite the fact that there are some applications which only run outside the pipelining concepts). Hence pipelines can be considered as a tool which gives a unified approach to most of the data and processes. Examples of pipelines in classical Computer Graphics are the rendering pipeline and the viewing pipeline ([59], [50], [51]).

Since both CAGD and Scientific Visualization have strong correlations to Computer Graphics, a logical question is to ask for their pipelines. We do so in the following sections 2.3.1 and 2.3.2. Section 2.3.3 summarizes which conclusions for the task of this work can be taken from the treatment of the pipelines.

2.3.1 The visualization pipeline

The visualization pipeline is a concept for a unified approach to Scientific Visualization which is widely accepted in the visualization community. Although there are argues about details of it, there is an agreement about the general structure of the visualization pipeline.

Figure 2.1 shows the visualization pipeline we want to use here. This pipeline is similar to the commonly accepted pipelines in [78] and [167]. The first step of the visualization pipeline, *data acquisition*, can also be considered as a pre-step of the pipeline. There is a variety of possibilities where the data to be visualized comes from. The data may be obtained by measurement or observation. Examples of this is data which was sent from satellites, or medical data from CT (Computer Tomography) devices. On the other hand, the data can be obtained by modeling and simulating certain processes. Examples of this are climate simulation models. However, for the following steps in the visualization pipeline it does not matter where the data to be visualized comes from. What matters is the fact that the data sets are usually rather large. For smaller data sets a visual analysis might not be necessary at all.

The result of the data acquisition step is *raw data* which goes into the next step of the visualization pipeline: the *filtering step*. It is the purpose of this step to modify the data to prepare it for the next steps. The filtering step might contain operations for data completion, data reduction, filtering, smoothing or extracting metadata out of the raw data. Data completion may be applied for incomplete data (i.e. if the data values are not available for all grid points). To do so, simple interpolations (nearest neighbor, (bi/tri)linear) may be applied. Since in this step only values for particular grid points have to be found, more involved interpolation schemes which consider a certain smoothness of the data are usually not necessary here.

Data reduction is necessary if the data set is either too large to be processed by the visualization system, or if the data set contains strong redundancies. This reduction may be done interactively by applying selection or projection algorithms, or it may be done (semi)automatically by applying statistical approaches to detect redundancies and areas of high information.

Filtering and smoothing operations may be applied to the raw data in order to remove the noise which comes from measuring the data. To do so, operations such as Gaussian filters or Laplacian smoothing may be applied.

Metadata collects information about the current data set. It may contain quantitative and qualitative statements about the data. This data may be used to steer and control the following steps of the visualization pipeline.

After applying the filtering step to the data, the filtered data goes to the *mapping step* of the visualization pipeline. This step is actually the core of the visualization pipeline. The filtered data has to be mapped to geometric primitives and their drawing attributes. Since a variety of primitives is possible, care has to be taken to choose them in such a way that the information in the filtered data is represented in an appropriate way.

In the *rendering step* of the visualization pipeline, the geometric primitives have to be mapped onto the 2D screen. This issue is not a specific problem in visualization. Instead, standard approaches of Computer Graphics ([59]) can be applied here. The resulting image / sequence of images can now be visually analyzed by the scientist.

As we can see in figure 2.1, the visualization pipeline is an iterative process. Analyzing the resulting images, the scientist may decide to go back in the visualization pipeline and change parameters in one of the upper steps. This way the new visualization may give better results to the scientist who can repeat these iterative steps as often as necessary. Of course, iterations to higher levels are possible at virtually every step of the visualization pipeline.

The visualization pipeline in figure 2.1 serves as a model for general scientific data. For particular data classes, special pipelines have been introduced (see chapters 3–5). These pipelines differ in certain details but have a globally similar structure to the general pipeline of figure 2.1.

Considering the amount of data which is present in the visualization pipeline, we can make the following statement: the further the visualization pipeline is processed, the less data is present. This is justified by the fact that the original

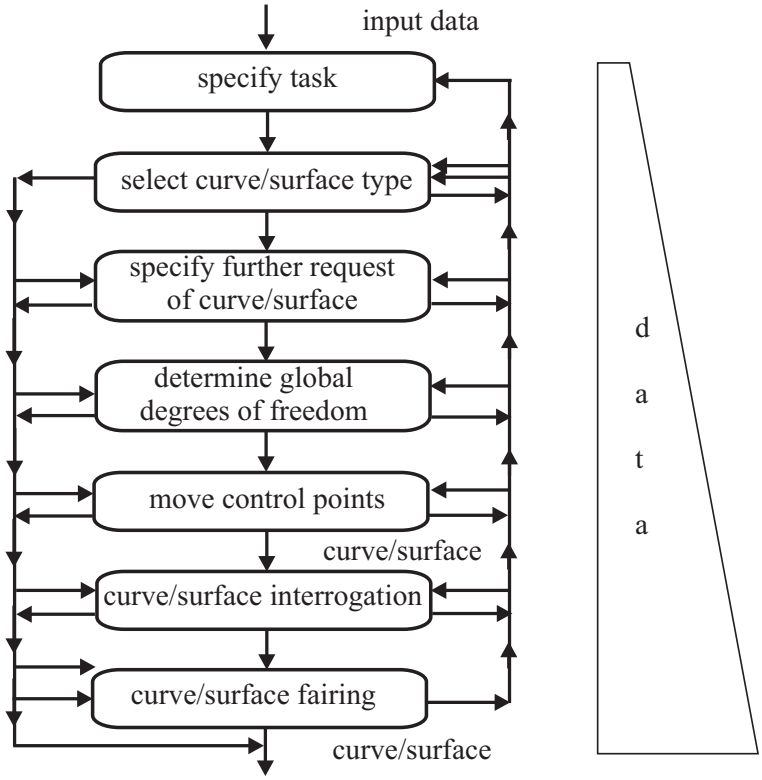


Figure 2.2: The CAGD pipeline.

large amount of raw data has to be decreased as much as necessary to place it onto the screen. Of course, this global statement is only a trend and has exceptions. For instance a data completion in the filtering step may increase the amount of data temporarily.

2.3.2 The CAGD pipeline

Since CAGD has some of its foundations in Computer Graphics, it is a rather natural step to bring the processes which belong to CAGD into a pipeline. Modern CAGD consists of a variety of problems, approaches, processes, and open questions; the ordering of them in a pipeline seems to be useful but not done yet. It is the purpose of this section to introduce a pipeline where we insert the most relevant processes of CAGD in a systematic way. We are aware that not all CAGD processes can be included in such a pipeline, but this is a problem of other pipelines as well.

There is another reason to try to put the CAGD process into a pipeline: it makes CAGD comparable to the visualization pipeline. Hence, from comparing the CAGD pipeline and the visualization pipeline we might derive more expectations on how to apply visualization to CAGD, and vice versa.

Figure 2.2 shows the CAGD pipeline we want to consider here. The pipeline starts with some *input data*. This input data may be rather vague and not

formalized. The designer may have an idea of the curve/surface to be designed in mind, or he/she may have a rough hand drawing of the desired shape. The input data may also be a set of measured points to be interpolated or approximated later.

In the next step of the CAGD pipeline, the designer has to *specify the task* of the curve/surface to be designed. So he/she has to determine if the curve/surface should interpolate or approximate the input data, or to fit certain points/curves of the input data. It might also happen that – if no data points came as input data – no particular task of the resulting curve/surface has to be specified.

In the next step the designer has to select the *curve/surface type* to be used. He/she can choose between Bézier curves/surfaces, Bézier-spline curves/surfaces, B-spline curves/surfaces, Coons patches, or other curve and surface schemes. It has also to be decided if, for a polynomial curve/surface, a rational or non-rational version is used. In the surface case, it has also to be decided if triangular or tensorproduct surfaces are used. For special input data and special tasks, surfaces of general topology (n-sided patches) or other surface concepts like Gregory patches ([71]) may also be considered.

Obviously, the decision for an appropriate curve/surface type is crucial for the success of the design process, but it is not trivial at all. A number of aspects influence this decision, like the input data, the specified task, and also the question which curves/surfaces are available for the design system to be used.

Once the type of the curves/surface to be used is specified, *further requests of the curves/surfaces* to be designed have to specified in the next step. For instance for piecewise polynomial curves/surfaces, a certain global continuity may be requested. Another request to the curve/surface may be the preservation of convexity properties, or the minimization of a certain energy functional.

In the next step of the CAGD pipeline, *global degrees of freedom* have to be determined. The kind of these degrees of freedom depend on the decisions made above. For Bézier- and B-spline curves, the degree of freedom to be determined are parameterization and the end conditions of the curve. For tensor product Bézier- and B-spline surfaces, the twist vectors additionally have to be fixed.

The next step of the CAGD pipeline is the core of the design process. After all the decisions above have been made, the designer can interactively *move the control points* until an appropriate curve/surface appears on screen. Depending on the chosen curve/surface scheme, the control points to be moved might be Bézier points (for Bézier- or Bézier-spline curves/surfaces), de Boor points (for B-spline curves/surfaces), or Farin points (for rational curves/surfaces). The result of this step is a first version of the curve/surface which has to be evaluated in the next step of the visualization pipeline.

In the *curve/surface interrogation* part of the CAGD pipeline the curve/surface is checked for any undesired behavior which is not visible at first glance. Typical methods for curve/surface interrogation are curvature plots, isophotes, or reflection lines. The results of curve/surface interrogation methods are the

parts of the curve/surface where a redesign is necessary.

In the last step of the CAGD pipeline, a *curve/surface fairing* is done. Automatic methods are applied to move the control points by minimal amounts to increase the fairness of the surface while keeping the general shape. Curve/ surface fairing methods may be based on curve/surface interrogation algorithms, or they may focus on obtaining certain higher continuities of the curve/surface.

As we can see in figure 2.2, the CAGD pipeline, too, describes an iterative process. After every step of the pipeline, the designer may go back to earlier steps of the pipeline to apply a redesign there. In figure 2.2, this is illustrated by the upward arrows on the right-hand side of the boxes.

In the CAGD pipeline not all parts have to be processed. For instance, it is possible to omit the part "select curve/surface type" and go directly to the part "specify further requests of curve/surface". If rather strong requests are formulated here (for instance to minimize certain integrals), the type of the curve/surface can be derived directly from this. For example, many spline functions have been developed to minimize certain functionals.

Another example of omitting parts of the CAGD pipeline are variational design approaches in curve and surface modeling ([80], [20], [72], [24]). There the curve/surface type and rather strong further requests (minimizing functionals) are defined in such a way that the curve/surface is already uniquely defined. In this case, no further degrees of freedom have to be determined and no control points have to be moved.

The fact that parts of the CAGD pipeline can be left out is illustrated by the downward arrows on the left-hand side of the boxes in figure 2.2.

Considering the amount of data which has to be dealt with in the CAGD pipeline, we can make the following statement: the amount of data increases from the top to the bottom of the pipeline. This is not surprising because the design process starts with almost nothing and ends in curves/surfaces which are rather complex geometric objects. As in the case of the visualization pipeline, this statement is a global one. Local variation are possible.

2.3.3 Conclusions from the pipelines

In this section we evaluate the pipelines for visualization and CAGD in such a way that we get more detailed information about the question where to apply CAGD methods for visualization, and vice versa. Before doing so, we have to mention that the two pipelines were introduced at different levels of abstraction. The visualization pipeline was introduced in a quite rough level of detail. In fact, more detailed visualization pipelines have been introduced especially for particular data classes. The CAGD pipeline here was newly suggested in this work and thus needed a more detailed description. Despite of the different levels of abstraction of the two pipelines, we do a comparison of them in the following.

As shown in figure 2.1, visualization is especially useful for rather large input data sets. Since in the CAGD pipeline the amount of data is growing when approaching the lower parts of the pipeline, we can formulate the following

Expectation 5 *Most applications of Scientific Visualization in CAGD can be expected in the lower parts of the CAGD pipeline.*

Indeed, as we will see later in chapter 7, the main applications of Scientific Visualization are in the part "move control points" and "curve/surface interrogation" of the CAGD pipeline.

Asking the reverse question, where CAGD methods can be applied in the visualization pipeline, a similar statement to expectation 5 is not possible. This is due to the fact that we defined the application of CAGD in a rather general way: we considered both a simple usage of curves/surfaces and for instance the application of interpolation methods as CAGD applications. Hence we can formulate the following

Expectation 6 *Applications of CAGD methods can be found in almost all parts of the visualization pipeline.*

Later we will apply CAGD methods for data acquisition, for filtering, and for the mapping step. Only the rendering step was left untouched because there standard methods of Computer Graphics apply, and no specific visualization background is necessary.

2.4 Strategies for Proceeding Further

After analyzing CAGD and Scientific Visualization concerning data, history and pipelines, we obtain the following strategies for proceeding further:

Concerning expectations 3 and 4, the treatment of applications of CAGD in Scientific Visualization is larger than the treatment of applications of Scientific Visualization Visualization in CAGD. Hence the application of CAGD in Scientific Visualization is treated in the four chapters 3–6 while for the treatment of the application of Scientific Visualization in CAGD only the one chapter 7 is reserved.

Following the trend in Scientific Visualization to focus on particular data classes, we split the treatment of applying CAGD to Scientific Visualization to the different data classes in Scientific Visualization. Hence the application of CAGD for volume data is treated in chapter 3, for flow data in chapter 4, for multiparameter data in chapter 5, and for the remaining data classes in chapter 6.

Concerning expectation 2, chapter 4 can be expected to be the largest.

Chapter 3

CAGD for Volume Visualization

Volume data is one of the most important data classes in scientific visualization. The main applications of volume visualization are in medicine, meteorology and other areas of natural science. Volume data may come from computer tomography (CT) devices, or from raster electron microscopes.

The volume data we consider here consists of single scalar values on a regular 3D grid. The grid is given by two points $\mathbf{x}_{min} = (x_{min},\ y_{min},\ z_{min})^T$, $\mathbf{x}_{max} = (x_{max},\ y_{max},\ z_{max})^T$ and the grid resolutions n_x, n_y, n_z in $x-, y-$ and $z-$ direction. Then the grid points can be computed as

$$\mathbf{x}_{i,j,k} = \begin{pmatrix} \frac{n_x - i}{n_x}\ x_{min} + \frac{i}{n_x}\ x_{max} \\ \\ \frac{n_y - j}{n_y}\ y_{min} + \frac{j}{n_y}\ y_{max} \\ \\ \frac{n_z - k}{n_z}\ z_{min} + \frac{k}{n_z}\ z_{max} \end{pmatrix} \quad \text{for} \quad \begin{array}{l} i = 0, ..., n_x \\ j = 0, ..., n_y \\ k = 0, ..., n_z \end{array} . \tag{3.1}$$

Research has also been done on volume data on curvilinear or irregular grids [176]. Since most of volume data comes on regular grids we restrict ourselves to the treatment of this type of grid. Here we also assume that the grid is normalized, i.e. $\mathbf{x}_{min} = (0,0,0)^T$, $\mathbf{x}_{max} = (n_x, n_y, n_z)^T$. This can always be achieved by applying an appropriate translation and scaling to the original grid. For the normalized grid we obtain

$$\mathbf{x}_{i,j,k} = \begin{pmatrix} i \\ j \\ k \end{pmatrix} \quad \text{for} \quad \begin{array}{l} i = 0, ..., n_x \\ j = 0, ..., n_y \\ k = 0, ..., n_z \end{array} . \tag{3.2}$$

Given a regular normalized grid $\mathbf{x}_{i,j,k}$, the volume data is simply described by a 3D array $c_{i,j,k}$ of scalar values. It means that at the grid point $\mathbf{x}_{i,j,k}$ the scalar value $c_{i,j,k}$ was measured or computed. Concerning the data classification of [23], volume data can be described as $E^S_{[3]}$.

Before visualizing a volume data set given by $\mathbf{x}_{i,j,k}$ and $c_{i,j,k}$, it has to be converted into a *scalar field*

$$s : [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}] \rightarrow \mathbb{R}$$

by applying an interpolation between the grid points. This means that s has to be chosen in such a way that

$$s(\mathbf{x}_{i,j,k}) = c_{i,j,k} \quad \text{for} \quad \begin{array}{l} i = 0, ..., n_x \\ j = 0, ..., n_y \\ k = 0, ..., n_z \end{array} .$$

The choice of an appropriate interpolation influences the behavior of the scalar field and thus the whole visualization process. Various ways of interpolating volume data are discussed in sections 3.3 – 3.8.

A variety of techniques have been developed to visualize volume data or the scalar fields derived from them. It is not the purpose of this work to survey existing techniques of volume visualization. Instead, here we try to find out for which classes of techniques in volume visualization approaches of CAGD may be applied. Surveys on volume visualization can be found in [107] and [167].

3.1 Techniques for Volume Visualization

The general approaches to visualize volume data can be classified into three groups:

- decomposition methods
- direct volume rendering
- isosurface extraction.

Decomposition methods visualize certain subsets of the scalar field. These subsets may be slices, particular points of the scalar field, or small geometric objects. A representative of the last-named subset is the vanishing cubes method ([143], [83]). Also in [143] a slicing approach is introduced which represents the data of certain slices as height surfaces. Given a point $\mathbf{x}_p = (x_p, y_p, z_p)^T$ in the volume, three bivariate scalar fields s^x, s^y, s^z can be defined out of s by

$$\begin{aligned} s^x(y,z) &= s(x_p, y, z) \\ s^y(x,z) &= s(x, y_p, z) \\ s^z(x,y) &= s(x, y, z_p). \end{aligned}$$

These scalar fields can be visualized using color coding or as height fields over their domain planes. Figure 3.1 gives an illustration of this.

Decomposition methods focus on simple geometric shapes such as planes or tiny cubes. More complicated shapes such as curves or surfaces are not considered in this group of visualization techniques. Also the choice of a particular interpolation between the grid points is of less importance here because decomposition methods usually use discrete shapes. This is the reason that further CAGD applications of decomposition methods are not known (except for the example in figure 3.1).

To visualize the volume data set using direct volume rendering, either ray casting or cell projection methods are used. In the ray casting approach, for

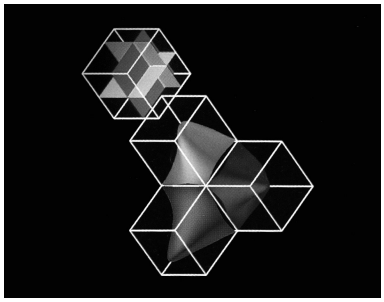


Figure 3.1: 2D slices for a point of interest in the volume; its three 2D scalar fields are visualized as height fields; the point of interest can be moved interactively (from [143]).

each pixel a ray is sent through the volume. Along the ray the contributions of the points in the scalar field are sampled. This means that the value for a pixel is a certain average of the scalar values along the ray. The ray casting approach can also be used to visualize isosurfaces. In this case an intersection of the ray with a particular isosurface has to be computed. See [147] for a discussion of this. This pixelwise representation of isosurfaces may be handled in real time on high-end graphics workstations. However, this representation is view dependent; the change of the view point requires a complete recomputation of the representation. An alternative is a surface oriented representation of the isosurface either as a triangular mesh or as parametric surfaces. These approaches, which are candidates for applying CAGD methods, are discussed in the sections 3.2 – 3.8.

Using cell projection for direct volume rendering, each cell is semitransparently projected to the screen. This way the final color of a pixel consists of an average of a number of projected cells. See [209] for a survey of direct volume rendering methods. Direct volume rendering is of particular usefulness when analyzing the whole scalar field. Particular surfaces are less emphasized here. (Even in the case of ray casting an isosurface, the result is a pixelwise description.) Also the question of what kind of interpolation to use plays a less important role because the scalar values are sampled at discrete points. Thus applications of CAGD methods to improve direct volume visualization are not known.

The majority of applications of CAGD methods can be expected for the extraction of isosurfaces for volume data. Here the graphical objects to be analyzed are representations of isosurfaces for a given threshold. Thus the remaining part of this chapter focuses on the question of how to apply CAGD methods to deal with isosurfaces of volume data.

3.2 Pipeline for Isosurface Extraction of Volume Data

Figure 3.2 shows the pipeline for the process of isosurface extraction for volume data. We recognize the three steps (filtering, mapping, rendering) of the usual visualization pipeline (see section 2.3.1).

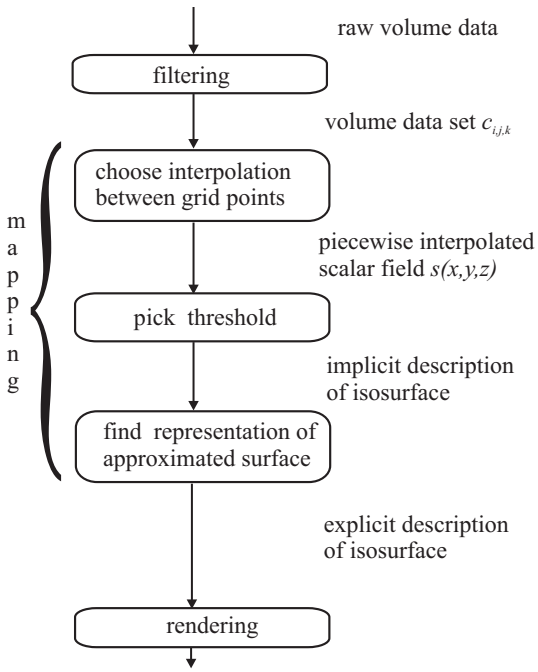


Figure 3.2: Pipeline for isosurface extraction of volume data.

As the first step of the mapping process, an appropriate interpolation between the grid points has to be chosen. This process transforms the volume data into a piecewise interpolated scalar field. Note that this step is done before a particular threshold is chosen. [1]

The scalar field and a picked threshold give an implicit description of the isosurface (contour). It consists of all points in the scalar field which attain the chosen scalar value. It is the task of the next step of the pipeline to convert this implicit description of the surface into an explicit representation. This explicit representation is finally sent to the rendering process.

In [76] and [65] extensions of isosurface extractions algorithms are treated by considering interval sets. Interval sets are given by the definition of two thresholds and consist of all points in the scalar field which have a scalar value between the two thresholds. In this work we restrict ourselves to isosurfaces because surfaces are classical objects treated in CAGD.

The choice of a particular interpolation of the volume data has a great influence on the rest of the mapping process for isosurface extraction. The following kinds of interpolation between the grid points are possible:

[1] If the volume data set is incomplete (i.e. the scalar values at certain grid points are unknown) a data completion has to be performed as part of the filtering step (see [167]). In this case the unknown values are obtained by interpolation of the values at adjacent grid points. We want to distinguish between this kind of interpolation in the filtering process and the interpolation in the mapping process. The interpolation in the filtering process computes additional scalar values at a finite number of grid points while the interpolation in the mapping step applies to the whole volume.

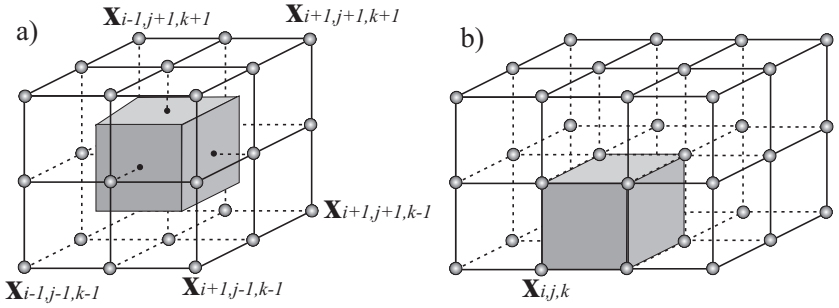


Figure 3.3: a) voxel $V_{i,j,k}$; b) cell $C_{i,j,k}$.

- piecewise constant interpolation
- piecewise linear interpolation
- piecewise trilinear interpolation
- piecewise higher order polynomial interpolation
- piecewise trilinear interpolation with local reparametrization
- piecewise trilinear interpolation over larger areas

We want to study each of these kinds of interpolation to explore where CAGD methods can be applied. To do this we reserve one of the sections 3.3 – 3.8 for each kind of interpolation. Sections 3.3 – 3.8 are organized in the following way: first we study the nature of the isosurface for the particular interpolation, then ways of their explicit graphical representation are discussed. Here it is our particular interest to study if an effective description as a parametric surface is possible.

3.3 Piecewise Constant Interpolation

For doing a piecewise constant interpolation of the volume data, a Dirichlet tessellation of the grid points (see [55]) is applied. All points inside the Dirichlet cell of a certain grid point $\mathbf{x}_{i,j,k}$ get the scalar value $c_{i,j,k}$. For a regular grid, the Dirichlet cells around the grid point $\mathbf{x}_{i,j,k}$ form a box with its center in $\mathbf{x}_{i,j,k}$. This box is called a *voxel* $V_{i,j,k}$. Figure 3.3a gives an illustration. [2]

An "isosurface" of a piecewise constant scalar field consists of a number of voxels - all voxels with its scalar value within a certain tolerance to the picked threshold. To find a graphical representation of this set of voxels, they may be sent directly to the renderer, or they may be transferred to a triangular representation. Representatives of the last-named strategy are the cuberilles introduced in [89]. Figure 3.4 shows an example of a conversion of voxel data into a triangular mesh.

[2]Sometimes a voxel is not defined as shown in figure 3.3a but as shown in figure 3.3b. In this case all points inside the box have the constant scalar value $c_{i,j,k}$ of one particular vertex $\mathbf{x}_{i,j,k}$. The properties and results of both voxel definitions coincide.

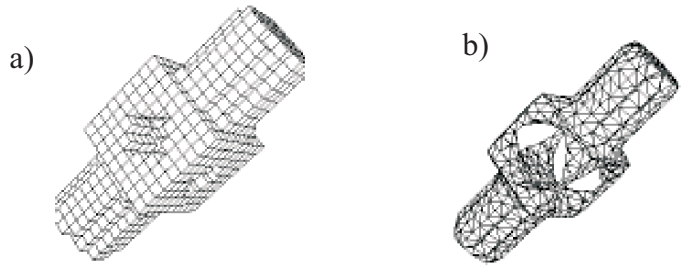


Figure 3.4: Transforming voxel data (a) to a triangular mesh (b) (from [132]).

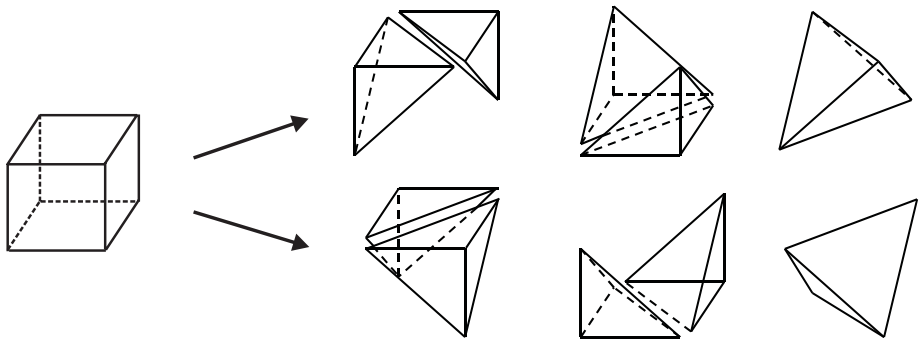


Figure 3.5: Two ways of subdividing a cell into 5 tetrahedra.

If the voxels are large, both representations (voxel and triangular) look rather rough. In this case better interpolations of the volume data should be applied (see sections 3.4 – 3.8). For smaller voxels both representations may give optically good results. In this case it depends on the specialization of the graphical workstation if voxel data or triangular data should be chosen for representation.

Due to the discrete nature of a piecewise constant interpolation, approaches to apply parametric surfaces for piecewise constant volume data are not known.

3.4 Piecewise Linear Interpolation

To obtain a piecewise linear interpolation of the volume data we consider *cells* in the volume data. The cell $C_{i,j,k}$ is a box defined by the eight grid points $\mathbf{x}_{i,j,k}$, $\mathbf{x}_{i+1,j,k}$, $\mathbf{x}_{i,j+1,k}$, $\mathbf{x}_{i+1,j+1,k}$, $\mathbf{x}_{i,j,k+1}$, $\mathbf{x}_{i+1,j,k+1}$, $\mathbf{x}_{i,j+1,k+1}$, $\mathbf{x}_{i+1,j+1,k+1}$ obtained from (3.1):

$$C_{i,j,k} = [x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}]. \tag{3.3}$$

Figure 3.3b gives an illustration. To apply a piecewise linear interpolation, each cell has to be subdivided into a number of tetrahedra. Several approaches exist for doing this. Figure 3.5 shows two different ways to subdivide a cell into 5 tetrahedra. Subdivisions into 6 or 24 tetrahedra are applied as well (see [142]).

For computing the scalar value of a point $\mathbf{x}$ inside a certain tetrahedron we apply a linear interpolation of the scalar values in the vertices of the tetrahedron. This way the scalar value at $\mathbf{x}$ is a weighted sum of the scalar values

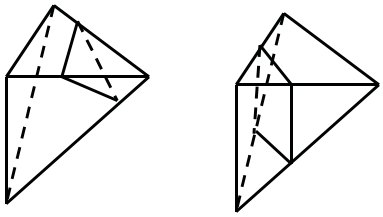


Figure 3.6: Two cases of the marching tetrahedra algorithm.

at the vertices where the weights are obtained by computing the barycentric coordinates of $\mathbf{x}$ relative to the tetrahedron vertices.

An isosurface of a piecewise linear interpolated scalar field inside a tetrahedron is always plane. Thus it is a natural approach to represent the contour of a piecewise linear scalar field as a triangular mesh. One algorithm which computes the isosurface for all tetrahedra of a scalar field is the marching tetrahedra algorithm ([37]). For each tetrahedron, the scalar values at the vertices are checked to see if they are larger or smaller than the picked threshold. If all four values are smaller (or larger) than the threshold, the isosurface does not pass the tetrahedron. In all other cases the exact isosurface can be computed by obtaining the intersections of the isosurface and the edges of the tetrahedron. To do this, linear interpolations of the vertices along the edges are applied. Here two cases are possible, which are illustrated in figure 3.6. An improvement of the marching tetrahedra algorithm is introduced in [174]. There, data structures are built to prevent tetrahedra where the isosurface does not pass through from being processed by the algorithm.

The marching tetrahedra algorithm computes the isosurface of a piecewise linear scalar field exactly. It was originally developed to overcome the ambiguities of the Marching Cubes algorithm (described in section 3.5.2.2). In fact, once the grid is tetrahedrized, only the simple cases shown in figure 3.6 appear. The problem is the choice of a particular tetrahedrization of the cells. This decision has to be done independently of the particular data but has an influence on the shape of the isosurface. This means that the ambiguities from which the Marching Cubes algorithm suffers exist for the marching tetrahedra algorithm as well. They are only shifted into the selection of an appropriate tetrahedrization. This is one reason why in most cases a trilinear interpolation is preferred to a piecewise linear interpolation. Another reason is that a tetrahedrization of the cells usually produces a higher number of triangles.

Since an isosurface of a piecewise linear scalar field is piecewise plane, the piecewise triangular representation is optimal; no further CAGD applications for this kind of interpolation are known.

3.5 Piecewise Trilinear Interpolation

As in the case of piecewise linear interpolation, the volume is subdivided into cells by applying (3.3). In each cell a local trilinear interpolation of the scalar values in the vertices is carried out to get the scalar values inside the cell.

To study properties of the piecewise trilinear interpolation, we consider the cell $C_{000} = [0,1]^3$ of a regular normalized grid given by (3.2). Then the scalar

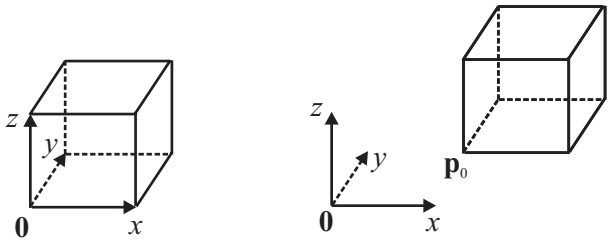


Figure 3.7: Translating the coordinate system of the cell C_{000}.

field in C_{000} can be written as

$$\begin{aligned} s(x,y,z) = \quad & (1-x)(1-y)(1-z)\,c_{000} \;+\; (1-x)(1-y)\,z\,c_{001} \\ + \quad & (1-x)\,y\,(1-z)\,c_{010} \;+\; (1-x)\,y\,z\,c_{011} \\ + \quad & x\,(1-y)(1-z)\,c_{100} \;+\; x\,(1-y)\,z\,c_{101} \\ + \quad & x\,y\,(1-z)\,c_{110} \;+\; x\,y\,z\,c_{111} \end{aligned} \tag{3.4}$$

where c_{ijk} $(i,j,k \in \{0,1\})$ are the scalar values in the vertices $(i,j,k)^T$ of C_{000}. A contour (isosurface) is given by specifying a threshold r; it consists of all points $(x,y,z)^T$ with

$$s(x,y,z) \;=\; r. \tag{3.5}$$

The trilinear interpolation is widely used and has a variety of applications. Section 3.5.1 gives a collection of properties of trilinear contours. These properties are used to survey and improve algorithms for extracting trilinear contours in section 3.5.2.

3.5.1 Properties of trilinear contours

The contours of a trilinear scalar field can be studied in two general ways.

1. We analyze the contour of a particular threshold defined by (3.4) and (3.5). This analysis can be done for a contour either in a cell or in the domain $\mathbb{R}^3$.

2. We analyze all contours passing through the cell $C_{i,j,k}$ to get global statements about the cell itself.

The following sections give characteristics of both methods of analysis.

3.5.1.1 Connectivity of a contour

Here we consider the contour of (3.4) and (3.5) not only in the cell $C_{i,j,k}$ but in the domain $\mathbb{R}^3$. In general, the contour consists of a number of surface parts which are not necessarily connected to each other. It is the purpose of this section to study how many unconnected surface parts the contour of (3.4) and (3.5) consists of. This gives a classification of all contours of (3.4) and (3.5). To do this, we apply a translation of the coordinate system as shown in Figure 3.7. This translation is defined by the scalar values $c_{i,j,k}$ $(i,j,k \in \{0,1\})$ at the grid

points $(i,j,k)^T$. Choosing

$$\begin{aligned} p &= c_{001}+c_{010}+c_{100}+c_{111}-c_{000}-c_{011}-c_{101}-c_{110} \\ \mathbf{p}_0 &= \begin{pmatrix} x_{\mathbf{p}_0} \\ y_{\mathbf{p}_0} \\ z_{\mathbf{p}_0} \end{pmatrix} = \frac{1}{p}\cdot \begin{pmatrix} c_{000}+c_{011}-c_{001}-c_{010} \\ c_{000}+c_{101}-c_{001}-c_{100} \\ c_{000}+c_{110}-c_{010}-c_{100} \end{pmatrix}, \end{aligned} \tag{3.6}$$

equation (3.4) can be written in the form

$$s(x,y,z) = a\,x + b\,y + c\,z + d\,x\,y\,z + e \tag{3.7}$$

with

$$\begin{aligned} a &= \frac{(c_{111}-c_{011})\cdot(c_{100}-c_{000})-(c_{110}-c_{010})\cdot(c_{101}-c_{001})}{p} \\ b &= \frac{(c_{111}-c_{101})\cdot(c_{010}-c_{000})-(c_{110}-c_{100})\cdot(c_{011}-c_{001})}{p} \\ c &= \frac{(c_{111}-c_{110})\cdot(c_{001}-c_{000})-(c_{101}-c_{100})\cdot(c_{011}-c_{010})}{p} \\ d &= p, \end{aligned}$$

and e is a certain constant. Thus, we only have to analyze the contours of

$$s(x,y,z) = a\,x + b\,y + c\,z + d\,x\,y\,z = r = \text{const} \tag{3.8}$$

in $\mathbb{R}^3$. A classification of (3.8) can be achieved by rewriting it as a height field

$$z(x,y) = \frac{r - a\,x - b\,y}{c + d\,x\,y} \tag{3.9}$$

and comparing the zeros of the numerator and denominator function. The zeros of the numerator function form a line in the $x-y-$plane, whereas the zeros of the denominator function give a hyperbola. Studying their interplay gives the following classification:

- case 1: $a\,b\,c\,d < 0$:
  - case 1.1: $r^2 > -\frac{4\,a\,b\,c}{d}$: (3.9) gives 3 unconnected surface parts [3]
  - case 1.2: $r^2 \le -\frac{4\,a\,b\,c}{d}$: (3.9) gives 2 unconnected surface parts [4]
- case 2: $a\,b\,c\,d > 0$: (3.9) consists of 1 connected part
- case 3: $a\,b\,c\,d = 0,\ d \neq 0$:
  - case 3.1: $r \neq 0$:

[3] In this case the line $r - a\,x - b\,y = 0$ does not intersect the hyperbola $c + d\,x\,y = 0$. This denominator hyperbola of (3.9) divides the $x-y-$ plane into three parts which correspond to the three unconnected surface parts of (3.9).

[4] In this case the line $r - a\,x - b\,y = 0$ intersects one branch of the hyperbola $c + d\,x\,y = 0$ twice. For these intersection points (x_1, y_1) and (x_2, y_2) we obtain $a\,x_1 + b\,y_1 + c\,z + d\,x_1\,y_1\,z = r$ and $a\,x_2 + b\,y_2 + c\,z + d\,x_2\,y_2\,z = r$ for any z. This means that the lines (x_1, y_1, z) and (x_2, y_2, z) for $z \in \mathbb{R}$ are on the contour defined by (3.8). Thus the height surface of (3.9) consists of two parts.

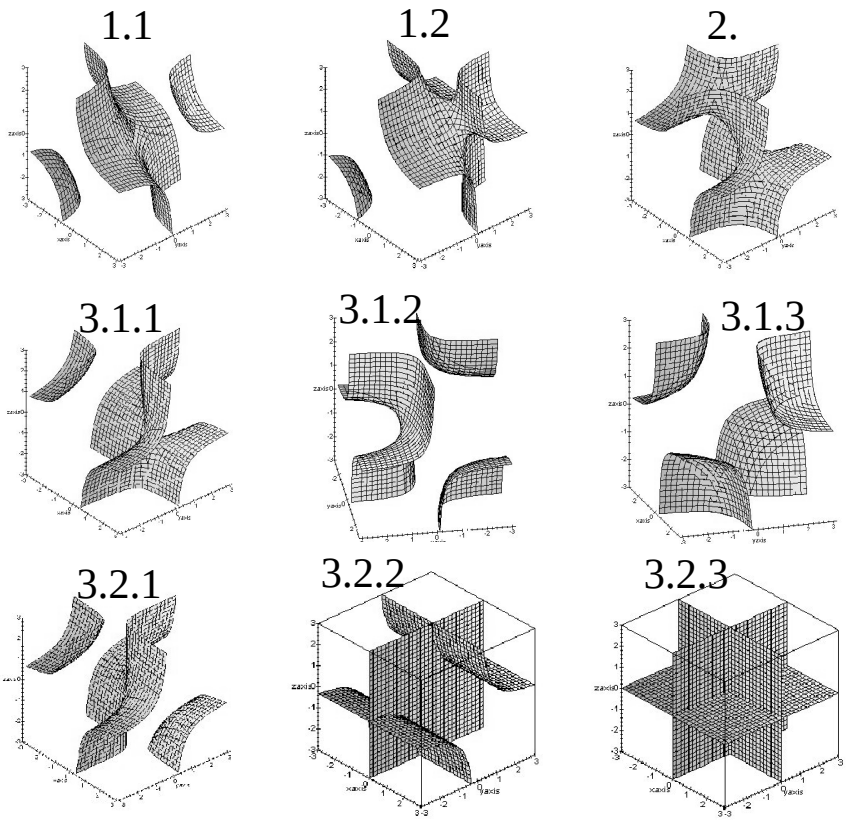


Figure 3.8: Classification of the contours of (3.8) in $\mathbb{R}^3$.

  - case 3.1.1: $a\,b \neq 0,\ c = 0$: (3.9) gives 2 unconnected surface parts
  - case 3.1.2: $a \neq 0, b = c = 0$: (3.9) gives 3 unconnected surface parts
  - case 3.1.3: $a = b = c = 0$: (3.9) gives 4 unconnected surface parts

- case 3.2: $r = 0$:
  - case 3.2.1: $a\,b \neq 0,\ c = 0$: (3.9) gives 3 unconnected surface parts
  - case 3.2.2: $a \neq 0, b = c = 0$: (3.9) gives 3 parts intersecting each other
  - case 3.2.3: $a = b = c = 0$: (3.9) gives 3 perpendicular planes.

Figure 3.8 gives illustrations of these cases.

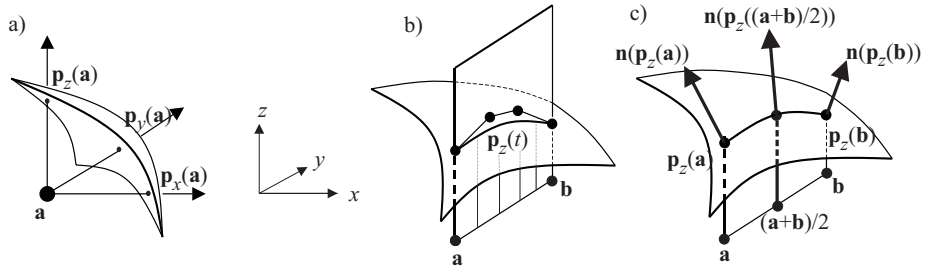


Figure 3.9: a) projections $\mathbf{p}_x(\mathbf{a})$, $\mathbf{p}_y(\mathbf{a})$, $\mathbf{p}_z(\mathbf{a})$ of a point $\mathbf{a}$ onto the contour in $x-$, $y-$ and $z-$direction; b) projection $\mathbf{p}_z(t)$ of the line segment $(1-t)\,\mathbf{a}+t\,\mathbf{b}$ onto the contour is a rational cubic curve; c) configuration for computing the control points of $\mathbf{p}_z(t)$.

3.5.1.2 Bézier representation of the contour

As already mentioned in [84], the contour defined by (3.4) and (3.5) is a rational cubic surface [5]. In this section we want to find a representation of it as a triangular rational cubic Bézier surface.

Given a point $\mathbf{a} = (x_\mathbf{a}, y_\mathbf{a}, z_\mathbf{a})^T$, we compute the intersection of a ray starting from $\mathbf{a}$ and the contour defined by (3.4) and (3.5). In general this computation ends in the solution of a cubic equation. For the special case that the ray is parallel to one of the coordinate axes, the problem simplifies to the solution of a linear equation: let $\mathbf{p}_x(\mathbf{a})$ be the intersection of the ray $\mathbf{a}+\lambda\cdot(1,0,0)^T$ with the contour defined by (3.4) and (3.5). Furthermore, let $\mathbf{p}_y(\mathbf{a})$ be the intersection of the ray $\mathbf{a}+\lambda\cdot(0,1,0)^T$ with the contour, and let $\mathbf{p}_z(\mathbf{a})$ be the intersection of the ray $\mathbf{a}+\lambda\cdot(0,0,1)^T$ with the contour. Then we obtain from (3.4) and (3.5):

$$\mathbf{p}_x(\mathbf{a}) = \left(\frac{\begin{array}{c} r - c_{000}\,(1-y_\mathbf{a})(1-z_\mathbf{a}) - c_{001}\,(1-y_\mathbf{a})\,z_\mathbf{a} \\ - c_{010}\,y_\mathbf{a}\,(1-z_\mathbf{a}) - c_{011}\,y_\mathbf{a}\,z_\mathbf{a} \end{array}}{\begin{array}{c} (c_{100}-c_{000})(1-y_\mathbf{a})(1-z_\mathbf{a}) + (c_{101}-c_{001})(1-y_\mathbf{a})\,z_\mathbf{a} \\ + (c_{110}-c_{010})\,y_\mathbf{a}\,(1-z_\mathbf{a}) + (c_{111}-c_{011})\,y_\mathbf{a}\,z_\mathbf{a} \end{array}},\ y_\mathbf{a}\ ,\ z_\mathbf{a} \right)^T$$

$$\mathbf{p}_y(\mathbf{a}) = \left(x_\mathbf{a}\ ,\ \frac{\begin{array}{c} r - c_{000}\,(1-x_\mathbf{a})(1-z_\mathbf{a}) - c_{100}\,x_\mathbf{a}\,(1-z_\mathbf{a}) \\ - c_{001}\,(1-x_\mathbf{a})\,z_\mathbf{a} - c_{101}\,x_\mathbf{a}\,z_\mathbf{a} \end{array}}{\begin{array}{c} (c_{010}-c_{000})(1-x_\mathbf{a})(1-z_\mathbf{a}) + (c_{110}-c_{100})\,x_\mathbf{a}\,(1-z_\mathbf{a}) \\ + (c_{011}-c_{001})(1-x_\mathbf{a})\,z_\mathbf{a} + (c_{111}-c_{101})\,x_\mathbf{a}\,z_\mathbf{a} \end{array}}\ ,\ z_\mathbf{a} \right)^T$$

$$\mathbf{p}_z(\mathbf{a}) = \left(x_\mathbf{a}\ ,\ y_\mathbf{a}\ ,\ \frac{\begin{array}{c} r - c_{000}\,(1-x_\mathbf{a})(1-y_\mathbf{a}) - c_{010}\,(1-x_\mathbf{a})\,y_\mathbf{a} \\ - c_{100}\,x_\mathbf{a}\,(1-y_\mathbf{a}) - c_{110}\,x_\mathbf{a}\,y_\mathbf{a} \end{array}}{\begin{array}{c} (c_{001}-c_{000})(1-x_\mathbf{a})(1-y_\mathbf{a}) + (c_{011}-c_{010})(1-x_\mathbf{a})\,y_\mathbf{a} \\ + (c_{101}-c_{100})\,x_\mathbf{a}\,(1-y_\mathbf{a}) + (c_{111}-c_{110})\,x_\mathbf{a}\,y_\mathbf{a} \end{array}} \right)^T .$$

We call $\mathbf{p}_x(\mathbf{a})$, $\mathbf{p}_y(\mathbf{a})$, $\mathbf{p}_z(\mathbf{a})$ the *projections of* $\mathbf{a}$ *onto the contour* in $x-$, $y-$ and $z-$direction. This means that we can construct three points on the contour for a given point $\mathbf{a}$ in a simple way. Figure 3.9a gives an illustration.

[5] After mentioning that the contour of (3.4) and (3.5) is a rational cubic, [84] approximates it by a number of rational quadratic surfaces. This was motivated by the fact that the intersections of the contours with the faces of a cell are hyperbolas and therefore exactly describable by rational quadratic curves.

Given a point $\mathbf{a}$, there is one and only one contour defined by (3.4) through it. We can compute its (unnormalized) normal vector $\mathbf{n}(\mathbf{a})$ in $\mathbf{a}$ by

$$\mathbf{n}(\mathbf{a}) = \text{grad}(s(x_{\mathbf{a}}, y_{\mathbf{a}}, z_{\mathbf{a}})) = (s_x(x_{\mathbf{a}}, y_{\mathbf{a}}, z_{\mathbf{a}}), s_y(x_{\mathbf{a}}, y_{\mathbf{a}}, z_{\mathbf{a}}), s_z(x_{\mathbf{a}}, y_{\mathbf{a}}, z_{\mathbf{a}}))^T \tag{3.10}$$

where s_x, s_y, s_z are the partial derivatives of s defined in (3.4).

Now we construct curves on the contour by projecting line segments onto it. Given is the line segment $\mathbf{x}(t) = (1-t)\,\mathbf{a} + t\,\mathbf{b}$. Then the curves

$$\mathbf{p}_x(t) = \mathbf{p}_x(\mathbf{x}(t)) \;\;,\;\; \mathbf{p}_y(t) = \mathbf{p}_y(\mathbf{x}(t)) \;\;,\;\; \mathbf{p}_z(t) = \mathbf{p}_z(\mathbf{x}(t)) \tag{3.11}$$

are obtained by projecting each point of $\mathbf{x}(t)$ onto the contour in $x-, y-$, or $z-$direction . Figure 3.9b gives an illustration for $\mathbf{p}_z(t)$.

It is a straightforward exercise in algebra to show that the curves $\mathbf{p}_x(t)$, $\mathbf{p}_y(t)$, $\mathbf{p}_z(t)$ on the contour defined by (3.4) and (3.5) are rational cubics. The curve $\mathbf{p}_z(t)$ can be expressed as

$$\mathbf{p}_z(t) = \frac{\sum_{i=0}^{3} w_i\,\mathbf{b}_i\,B_i^3(t)}{\sum_{i=0}^{3} w_i\,B_i^3(t)} \tag{3.12}$$

where $B_i^3(t)$ are the Bernstein polynomials (see [55]) and

$$\begin{aligned}
w_0 = z_{\mathbf{n}(\mathbf{p}_z(\mathbf{a}))} \;\;,\;\; w_1 &= \frac{4}{3}\,z_{\mathbf{n}(\mathbf{p}_z(\frac{\mathbf{a}+\mathbf{b}}{2}))} - \frac{1}{3}\,z_{\mathbf{n}(\mathbf{p}_z(\mathbf{b}))} \\
w_3 = z_{\mathbf{n}(\mathbf{p}_z(\mathbf{b}))} \;\;,\;\; w_2 &= \frac{4}{3}\,z_{\mathbf{n}(\mathbf{p}_z(\frac{\mathbf{a}+\mathbf{b}}{2}))} - \frac{1}{3}\,z_{\mathbf{n}(\mathbf{p}_z(\mathbf{a}))} \\
\mathbf{b}_0 = \mathbf{p}_z(\mathbf{a}) \quad &, \quad \mathbf{b}_3 = \mathbf{p}_z(\mathbf{b})
\end{aligned} \tag{3.13}$$

$$\mathbf{b}_1 = \begin{pmatrix} (1-\frac{w_0}{3\,w_1})\,x_{\mathbf{b}_0} + \frac{w_0}{3\,w_1}\,x_{\mathbf{b}_3} \\ (1-\frac{w_0}{3\,w_1})\,y_{\mathbf{b}_0} + \frac{w_0}{3\,w_1}\,y_{\mathbf{b}_3} \\ (1+\frac{w_3}{3\,w_1})\,z_{\mathbf{p}_z(\frac{\mathbf{a}+\mathbf{b}}{2})} - \frac{w_3}{3\,w_1}\,z_{\mathbf{b}_3} \end{pmatrix}$$

$$\mathbf{b}_2 = \begin{pmatrix} \frac{w_3}{3\,w_2}\,x_{\mathbf{b}_0} + (1-\frac{w_3}{3\,w_2})\,x_{\mathbf{b}_3} \\ \frac{w_3}{3\,w_2}\,y_{\mathbf{b}_0} + (1-\frac{w_3}{3\,w_2})\,y_{\mathbf{b}_3} \\ (1+\frac{w_0}{3\,w_2})\,z_{\mathbf{p}_z(\frac{\mathbf{a}+\mathbf{b}}{2})} - \frac{w_0}{3\,w_2}\,z_{\mathbf{b}_0} \end{pmatrix}.$$

Figure 3.9c gives an illustration of the components used in (3.13). The curves $\mathbf{p}_x(t)$, $\mathbf{p}_y(t)$ can be computed as rational cubics in a similar way.

Now we extend the concept of curves on the contour to parametric surfaces on the contour defined by (3.4) and (3.5). Given is a triangle

$$\mathbf{x}(u,v,w) = u\,\mathbf{a} + v\,\mathbf{b} + w\,\mathbf{c} \tag{3.14}$$

in barycentric coordinates of the vertices $\mathbf{a}, \mathbf{b}, \mathbf{c}$, i.e. $u + v + w = 1$. Then $\mathbf{p}_x(\mathbf{x}(u,v,w))$, $\mathbf{p}_y(\mathbf{x}(u,v,w))$, $\mathbf{p}_z(\mathbf{x}(u,v,w))$ are the projections of $\mathbf{x}$ onto the contour defined by (3.4) and (3.5). Figure 3.10a gives an illustration for $\mathbf{p}_z(\mathbf{x}(u,v,w))$.

The surfaces $\mathbf{p}_x(\mathbf{x}(u,v,w))$, $\mathbf{p}_y(\mathbf{x}(u,v,w))$, $\mathbf{p}_z(\mathbf{x}(u,v,w))$ are rational cubics. For example, $\mathbf{p}_z(\mathbf{x}(u,v,w))$ can be described as a rational Bézier triangle

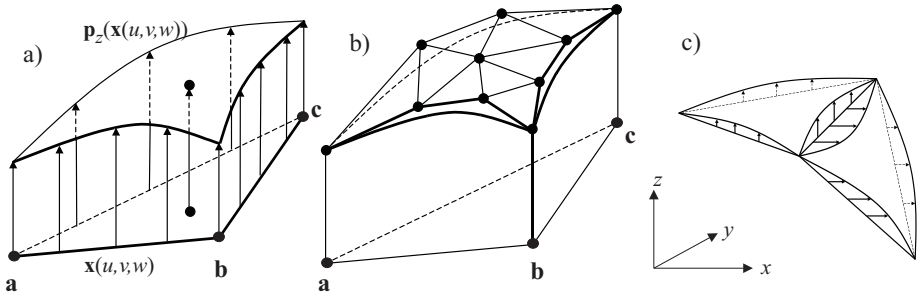


Figure 3.10: a) $\mathbf{p}_z(\mathbf{x}(u,v,w))$ is obtained by projecting every point of $\mathbf{x}(u,v,w) = u\,\mathbf{a} + v\,\mathbf{b} + w\,\mathbf{c}$ into $z-$direction onto the contour defined by (3.4) and (3.5); b) $\mathbf{p}_z(\mathbf{x}(u,v,w))$ is a rational cubic surface; c) two adjacent triangles projected in different directions: the resulting contour patches may have gaps.

(see [55])

$$\mathbf{p}_z(\mathbf{x}(u,v,w)) = \frac{\sum_{i+j+k=3} w_{ijk}\,\mathbf{b}_{ijk}\,B^3_{ijk}(u,v,w)}{\sum_{i+j+k=3} w_{ijk}\,B^3_{ijk}(u,v,w)} \tag{3.15}$$

where the Bézier points and their weights on the boundary curves can be computed as in (3.13) above, and

$$w_{111} = \frac{w_{201}+w_{102}+w_{021}+w_{012}+w_{120}+w_{210}}{4} - \frac{w_{300}+w_{030}+w_{003}}{6}$$

$$\begin{aligned} x_{\mathbf{b}_{111}} &= \left(\frac{w_{012}+w_{021}}{4\,w_{111}} - \frac{w_{030}+w_{003}}{12\,w_{111}}\right) x_{\mathbf{a}} \\ &+ \left(\frac{w_{102}+w_{201}}{4\,w_{111}} - \frac{w_{300}+w_{003}}{12\,w_{111}}\right) x_{\mathbf{b}} \\ &+ \left(\frac{w_{120}+w_{210}}{4\,w_{111}} - \frac{w_{300}+w_{030}}{12\,w_{111}}\right) x_{\mathbf{c}} \end{aligned}$$

$$\begin{aligned} y_{\mathbf{b}_{111}} &= \left(\frac{w_{012}+w_{021}}{4\,w_{111}} - \frac{w_{030}+w_{003}}{12\,w_{111}}\right) y_{\mathbf{a}} \\ &+ \left(\frac{w_{102}+w_{201}}{4\,w_{111}} - \frac{w_{300}+w_{003}}{12\,w_{111}}\right) y_{\mathbf{b}} \\ &+ \left(\frac{w_{120}+w_{210}}{4\,w_{111}} - \frac{w_{300}+w_{030}}{12\,w_{111}}\right) y_{\mathbf{c}} \end{aligned}$$

$$\begin{aligned} z_{\mathbf{b}_{111}} &= \frac{w_{210}\,z_{\mathbf{b}210} + w_{201}\,z_{\mathbf{b}201} + w_{120}\,z_{\mathbf{b}120}}{4\,w_{111}} \\ &+ \frac{w_{021}\,z_{\mathbf{b}021} + w_{012}\,z_{\mathbf{b}012} + w_{102}\,z_{\mathbf{b}102}}{4\,w_{111}} \\ &- \frac{w_{300}\,z_{\mathbf{p}_z(\mathbf{a})} + w_{030}\,z_{\mathbf{p}_z(\mathbf{b})} + w_{003}\,z_{\mathbf{p}_z(\mathbf{c})}}{12\,w_{111}}. \end{aligned}$$

Figure 3.10b gives an illustration. The surfaces $\mathbf{p}_x(\mathbf{x})$, $\mathbf{p}_y(\mathbf{x})$ can be described in a similar way.

In this section we have shown that a contour of (3.4) and (3.5) can be described as rational cubic Bézier triangles. To make this description usable to Computer Graphics, more problems have to be solved:

- we have to choose the triangles $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ to be projected
- we have to choose the projection direction for each triangle
- we have to eliminate gaps between the patches which happen by projecting adjacent triangles in different directions (see figure 3.10c for an example)
- we have to make sure that the weights of the rational cubic surfaces do not have alternating signs

These problems will be discussed in section 3.5.2.3 .

3.5.1.3 Special points on the contour - the inner ring

In this section we want to study special points on the contour defined by (3.4) and (3.5). These points will later be useful to find suitable triangulations of the contour. The points we consider are all points on the contour where the surface normal points either in $x-, y-$ or $z-$direction.

Given a contour defined by (3.4) and (3.5), there are at most two points $\mathbf{x}_0$, $\mathbf{x}_1$ on the contour with a normal in x-direction. Similarly, there are at most two points $\mathbf{y}_0$, $\mathbf{y}_1$ on the contour with a normal in y-direction, and there are at most two points $\mathbf{z}_0$, $\mathbf{z}_1$ on the contour with a normal in z-direction. These six points can be computed as

$$\begin{aligned}
\mathbf{x}_0 &= \begin{pmatrix} x_m - x_p\sqrt{\delta} \\ y_m + y_p\sqrt{\delta} \\ z_m + z_p\sqrt{\delta} \end{pmatrix}, \quad \mathbf{x}_1 = \begin{pmatrix} x_m + x_p\sqrt{\delta} \\ y_m - y_p\sqrt{\delta} \\ z_m - z_p\sqrt{\delta} \end{pmatrix} \\
\mathbf{y}_0 &= \begin{pmatrix} x_m + x_p\sqrt{\delta} \\ y_m - y_p\sqrt{\delta} \\ z_m + z_p\sqrt{\delta} \end{pmatrix}, \quad \mathbf{y}_1 = \begin{pmatrix} x_m - x_p\sqrt{\delta} \\ y_m + y_p\sqrt{\delta} \\ z_m - z_p\sqrt{\delta} \end{pmatrix} \\
\mathbf{z}_0 &= \begin{pmatrix} x_m + x_p\sqrt{\delta} \\ y_m + y_p\sqrt{\delta} \\ z_m - z_p\sqrt{\delta} \end{pmatrix}, \quad \mathbf{z}_1 = \begin{pmatrix} x_m - x_p\sqrt{\delta} \\ y_m - y_p\sqrt{\delta} \\ z_m + z_p\sqrt{\delta} \end{pmatrix}
\end{aligned} \tag{3.16}$$

with

$$\begin{aligned}
x_m &= \frac{\begin{array}{l}(c_{111}-c_{011})(r-c_{000}) - (c_{110}-c_{010})(r-c_{001}) \\ +(c_{100}-c_{000})(r-c_{011}) - (c_{101}-c_{001})(r-c_{010})\end{array}}{2\left((c_{111}-c_{011})(c_{100}-c_{000}) - (c_{110}-c_{010})(c_{101}-c_{001})\right)} \\
y_m &= \frac{\begin{array}{l}(c_{111}-c_{101})(r-c_{000}) - (c_{110}-c_{100})(r-c_{001}) \\ +(c_{010}-c_{000})(r-c_{101}) - (c_{011}-c_{001})(r-c_{100})\end{array}}{2\left((c_{111}-c_{101})(c_{010}-c_{000}) - (c_{110}-c_{100})(c_{011}-c_{001})\right)} \\
z_m &= \frac{\begin{array}{l}(c_{111}-c_{110})(r-c_{000}) - (c_{101}-c_{100})(r-c_{010}) \\ +(c_{001}-c_{000})(r-c_{110}) - (c_{011}-c_{010})(r-c_{100})\end{array}}{2\left((c_{111}-c_{110})(c_{001}-c_{000}) - (c_{101}-c_{100})(c_{011}-c_{010})\right)}
\end{aligned}$$

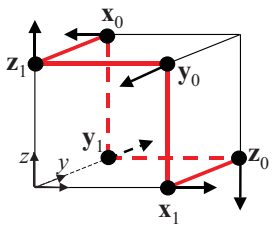


Figure 3.11: The points $(\mathbf{x}_0, \mathbf{y}_1, \mathbf{z}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{z}_1)$ on the contour which have normals either in $x-$, $y-$, or $z-$direction form a closed polygon on the edges of a box - the inner ring; this inner ring lies completely on the contour.

$$
\begin{aligned}
x_p &= \frac{1}{2\,((c_{111}-c_{011})(c_{100}-c_{000})-(c_{110}-c_{010})(c_{101}-c_{001}))} \\
y_p &= \frac{1}{2\,((c_{111}-c_{101})(c_{010}-c_{000})-(c_{110}-c_{100})(c_{011}-c_{001}))} \\
z_p &= \frac{1}{2\,((c_{111}-c_{110})(c_{001}-c_{000})-(c_{101}-c_{100})(c_{011}-c_{010}))} \\
\delta &= \alpha\, r^2 + \beta\, r + \gamma \\
\alpha &= (-c_{111}+c_{110}+c_{101}-c_{100}-c_{001}+c_{000}+c_{011}-c_{010})^2
\end{aligned}
\tag{3.17}
$$

$$
\begin{aligned}
\beta &= 2\,(c_{001}\,c_{110}+c_{011}\,c_{100}+c_{101}\,c_{010}+c_{111}\,c_{000}) \\
&\qquad \cdot(c_{111}+c_{000}+c_{101}+c_{110}+c_{100}+c_{010}+c_{011}+c_{001}) \\
&\quad -4\,(\,c_{000}\,c_{111}\,(c_{111}+c_{000})+c_{001}\,c_{110}\,(c_{110}+c_{001}) \\
&\qquad +c_{010}\,c_{101}\,(c_{101}+c_{010})+\,c_{011}\,c_{100}\,(c_{100}+c_{011})\,) \\
&\quad -4\,(c_{000}\,c_{110}\,c_{101}+c_{000}\,c_{011}\,c_{110}+c_{000}\,c_{011}\,c_{101}+c_{110}\,c_{101}\,c_{011}) \\
&\quad -4\,(c_{111}\,c_{001}\,c_{010}+c_{111}\,c_{100}\,c_{001}+c_{010}\,c_{100}\,c_{111}+c_{001}\,c_{010}\,c_{100}) \\
\gamma &= -\,(c_{001}\,c_{110}+c_{011}\,c_{100}+c_{101}\,c_{010}+c_{111}\,c_{000})^2 \\
&\quad +2\,(c_{111}^2\,c_{000}^2+c_{101}^2\,c_{010}^2+c_{001}^2\,c_{110}^2+c_{011}^2\,c_{100}^2) \\
&\quad +4\,(c_{000}\,c_{110}\,c_{101}\,c_{011}+c_{010}\,c_{100}\,c_{111}\,c_{001}).
\end{aligned}
$$

Depending on the value δ, all these six points are either real points on the contour, or all have imaginary values. If they are real (i.e. if $\delta > 0$), then the closed polygon $(\mathbf{x}_0, \mathbf{y}_1, \mathbf{z}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{z}_1)$ lies on the edges of a box. Note that this polygon lies completely on the contour[6]. We call this closed polygon the *inner ring* of the contour. Figure 3.11 gives an illustration.

[6]To show this, compute $s(x, y, z)$ using (3.4) for $(x, y, z)^T = (1-t)\,\mathbf{x}_0 + t\,\mathbf{y}_1$. It is a straightforward exercise in algebra to show that $s(x, y, z) = r$ for any $t \in \mathbb{R}$. This means that the line through $\mathbf{x}_0, \mathbf{y}_1$ lies on the contour. The remaining lines of the polygon $(\mathbf{x}_0, \mathbf{y}_1, \mathbf{z}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{z}_1)$ are treated in a similar way.

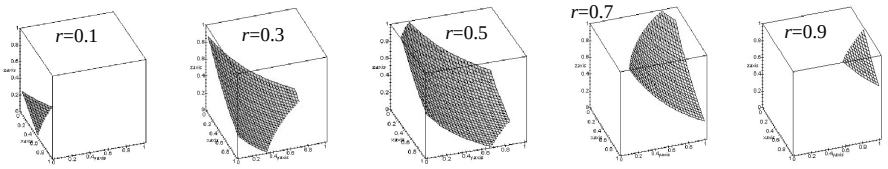


Figure 3.12: Contours of a cell C with $\text{seg}(C) = 1$: any contour in C consists of at most one connected surface part.

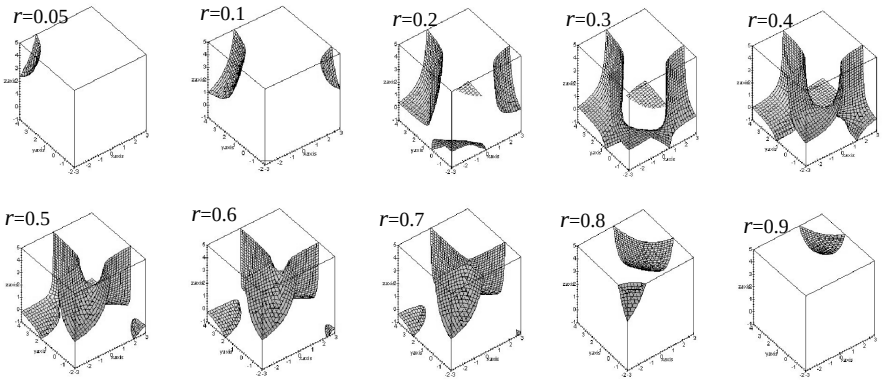


Figure 3.13: Contours of a cell C with $\text{seg}(C) = 4$: for $r = 0.2$ the contour consists of 4 unconnected surface parts.

3.5.1.4 Segment number of a cell

In this section we study all contours of (3.4) in the cell C_{000} following [191]. We are especially interested in the connectivity of the contours in the cell. Unfortunately the results of section 3.5.1.1 are not directly applicable here because one connected surface part may intersect C_{000} more than once.

To study all contours of (3.4) in C_{000} we apply – as in section 3.5.1.1 – the translation (3.6) of the local coordinate system and have to study the contours of (3.8) in the cell $C = [x_{\mathbf{p}_0}, x_{\mathbf{p}_0} + 1] \times [y_{\mathbf{p}_0}, y_{\mathbf{p}_0} + 1] \times [z_{\mathbf{p}_0}, z_{\mathbf{p}_0} + 1]$.

Varying the threshold r in (3.8), the contours change. So does the number of unconnected surface parts of the contour. We define

Definition 1 *Given the trilinear scalar field* $s(x, y, z) = a\,x + b\,y + c\,z + d\,x\,y\,z$ *in the domain of the cell* $C = [x_{\mathbf{p}_0}, x_{\mathbf{p}_0} + 1] \times [y_{\mathbf{p}_0}, y_{\mathbf{p}_0} + 1] \times [z_{\mathbf{p}_0}, z_{\mathbf{p}_0} + 1]$, *the* segment number $\text{seg}(C)$ *of* C *is the maximal number of unconnected surface parts of the contour* $s(x, y, z) = r = \text{const}$ *in* C *for any threshold* r.

Figure 3.12 gives an example of a cell with $\text{seg}(C) = 1$. Changing the value of r, the isosurface "moves" through the cell. It consists of at most one connected part for any r.

Figure 3.13 shows a cell with $\text{seg}(C) = 4$: for a particular threshold the contour consists of four unconnected parts.

The segment number is a threshold-independent characterization of a cell C. For any C we get $\text{seg}(C) \in \{1, 2, 3, 4\}$. To show this we use the fact that

a surface inside a cell has at least 3 intersections with the edges of the cell. Since the cell consists of 12 edges and at most one intersection of an edge and a contour exists, the maximal number of unconnected surface parts is 4.

The segment number may be used to estimate how complicated a surface extraction algorithm will be before picking a particular threshold. We are particularly interested in cells C with $\mathrm{seg}(C) = 1$. Since for them the contours always consist of one connected surface part, these cells are candidates for applying enhanced surface extraction methods or for merging them with adjacent cells (see section 3.8).

Now we give necessary and sufficient geometric conditions for a cell to have $\mathrm{seg}(C) = 1$. Again, we consider the contour of (3.8) in the cell $C = [x_{\mathbf{p}_0}, x_{\mathbf{p}_0} + 1] \times [y_{\mathbf{p}_0}, y_{\mathbf{p}_0} + 1] \times [z_{\mathbf{p}_0}, z_{\mathbf{p}_0} + 1]$.

To formulate the conditions for $\mathrm{seg}(C) = 1$, we need to introduce the concept of *characteristic hyperbolas*. The first characteristic hyperbola $\mathbf{h}_1$ in $\mathbb{R}^3$ consists of all points with $s_y(x, y, z) = 0$ and $s_z(x, y, z) = 0$ in (3.8). $\mathbf{h}_1$ can be written as rational quadratic Bézier curve described by two control vectors $\mathbf{b}_0^1$, $\mathbf{b}_2^1$ and a control point $\mathbf{b}_1^1$ (see [54]). We set

$$\mathbf{b}_0^1 = \begin{pmatrix} (-4\,b\,c)/d \\ 0 \\ 0 \end{pmatrix} \quad , \quad \mathbf{b}_1^1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad , \quad \mathbf{b}_2^1 = \begin{pmatrix} 0 \\ 1/b \\ 1/c \end{pmatrix} \quad , \quad w_1^1 = 1$$

where w_1^1 is the weight of $\mathbf{b}_1^1$. Then we obtain

$$\mathbf{h}_1(t) = \frac{\mathbf{b}_0^1\, B_0^2(t) + w_1^1\, \mathbf{b}_1^1\, B_1^2(t) + \mathbf{b}_2^1\, B_2^2(t)}{w_1^1\, B_1^2(t)}.$$

In a similar way we define the characteristic hyperbola $\mathbf{h}_2$ by $s_x(x, y, z) = 0$ and $s_z(x, y, z) = 0$. The characteristic hyperbola $\mathbf{h}_3$ is defined by $s_x(x, y, z) = 0$ and $s_y(x, y, z) = 0$. The Bézier point $\mathbf{b}_1^2$ with the corresponding weight w_1^2 and the control vectors $\mathbf{b}_0^2, \mathbf{b}_2^2$ describing $\mathbf{h}_2$ are

$$\mathbf{b}_0^2 = \begin{pmatrix} 0 \\ (-4\,a\,c)/d \\ 0 \end{pmatrix} \quad , \quad \mathbf{b}_1^2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad , \quad \mathbf{b}_2^2 = \begin{pmatrix} 1/a \\ 0 \\ 1/c \end{pmatrix} \quad , \quad w_1^2 = 1.$$

The hyperbola $\mathbf{h}_3$ is described by

$$\mathbf{b}_0^3 = \begin{pmatrix} 0 \\ 0 \\ (-4\,a\,b)/d \end{pmatrix} \quad , \quad \mathbf{b}_1^3 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad , \quad \mathbf{b}_2^3 = \begin{pmatrix} 1/a \\ 1/b \\ 0 \end{pmatrix} \quad , \quad w_1^3 = 1.$$

If $a\,b\,c\,d < 0$ then $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ intersect in two common points $\mathbf{s}_1$ and $\mathbf{s}_2$. They can be computed as

$$\mathbf{s}_1 = \frac{-1}{\sqrt{-a\,b\,c\,d}} \begin{pmatrix} b\,c \\ a\,c \\ a\,b \end{pmatrix} \quad , \quad \mathbf{s}_2 = \frac{1}{\sqrt{-a\,b\,c\,d}} \begin{pmatrix} b\,c \\ a\,c \\ a\,b \end{pmatrix}. \tag{3.18}$$

From (3.8) and (3.18) we obtain

$$s(\mathbf{s}_1) = \frac{-2\,a\,b\,c}{\sqrt{-a\,b\,c\,d}} \quad , \quad s(\mathbf{s}_2) = \frac{2\,a\,b\,c}{\sqrt{-a\,b\,c\,d}}. \tag{3.19}$$

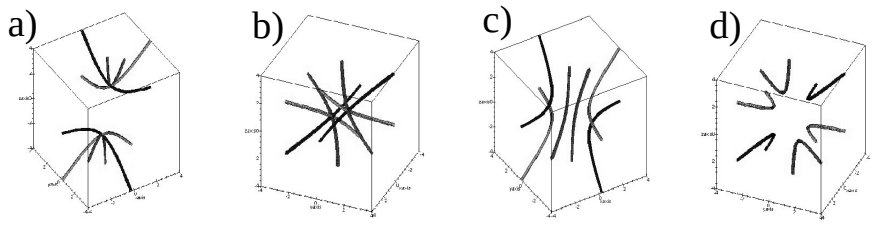


Figure 3.14: Location of characteristic hyperbolas $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$; a),b): $a\,b\,c\,d < 0$; c),d):$a\,b\,c\,d > 0$.

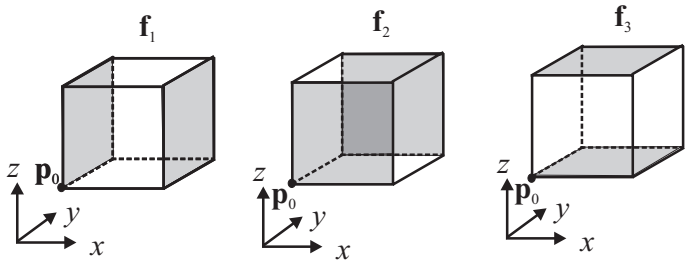


Figure 3.15: The faces $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3$ of a cell.

Figures 3.14a and b illustrate this situation from two different viewpoints. If $a\,b\,c\,d > 0$ then $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ do not have any intersections. Figures 3.14c and d show this from different viewpoints. The degenerate case $a\,b\,c\,d = 0$ is omitted here.

To formulate conditions for $\text{seg}(C) = 1$, we have to classify the faces of C. Given the cell $C = [x_{\mathbf{p}_0}, x_{\mathbf{p}_0}+1] \times [y_{\mathbf{p}_0}, y_{\mathbf{p}_0}+1] \times [z_{\mathbf{p}_0}, z_{\mathbf{p}_0}+1]$, let $\mathbf{f}_1 = \{(x, y, z) \in C : x = x_{\mathbf{p}_0} \text{ or } x = x_{\mathbf{p}_0} + 1\}$, $\mathbf{f}_2 = \{(x, y, z) \in C : y = y_{\mathbf{p}_0} \text{ or } y = y_{\mathbf{p}_0} + 1\}$, and $\mathbf{f}_3 = \{(x, y, z) \in C : z = z_{\mathbf{p}_0} \text{ or } z = z_{\mathbf{p}_0} + 1\}$. See figure 3.15 for an illustration of the faces.

Now we can formulate

Theorem 1 *Let* $C = [x_{\mathbf{p}_0}, x_{\mathbf{p}_0} + 1] \times [y_{\mathbf{p}_0}, y_{\mathbf{p}_0} + 1] \times [z_{\mathbf{p}_0}, z_{\mathbf{p}_0} + 1]$ *be a cell in the scalar field defined by* (3.8)*. Then the condition* $\text{seg}(C) = 1$ *is equivalent to the three conditions* $\mathbf{h}_1 \cap \mathbf{f}_1 = \emptyset$ *and* $\mathbf{h}_2 \cap \mathbf{f}_2 = \emptyset$ *and* $\mathbf{h}_3 \cap \mathbf{f}_3 = \emptyset$*.*

Figure 3.16 illustrates the idea of the proof. Suppose $\mathbf{h}_3$ intersects $\mathbf{f}_3$ as shown in Figure 3.16a. Figure 3.16b is a magnification of the cell and $\mathbf{h}_3$ in Figure 3.16a. We compute the intersection point of $\mathbf{h}_3$ and $\mathbf{f}_3$, and consider the contour passing through this point. As shown in Figure 3.16b, this contour consists of at least two surface parts. The cases that $\mathbf{h}_1$ intersects $\mathbf{f}_1$, or $\mathbf{h}_2$ intersects $\mathbf{f}_2$, are treated in a similar way.

For the proof of the converse statement of theorem 1, we assume that for a certain threshold r the contour consists of at least two unconnected surface parts. Then we can find a face of C which has two intersection curves with the contour. (In the worst case we have to vary r to find such a face.) (Figure 3.16c shows two surface parts of the contour which produce two intersection curves in the upper face of $\mathbf{f}_3$.) Then we can find a point on this face which is the

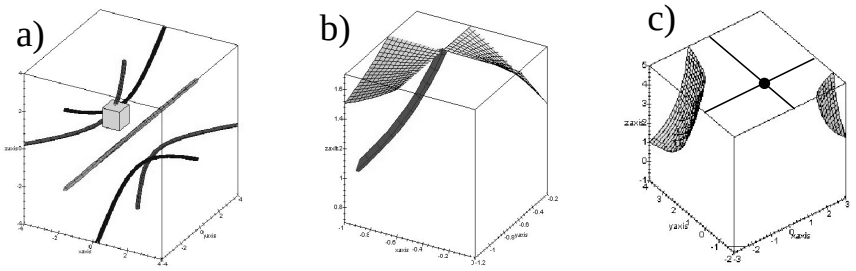


Figure 3.16: Proof idea of theorem 1.

intersection point with the corresponding characteristic hyperbola. (In Figure 3.16c, the marked point on the upper part of $\mathbf{f}_3$ is the intersection with $\mathbf{h}_3$.)

Theorem 1 gives geometric conditions for $\text{seg}(C) = 1$. This means that for a given cell C theorem 1 can be applied to distinguish between $\text{seg}(C) = 1$ and $\text{seg}(C) > 1$. Since only for $\text{seg}(C) = 1$ is it certain that no ambiguities appear for any threshold and any face of the cell, only cells with $\text{seg}(C) = 1$ are candidates for accelerated isosurface extraction algorithms. Thus we did not continue to search for geometric conditions for $\text{seg}(C) = 2$.

3.5.2 Graphical representation for piecewise trilinear contours

After studying properties of the piecewise trilinear contour in section 3.5.1, this section discusses algorithms to find graphical representations of the contour. We consider three approaches for doing this:

- voxel representation
- triangular representation
- representation as parametric surfaces

Each approach is discussed separately in one of the sections 3.5.2.1 – 3.5.2.3.

3.5.2.1 Voxel representation of piecewise trilinear contours

This approach creates voxels out of the scalar field by applying piecewise trilinear interpolations. The voxel size is usually smaller than the size of the grid cells $C_{i,j,k}$; it may be chosen adaptively. The voxels may be effectively rendered by using workstations with 3D frame buffers. The dividing cubes algorithm introduced in [126] is a representative of these approaches.

3.5.2.2 Triangular representation of piecewise trilinear contours

The representation of piecewise trilinear contours as triangular mesh is a natural and widespread approach because most of todays graphics workstations are highly specialized in processing large numbers of triangles.

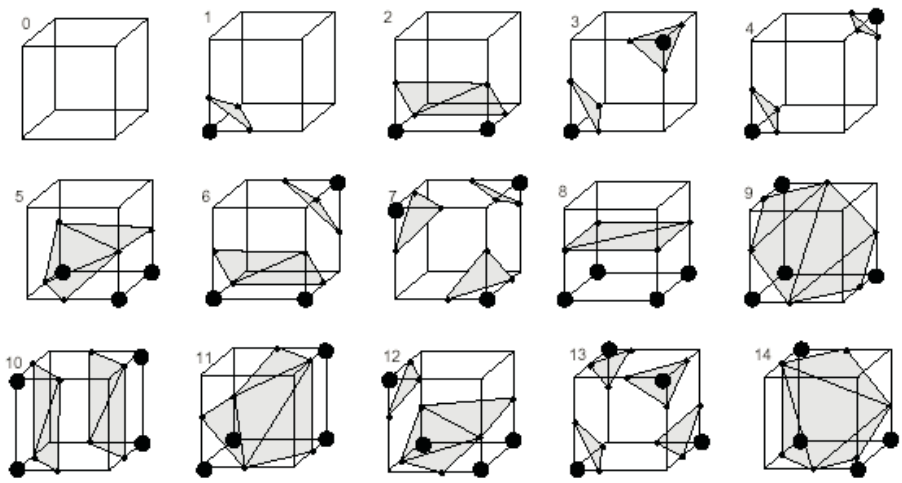


Figure 3.17: Cases of the Marching Cubes algorithm from [130]. A solid dot on a vertex means a "+" classification, i.e. the scalar value at this vertex is larger the considered threshold. Note that the cases 11 and 14 coincide due to symmetries.

Two general approaches exist to find a triangular approximation of the contour. The one is to find the contour lines in parallel slices and connect the contours of adjacent slices by applying a triangulation. Contributions to this approach can be found in [2], [135], [106] and [9].

The other general approach to extract contours works directly on the 3D cells. The standard algorithm here is the Marching Cubes (MC) algorithm introduced in [130]. Here, for every cell the scalar values at the vertices are checked for being smaller or larger than the picked threshold. Depending on the result of this check, each vertex of a cell is marked as "+" or "-". Considering symmetries in the cells, 14 different configurations of the "+" and "-" arrangement are possible. They are illustrated in figure 3.17. Based on this classification, the intersections of the contour with the edges of the cells are computed by applying linear interpolations along the edges of the cell. Finally, a triangulation which is built out of these intersection points gives the approximation of the contour (see figure 3.17).

Soon after the Marching Cubes algorithm was published it was realized that it does not necessarily give topologically exact representations of the contour ([47]). In fact, it may happen that adjacent cells create different contour curves on the face they share. In these cases the global contour has a hole in the area of these two adjacent cells. This may appear when one face of a cell has exactly two vertices with a "+" classification which are located diagonally to each other. In this case, four intersections of the contour with the edges of the face exist. There are two ways of connecting these four intersection points. One approach to solving these ambiguities was introduced in [139]. There it is shown that the intersection of the contour with a face of a cell is a hyperbola. Similar to algorithms for isoline extraction of 2D scalar data, its correct representation can be found by checking an additional point inside the face. Depending on the scalar value of this point the topological correct representation of the contour

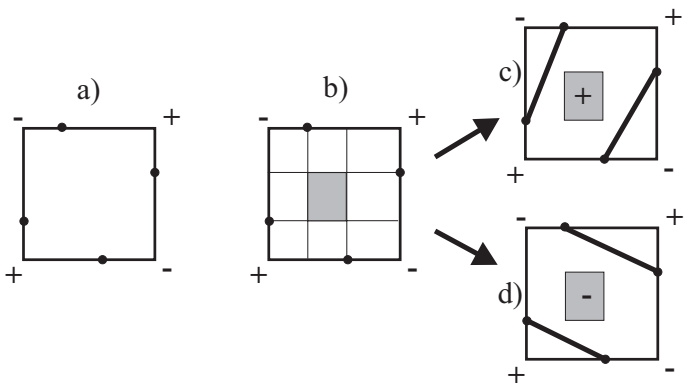


Figure 3.18: Solving ambiguities for the intersection of a trilinear contour and a cell [139]. a) If exactly two opposite vertices have a scalar value bigger than the threshold, exactly four intersections of the contour with the edges of the face exist. b) Depending on the scalar value of any point in the area marked grey, the correct representation of the contour is either c) or d).

curve can be found. Figure 3.18 illustrates this.

Applying this approach to all ambiguous faces of the cell, the result is up to four closed polygons on the faces (see [139]). To get the representation of the contours, these polygons are triangulated independently of each other. A similar approach to solving the ambiguities on the cell faces was introduced in [197]. Another way of preventing unwanted holes in the contour across the cell faces is suggested in [134]. The approach reported there is faster than [139] but does not yield topologically exact contour curves on the faces of the cell. A survey of algorithms to treat ambiguities on the cell faces can be found in [144]. One approach to combine a piecewise trilinear interpolation with a tetrahedrization of the cells is introduced in [212]. There the fact is used that a tetrahedral edge which lies on a face of a cell may have up to two intersections with the contour. Thus more involved lookup tables than shown in figure 3.6 are introduced. They make sure that the topology of the approximation of the contour does not depend on the particular tetrahedrization.

The extension in [139] of the Marching Cubes algorithm guarantees a topologically exact representation of the contour curves on the faces of a cell. Unfortunately it does not guarantee a topologically exact representation of the isosurface inside a cell [7]. Consider figure 3.19 for an example.

One approach to overcome this problem is suggested in [138]. There the topology of the contour inside a cell is described by the "connectivity" of the cell vertices: two vertices are connected if there is a curve inside the cell (or on its boundary) which connects the vertices and does not intersect the contour. In order to find the pairs of vertices which are connected inside the cell (i.e. the connecting curve is not on the cell boundaries), the scalar values of the vertices are additionally checked against the scalar value of a "body saddle point" inside the cell. This point is one of the points $\mathbf{s}_1, \mathbf{s}_2$ in (3.18). Since both points $\mathbf{s}_1, \mathbf{s}_2$ may be located inside a cell or both points may not exist at all, the approach in [138] does not work for any case. Also, [138] does not give any information

[7] The same statement applies for [212].

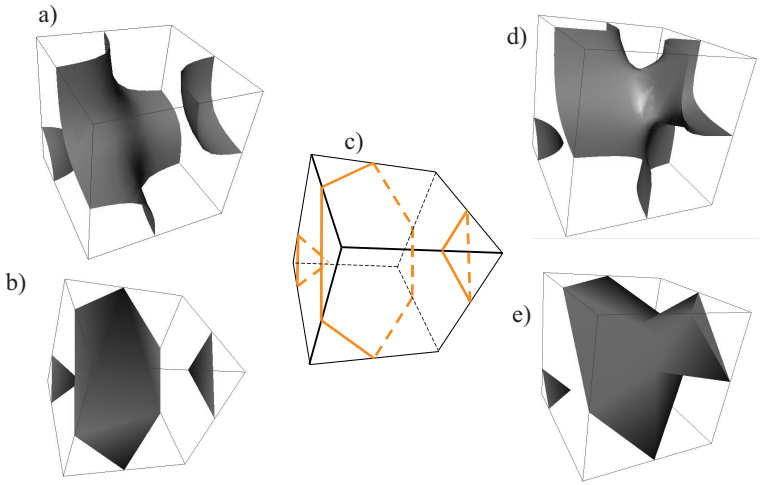


Figure 3.19: a) and d): two possible contours of (3.4) and (3.5) where the MC algorithm of [130] and [139] gives the same set of closed polygons on the faces of the cell shown in c); depending on certain inner points, the exact triangulation is either b) or e).

to actually apply a topological exact triangulation.

From figure 3.19 it is obvious that in order to get a guaranteed topologically exact representation of the contour, certain inner points of the cell have to be incorporated into the triangulation. In the following we want to find a set of inner points which are sufficient to get a topologically exact triangulation for every case. To explain the main idea we start with an example. Given is the contour shown in figure 3.20a. The MC algorithm of [130] and [139] gives the closed polygon shown in figure 3.20b. We name the vertices of the polygon $\mathbf{v}_1, ..., \mathbf{v}_6$. Triangulating this polygon, the edges $(\mathbf{v}_2, \mathbf{v}_6)$ and $(\mathbf{v}_3, \mathbf{v}_5)$ must not be used because the solution for the ambiguities on the upper face of the cell had excluded these edges from a valid triangulation. Here it makes sense to define one inner point $\mathbf{v}$ on the contour and apply a triangulation shown in figure 3.20d. A good candidate for $\mathbf{v}$ is the point on the contour which has a contour surface normal in z-direction (see figure 3.20a).

The example of figure 3.20 gives the key to find a set of inner points which are sufficient for a topological exact triangulation for every case. It turns out that this set of inner points is the inner ring introduced in section 3.5.1.3. A topologically exact Marching Cubes algorithm can be described in the following way:

1. Create the closed polygons on the cell faces following [130] and [139]. We obtain up to four closed polygons and call them *outer rings*.

2. Compute the points of the inner ring by applying (3.16) and (3.17).

3. If the inner ring is not real or if the inner ring is completely outside the cell, then triangulate the outer rings independently of each other; otherwise continue with 4.

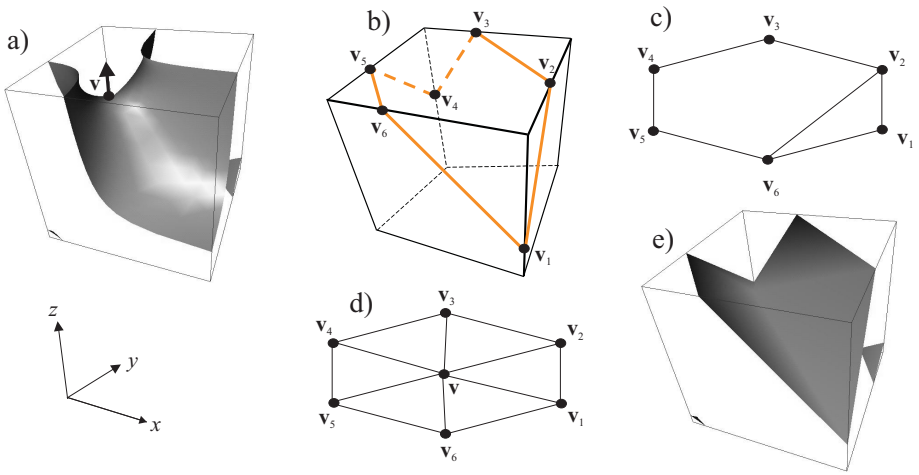


Figure 3.20: a) contour and point $\mathbf{v}$ with normal in $z-$direction on it; b) closed polygons resulting from the MC algorithm of [130] and [139]; c) part of a wrong triangulation of b); d) triangulation applied here; e) triangulation of d) in 3D.

4. Check the connectivity between the inner ring and each of the outer rings. If the inner ring and one of the outer rings belong to the same contour segment: triangulate the area between the inner ring and this outer ring.

5. If only one outer ring was connected to the inner ring, the inner ring itself has to be triangulated.

Figure 3.21 illustrates this algorithm.

To check the connectivity between the inner ring and one outer ring, we intersect the lines of the inner ring with the faces of the cell. If one of these intersection points lies on the outer ring, it is connected to the inner ring. Figure 3.21c illustrates this.

Figures 3.22-3.24 show examples of the application of the topologically exact Marching Cubes algorithm. Although we have not yet introduced how to compute the exact contours shown in the figures 3.19a, 3.19b, 3.20a, 3.22, 3.23d–f, 3.24a, we have inserted the images here to provide a comparison of the exact contours and the MC triangulations. The representation of the exact contours is treated in section 3.5.2.3. In the examples of figures 3.22 and 3.24 the MC algorithm of [130] and [139] fails, i.e. gives topologically wrong triangular approximations. In fact, the result for figure 3.22 would be a (topologically wrong) triangulation similar to figure 3.19b. Also [138] does not provide a topologically exact triangulation for figure 3.22 because it does not use any inner points for triangulation.

Improvements of the Marching Cubes Algorithm

The Marching Cubes algorithm is popular for finding a triangular approximation of contour. Together with extensions in [139] and in this section, it is able to triangulate any piecewise trilinear contour topologically exact. Nevertheless there are disadvantages of the algorithm as well:

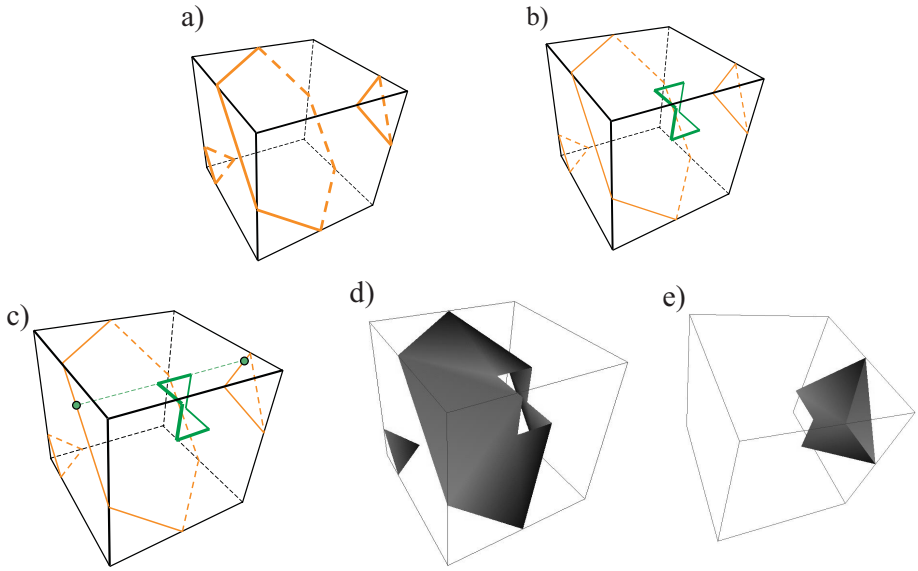


Figure 3.21: Illustration of a topologically exact MC algorithm; a) create outer rings following [130] and [139]; b) compute inner ring; c) check connectivity between inner ring and outer rings by intersecting the lines of the inner ring with all faces of the cell; here the inner ring is connected to two outer rings; d), e) triangulate the areas between inner ring and outer rings.

- since every cell has to be treated separately the algorithm is rather time consuming
- the resulting triangular mesh might be too fine (see figure 3.25a[8] for an example)

Several solutions have been proposed to overcome these disadvantages.

If the number of triangles is too high (i.e. the triangular mesh is too fine) a variety of mesh reduction algorithms exist. The treatment of mesh reduction algorithms is not the subject of this work. Contributions on that area are in [68], [93], [166], [154] and [94].

Another popular approach to deal with large triangular meshes is to find a multiresolution representation of the mesh (see [52] and [75] for examples). Since this approach also does not make use of CAGD applications, it is not treated in this work. Note that both mesh reduction algorithms and multiresolution representations of triangular meshes are in general not only for MC triangulations but for arbitrary triangular meshes.

One approach to reducing the number of triangles directly in the MC algorithm is the discrete MC approach in [133]. There only the midpoints of the cell edges are used to build the triangulation. This way a number of triangles from the original MC algorithm may collapse to bigger polygons.

To speed up the MC algorithm itself, storing the data in an octree has been proven to be useful. In [208] the volume data is stored in a modified octree called branch-on-need octree (BONO). In each node of the octree the minimal

[8]The data set in this image is property of Siemens Medical Systems, Inc., Iselin, NJ

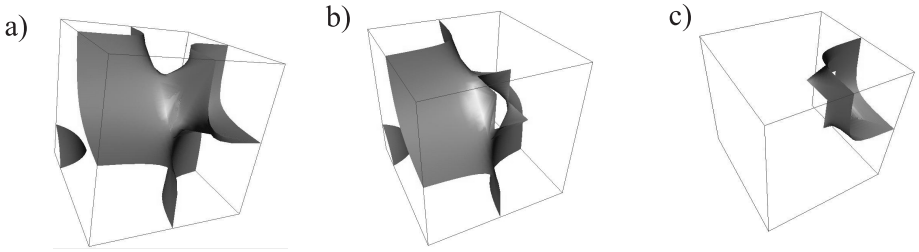


Figure 3.22: a) Exact contour over the triangulation shown in figure 3.21d) and 3.21e); b) exact contour over triangulation in figure 3.21d); c) exact contour over triangulation in figure 3.21e). We can clearly see that inner ring is part of the contour.

and maximal scalar value for all cells represented by this node are stored. For a certain threshold the tree is traversed top down to exclude those cells from the MC algorithm through which the contour does not pass. Furthermore, this approach can also be used to reduce the number of triangles produced by the MC algorithm. This is done in [170] by merging the octree leaves in a suitable way. An extension of the octree approach to time varying fields can be found in [181].

Other approaches to speeding up the MC algorithm use seed cells. A seed cell is a cell which a certain contour passes through. Starting from it, only the adjacent cells which the contour may pass through are searched until the global contour is completely extracted. The problem here is finding an appropriate number of seed cells for a ceratin threshold. In [127] each cell is considered as a 2D point in a "span space" where the x-coordinate is the minimal scalar value and the y-coordinate is the maximal scalar value of the cell. Then the seed cells can be effectively obtained out of this span space. In [172] a grid resolution of the span space is introduced to speed up the search for the seed cells. Other methods which focus on finding one seed cell instead a complete set of seed cells for a threshold can be found in [102], [103], [7], [119]. An extension of the seed cell idea to time varying fields can be found in [171]. A case study on the performance of the improvements of the MC algorithm can be found in [182].

Another approach to speeding up the MC algorithm and reducing the number of triangles is treated in section 3.8. Since the approach described there uses other interpolation schemes of the volume data, it was put into a section of its own.

3.5.2.3 Representation of piecewise trilinear contours as parametric surfaces

Applying the Marching Cubes algorithm (or any other algorithm which gives a triangular approximation of the contour), the resulting triangular mesh may not only be too fine as treated in section 3.5.2.2, it may also be too coarse. This problem appears

- with low resolution volume data
- when exploring details in high resolution volume data.

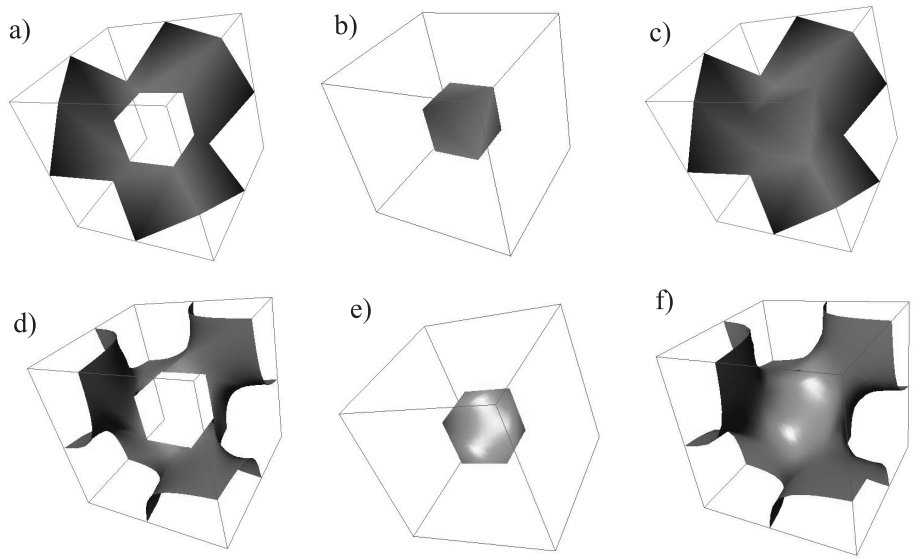


Figure 3.23: Example of a contour with one outer ring consisting of 12 edges and the inner ring being completely inside the cell; a) triangulation between inner ring and outer ring; b) triangulation of inner ring; c) whole triangulation; d)-f) exact contours over the triangulations a)-c).

Figure 3.25a shows an example of a volume data set where MC gives a triangular mesh which is too fine. Figure 3.25b shows a detail of the inner part of the surface shown in figure 3.25a. Here the mesh is too coarse.

One way to get a finer representation of the contour is to use a more detailed (and therefore larger) volume data set. Since this is usually not available, we try to find better approximations of the piecewise trilinear contour than the MC algorithm yields. In [5] a refined triangular representation of the contour is achieved by adaptively refining the triangles of the MC algorithm.

Another promising approach for a better representation of a piecewise trilinear contour is representation as piecewise parametric surfaces instead of triangles. Several approaches for doing this exist.

In [66], the contour is approximated by a number of bicubic patches. The approximated surface is G^0 continuous across the cell boundaries. Piecewise bicubic patches are also used in [105] to refine the results of a "contouring-and-connecting" approach. In [73], the contour is approximated using patches with 4, 5 or 6 boundary curves. The result is a surface which is G^0 continuous across the cell boundaries as well. [84] approximates the contours by rational quadratic triangular Bézier surface patches. This approach represents the boundary curves of the contour on the faces of the cells exactly but yields only G^0 continuous junctions of the patches both inside a cell and across the cell boundaries.

All the approaches mentioned above have something in common: each is just another approximation of the piecewise trilinear contour. In the following we want to find the exact representation of the trilinear contour as a piecewise parametric surface. We know from section 3.5.1.2 that the contour can be described in terms of rational cubic Bézier triangles. The general idea of describing the

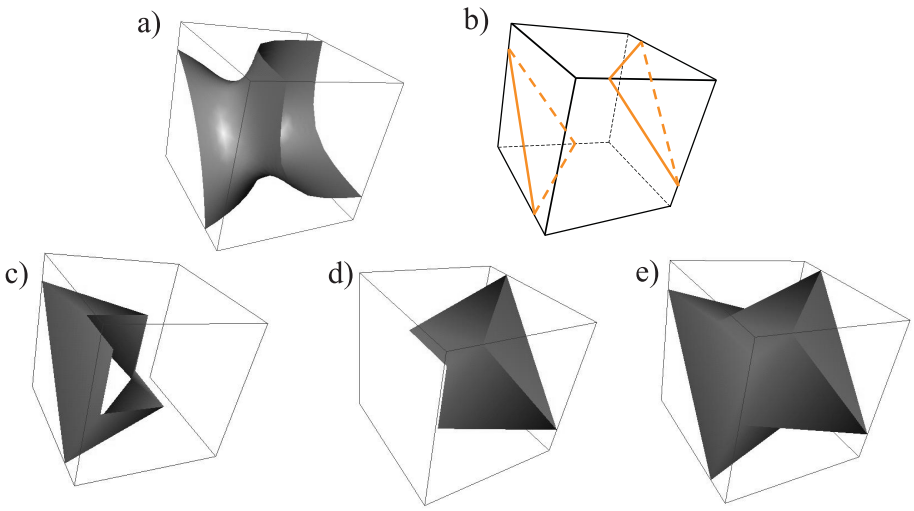


Figure 3.24: a) Contour which gives two outer rings and the inner ring completely inside the cell; b) outer rings; c) triangulation between inner ring and one outer ring; d) triangulation between inner ring and the other outer ring; e) whole triangulation.

whole contour in a cell is to build a triangular patch over each triangle of the MC triangulation.

Unfortunately, triangular patches of adjacent triangles may have gaps. Figure 3.10c gives an example. To overcome this problem, we use trimmed surfaces (see [55]) of the triangular rational cubics instead of the triangular patches themselves. This way the domain of the patches is not a triangle but a more complex shape which is bounded by three rational cubic curves. We use the following algorithm to compute the surface patch over an MC triangle:

Given is the triangle $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ which is obtained from the MC algorithm. This means that $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are on the contour.

1. Determine the projection directions $q_{\mathbf{ab}}, q_{\mathbf{bc}}, q_{\mathbf{ca}} \in \{x, y, z\}$ of the boundary curves.

2. Determine the projection direction $q_{\mathbf{abc}} \in \{x, y, z\}$ of the whole triangle.

3. Project the boundaries of the triangle $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ in the directions defined in step 1 onto the contour. We obtain the curves

$$\begin{aligned} \mathbf{x}_{\mathbf{ab}}(t) &= \mathbf{p}_{q_{\mathbf{ab}}}((1-t)\,\mathbf{a} + t\,\mathbf{b}) \\ \mathbf{x}_{\mathbf{bc}}(t) &= \mathbf{p}_{q_{\mathbf{bc}}}((1-t)\,\mathbf{b} + t\,\mathbf{c}) \\ \mathbf{x}_{\mathbf{ca}}(t) &= \mathbf{p}_{q_{\mathbf{ca}}}((1-t)\,\mathbf{c} + t\,\mathbf{a}) \end{aligned}$$

   on the contour. The curves $\mathbf{x}_{\mathbf{ab}}, \mathbf{x}_{\mathbf{bc}}, \mathbf{x}_{\mathbf{ca}}$ are the boundary curves of the final patch over the triangle $(\mathbf{a}, \mathbf{b}, \mathbf{c})$.

4. Project $\mathbf{x}_{\mathbf{ab}}, \mathbf{x}_{\mathbf{bc}}, \mathbf{x}_{\mathbf{ca}}$ in the direction $q_{\mathbf{abc}}$ into the plane defined by $\mathbf{a}, \mathbf{b}, \mathbf{c}$. We obtain the curves $\mathbf{y}_{\mathbf{ab}}, \mathbf{y}_{\mathbf{bc}}, \mathbf{y}_{\mathbf{ca}}$ in the plane $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ which are the boundary curves of the domain of the final patch.

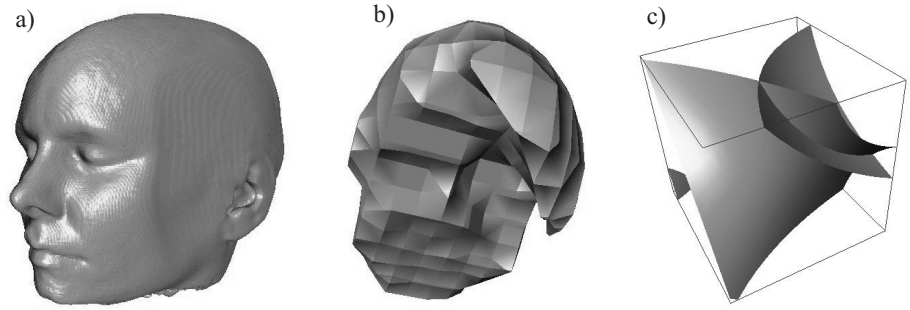


Figure 3.25: a) CT head consisting of 423.963 triangles - a candidate for mesh reduction algorithms; b) detail inside the same CT head - the triangular mesh is too coarse; c) the contour of a triquadratically interpolated scalar field may have self- intersections and complicated topologies.

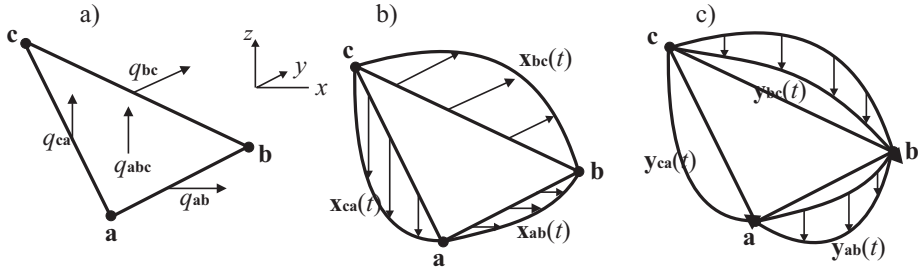


Figure 3.26: Example of the algorithm for constructing a triangular patch over an MC triangle $(\mathbf{a}, \mathbf{b}, \mathbf{c})$; a) determine the projection directions; here we have chosen $q_{\mathbf{ab}} = x$, $q_{\mathbf{bc}} = y$, $q_{\mathbf{ca}} = z$, $q_{\mathbf{abc}} = z$; b) project the boundaries of the triangle onto the contour; here we obtain the boundary curves $\mathbf{x}_{\mathbf{ab}}(t) = \mathbf{p}_x((1-t)\,\mathbf{a} + t\,\mathbf{b})$, $\mathbf{x}_{\mathbf{bc}}(t) = \mathbf{p}_y((1-t)\,\mathbf{b} + t\,\mathbf{c})$, $\mathbf{x}_{\mathbf{ca}}(t) = \mathbf{p}_z((1-t)\,\mathbf{c} + t\,\mathbf{a})$ on the contour; c) project $\mathbf{x}_{\mathbf{ab}}, \mathbf{x}_{\mathbf{bc}}, \mathbf{x}_{\mathbf{ca}}$ in z-direction onto the plane defined by the triangle $(\mathbf{a}, \mathbf{b}, \mathbf{c})$; we obtain the planar curves $\mathbf{y}_{\mathbf{ab}}, \mathbf{y}_{\mathbf{bc}}, \mathbf{y}_{\mathbf{ca}}$ which are the boundaries of the domain of the trimmed surface.

5. Compute the trimmed surface of the projected patch $\mathbf{p}_{q_{\mathbf{abc}}}(u\,\mathbf{a} + v\,\mathbf{b} + w\,\mathbf{c})$ with $u + v + w = 1$. The domain of the trimmed surface is given by the boundary curves $\mathbf{y}_{\mathbf{ab}}, \mathbf{y}_{\mathbf{bc}}, \mathbf{y}_{\mathbf{ca}}$.

Figure 3.26 illustrates an example of this algorithm.

To complete the algorithm, we have to answer two questions:
a) how to choose the projection directions
b) how to make sure that the cubic surfaces have all positive (or all negative) weights, i.e. no zeros in the denominator functions.

To a): Given the points $\mathbf{a} = (x_{\mathbf{a}}, y_{\mathbf{a}}, z_{\mathbf{a}})^T$ and $\mathbf{b} = (x_{\mathbf{b}}, y_{\mathbf{b}}, z_{\mathbf{b}})^T$, we choose $q_{\mathbf{ab}} = x$ if $\|x_{\mathbf{a}} - x_{\mathbf{b}}\| < \|y_{\mathbf{a}} - y_{\mathbf{b}}\|$ and $\|x_{\mathbf{a}} - x_{\mathbf{b}}\| < \|z_{\mathbf{a}} - z_{\mathbf{b}}\|$ and $\mathbf{a}$ and $\mathbf{b}$ are not on the same face of the MC cell, i.e. $\neg((x_{\mathbf{a}} = 0$ and $x_{\mathbf{b}} = 0)$ or $(x_{\mathbf{a}} = 1$ and $x_{\mathbf{b}} = 1))$. If the last named condition is false, the projection direction has to be chosen between y and z.

To compute $q_{\mathbf{abc}}$, we consider the normal $\mathbf{n} = (x_{\mathbf{n}}, y_{\mathbf{n}}, z_{\mathbf{n}})^T$ of the triangle $\mathbf{a}, \mathbf{b}, \mathbf{c}$. We choose $q_{\mathbf{abc}} = x$ if $\|x_{\mathbf{n}}\| > \|y_{\mathbf{n}}\|$ and $\|x_{\mathbf{n}}\| > \|z_{\mathbf{n}}\|$.

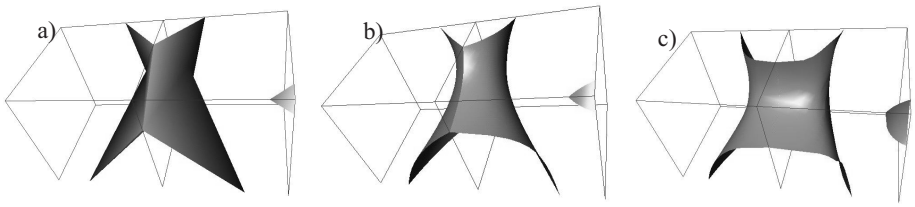


Figure 3.27: a) two cells and the triangular approximation of a certain contour using MC; b) the exact contour represented by a number of trimmed surfaces of rational cubic triangular patches; c) the global G^1 modification of the contour without changing its topology.

To b): To avoid zeros in the denominator functions of the rational patches, we have to make sure that the triangular approximation of the contour obtained by the MC algorithm is topologically equivalent to the contour itself. This is given by the extension of the MC algorithm of [130] and [139] which is treated in section 3.5.2.2 .

Examples of the exact representation of the trilinear contour as described in his section can be found in in the figures 3.19a, 3.19d, 3.20a, 3.22, 3.23d–f, 3.24a, 3.27b, 3.33b – 3.39b.

Another way of representing a piecewise trilinear contour exactly is a description as subdivision surface as done in [31]. Assuming a topologically exact Marching Cubes triangulation, this approach gives similar visual results to the parametric approach shown in this section.

3.6 Higher Order Polynomial Interpolation

The exact contour of a piecewise trilinear contour is G^∞ continuous inside a certain cell but only G^0 continuous across the cell boundaries. Figure 3.27b gives an example where the discontinuities across the boundary face of two cells is clearly visible. These discontinuities have a significant influence on the final shapes as shown in figures 3.33 – 3.39. In fact, for larger data sets the improvements from the MC triangulation (3.33a – 3.39a) to the exact contour (3.33b – 3.39b) are only marginal.

The discontinuities of the contours of the cell boundaries are due to the fact that the underlying interpolated piecewise trilinear scalar field is only C^0 continuous itself. Applying a smoother interpolation of the scalar field makes the contours smoother as well. Unfortunately, a higher order polynomial interpolation of the scalar field may destroy the topology of the contours. The contours obtained this way may have self-intersections and complicated topologies. Figure 3.25c gives an example of the contour of triquadratically interpolated scalar field. The self-intersections here indicate that standard algorithms like MC are not applicable here. In fact, no algorithm seems to be known that correctly deals with higher order interpolations of volume data.

In [197] a piecewise tricubic interpolation of the cells is suggested. In this way the authors obtain a globally C^1 continuous scalar field (and thus G^1 continuous contours). Beside the fact that piecewise tricubic interpolation is rather time consuming, no triangulation scheme is known for this kind of scalar field.

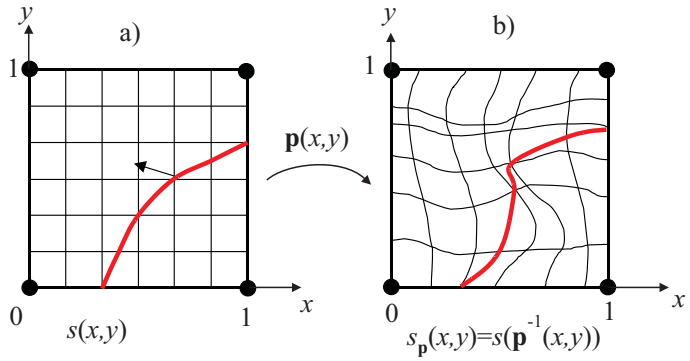


Figure 3.28: a) 2D bilinear scalar field in the domain of a cell; shown are isoparametric lines of the domain and one contour curve; b) apply a regular reparametrization $\mathbf{p}(x, y)$ of the domain onto itself: isoparametric lines and contour line may change their shape but not their topology.

In [22] 3D blending functions are used to eliminate the "overshooting effect" of the piecewise tricubic interpolation. Here too, appropriate surface extraction algorithms are not available.

In [8], the application of higher order interpolations was studied for 2D scalar fields. Here bicubic interpolations of the scalar field were used. To preserve the topology of the original scalar field, "damped partial derivatives" were used to restrict the interpolation parameters.

Due to the problems of higher order interpolation described above we introduce another approach here to get a globally G^1 continuous scalar field. This approach is described in section 3.7.

3.7 Piecewise Trilinear Interpolation with Local Reparametrization

We continue to search for globally G^1 interpolations of the scalar field which ensures that the resulting contours are G^1 as well. As an additional condition we demand that the resulting contours always have the same topology as the contours of the piecewise trilinear scalar field. This makes sure that standard algorithms such as MC are still applicable.

We achieve this by applying a local reparametrization of the domain of each cell. Figure 3.28 gives an example for a reparametrization in 2D.

Given is a trilinear scalar field $s(x, y, z)$ defined by (3.4) in the domain $C_{000} = [0, 1]^3$. The definition of a continuous one-to-one map

$$\begin{array}{rccl} \mathbf{p}: & [0,1]^3 & \rightarrow & [0,1]^3 \\ & (x,y,z) & \rightarrow & \mathbf{p}(x,y,z) = (x_{\mathbf{p}}(x,y,z),\ y_{\mathbf{p}}(x,y,z),\ z_{\mathbf{p}}(x,y,z)) \end{array}$$

creates a new scalar field $s_{\mathbf{p}}$ over $[0, 1]^3$:

$$s_{\mathbf{p}}(x, y, z) = s(\mathbf{p}^{-1}(x, y, z)).$$

If a point (x, y, z) lies on the contour $s(x, y, z) = r$, the point $\mathbf{p}(x, y, z)$ lies on the contour $s_{\mathbf{p}}(x, y, z) = r$. Thus the contours of $s_{\mathbf{p}}(x, y, z) = r$ can be computed

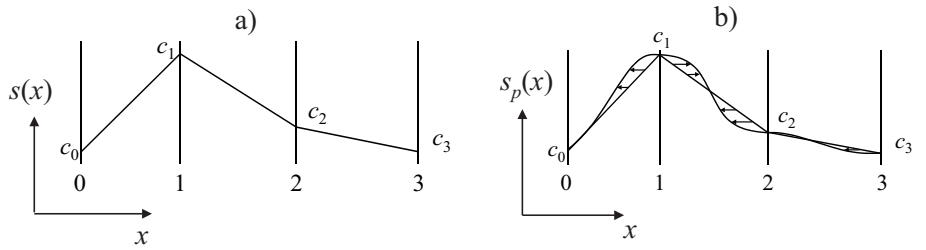


Figure 3.29: a) the piecewise linear curve $(x, s(x))^T$ is G^0 continuous; b) the curve $(p(x), s(x))^T$ is G^1 continuous, it has the same shape (but another parameterization) as the curve $(x, s_p(x))^T$.

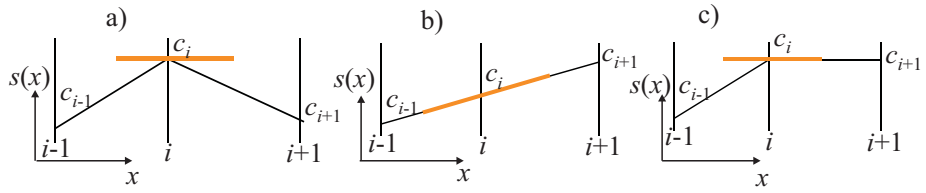


Figure 3.30: Estimate the rise $\dot{c}_i$ in the point $x = i$; a) $\dot{c}_i = 0$ if $(c_i - c_{i-1})$ and $(c_{i+1} - c_i)$ have opposite sign; b) $\dot{c}_i = c_i - c_{i-1}$ if $c_i - c_{i-1} = c_{i+1} - c_i$; c) $\dot{c}_i = 0$ if $c_i = c_{i-1}$ or $c_{i+1} = c_i$.

in a simple way: apply the map $\mathbf{p}$ to all contour points of $s(x, y, z) = r$. Since $\mathbf{p}$ is continuous and one-to-one, the contours of s and $s_{\mathbf{p}}$ have the same topology for any r.

We have to find appropriate maps $\mathbf{p}$ for each cell of the piecewise trilinear scalar field which makes it globally G^1 continuous. This way all contours are G^1 continuous as well.

We consider an 1D example to explain how to choose the maps $\mathbf{p}$. (Since in the 1D example $\mathbf{p}$ is a scalar function from $\mathbb{R}$ to $\mathbb{R}$, we simply write p instead of $\mathbf{p}$ here.) Given are the scalar values c_i which define a piecewise linear 1D scalar field

$$s(x) = (1 - t)\, c_i + t\, c_{i+1} \quad \text{with} \quad i = [x]\ ,\ t = x - [x].$$

See figure 3.29a for an illustration. The curve $(x, s(x))^T$ is piecewise linear and thus G^0 continuous. We have to find a domain reparametrization $p(x)$ in such a way that the curve $(p(x), s(x))^T$ is G^1 continuous, and $p(i) = i$, and p is continuous and one-to-one. Figure 3.29b illustrates this.

To define p, we first have to estimate the tangent directions of the curve $(p(x), s(x))^T$ at the junction points $x = i$. Let $\dot{c}_i$ be the estimated rise of s at $x = i$. We estimate $\dot{c}_i$ by

$$\dot{c}_i = \begin{cases} \frac{2\,(c_i - c_{i-1})(c_{i+1} - c_i)}{c_{i+1} + c_{i-1}} & \text{for} \quad (c_i - c_{i-1})(c_{i+1} - c_i) \geq 0 \\ \\ 0 & \text{else.} \end{cases} \tag{3.20}$$

Figure 3.30 illustrates special cases of this formula.

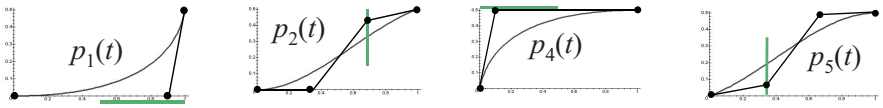


Figure 3.31: Auxiliary functions of formulas (3.21) for defining $p(t)$; the functions $p_1(t)$, $p_4(t)$ are reparametrization of parabola segments; $p_2(t)$, $p_5(t)$ are cubic functions.

Considering the interval $[i, i+1]$, and using a local parameter $t \in [0,1]$ in it, we obtain:

$$s(t) = (1-t)\, c_i + t\, c_{i+1} \quad , \quad \dot{c}_i = \frac{\dot{s}(0)}{\dot{p}(0)} \quad , \quad \dot{c}_{i+1} = \frac{\dot{s}(1)}{\dot{p}(1)}.$$

This gives the following conditions for the monotonous function $p(t)$ in $t \in [0,1]$:

$$p(0) = 0 \quad , \quad p(1) = 1 \quad , \quad \dot{p}(0) = \frac{c_{i+1} - c_i}{\dot{c}_i} \quad , \quad \dot{p}(1) = \frac{c_{i+1} - c_i}{\dot{c}_{i+1}}.$$

Since $\dot{p}(0)$ and $\dot{p}(1)$ can attain any value between 0 and $+\infty$, $p(t)$ cannot be described by a polynomial function of a fixed degree.

We construct a monotonous reparametrization function $p(t)$ out of four auxiliary functions $p_1(t), p_2(t), p_4(t), p_5(t)$. These functions are illustrated in figure 3.31. The function $p_1(t)$ is obtained by reparametrizing the parabola defined by the Bézier points $\mathbf{b}_0^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{b}_1^1 = \begin{pmatrix} x_1^1 \\ 0 \end{pmatrix}, \mathbf{b}_2^1 = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}$. This gives $p_1(t) = y_1({x_1}^{-1}(t))$ with $\begin{pmatrix} x_1(t) \\ y_1(t) \end{pmatrix} = \sum_{i=0}^{2} \mathbf{b}_i^1\, B_i^2(t)$. Moving x_1^1 between 0.5 and 1, $\dot{p}_1(1)$ ranges between 1 and $+\infty$. The function $p_2(t) = \sum_{i=0}^{3} y_i^2\, B_i^3(t)$ is a cubic with $y_0^2 = 0, y_1^2 = 0, y_3^2 = 0.5$. Moving y_2^2 between 1/6 and 0.5, $\dot{p}_2(1)$ ranges between 1 and 0. Furthermore we have $\dot{p}_1(0) = \dot{p}_2(0) = 0$.

While $p_1(t)$ and $p_2(t)$ are used to control the rise of p at $t = 1$, the functions $p_4(t)$ and $p_5(t)$ are used to control the rise at $t = 0$. The function $p_4(t)$ is a reparametrization of the parabola $\begin{pmatrix} x_4(t) \\ y_4(t) \end{pmatrix} = \sum_{i=0}^{2} \mathbf{b}_i^4\, B_i^2(t)$ with $\mathbf{b}_0^4 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{b}_1^4 = \begin{pmatrix} x_1^4 \\ 0.5 \end{pmatrix}, \mathbf{b}_2^4 = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}$ which gives $p_4(t) = y_4({x_4}^{-1}(t))$. Moving x_1^4 between 0 and 0.5, $\dot{p}_4(0)$ ranges between $+\infty$ and 1. The function $p_5(t) = \sum_{i=0}^{3} y_i^5\, B_i^3(t)$ is a cubic with $y_0^5 = 0, y_2^5 = 0.5, y_3^5 = 0.5$. Moving y_1^5 between 0 and 1/3, $\dot{p}_5(0)$ ranges between 0 and 1. Furthermore we have $\dot{p}_4(1) = \dot{p}_5(1) = 0$.

Now we can model the function $p(t)$ with $p(0) = 0$ and $p(1) = 1$ and given values for $\dot{p}(0)$ and $\dot{p}(1)$ between 0 and $+\infty$ as a linear combination of $p_1(t)$, $p_2(t)$, $p_4(t)$, $p_5(t)$:

$$
\begin{aligned}
p_1(t) &= \frac{1}{8}\left(\frac{1-2\,\dot{p}(1)+\sqrt{(1-2\,\dot{p}(1)\,)^2+4\,t\,\dot{p}(1)\,(1-\dot{p}(1)\,)}}{(1-\dot{p}(1)\,)}\right)^2\\
p_2(t) &= 3\,t^2\,(1-t)\,(\frac{1}{2}-\frac{1}{3}\,\dot{p}(1)\,)+\frac{1}{2}\,t^3\\
p_3(t) &= \begin{cases} p_1(t) & \text{for} \quad \dot{p}(1)>1\\ p_2(t) & \text{for} \quad 0\le\dot{p}(1)\le 1\end{cases}\\
p_4(t) &= \frac{2\,t\,\dot{p}(0)\,(1-\dot{p}(0))+(1-2\,\dot{p}(0))\,(1-\sqrt{1-4\,t\,\dot{p}(0)\,(1-\dot{p}(0))}\,)}{4\,(1-\dot{p}(0))^2}\\
p_5(t) &= t\,(1-t)^2\,\dot{p}(0)+\frac{3}{2}\,t^2\,(1-t)+\frac{1}{2}\,t^3\\
p_6(t) &= \begin{cases} p_4(t) & \text{for} \quad \dot{p}(0)>1\\ p_5(t) & \text{for} \quad 0\le\dot{p}(0)\le 1\end{cases}\\
p(t) &= p_3(t)+p_6(t).
\end{aligned}
\tag{3.21}
$$

Note that $p_1(t)$, $p_4(t)$ are reparametrization of a parabola segment while $p_2(t)$, $p_5(t)$ are cubic functions. Also note that

$$
\begin{aligned}
\lim_{\dot{p}(1)=1} p_1(t) = \frac{1}{2}\,t^2 \quad &, \quad \lim_{\dot{p}(1)=+\infty} p_1(t) = 1-\sqrt{1-t}-\frac{1}{2}\,t\\
\lim_{\dot{p}(0)=1} p_4(t) = t-\frac{1}{2}\,t^2 \quad &, \quad \lim_{\dot{p}(0)=+\infty} p_4(t) = \sqrt{t}-\frac{1}{2}\,t.
\end{aligned}
$$

Now we can apply this reparametrization to the 3D cells:

Given is a scalar field defined by (3.4) in the domain $C_{000}=[0,1]^3$. To find $\mathbf{p}(x,y,z)=(\,x_{\mathbf{p}}(x,y,z),\,y_{\mathbf{p}}(x,y,z),\,z_{\mathbf{p}}(x,y,z)\,)$, we consider the functions

$$
\begin{aligned}
c_{0x}(y,z) = s(0,y,z) \quad &, \quad c_{1x}(y,z) = s(1,y,z)\\
c_{-1x}(y,z) = s(-1,y,z) \quad &, \quad c_{2x}(y,z) = s(2,y,z)
\end{aligned}
$$

where c_{-1x} and c_{2x} are computed of the scalars adjacent to the cell considered here. The rise $\dot{c_{0x}}(y,z)$ on the cell face $x=0$ in x-direction is estimated by (3.20) using the values $c_{-1x}(y,z), c_{0x}(y,z), c_{1x}(y,z)$. In a similar way, the rise $\dot{c_{1x}}(y,z)$ on the cell face $x=1$ in x-direction is estimated by (3.20) using the values $c_{0x}(y,z), c_{1x}(y,z), c_{2x}(y,z)$. Then $x_{\mathbf{p}}(x,y,z)$ can be computed using (3.21) with

$$
t=x \;\;,\;\; \dot{p}(0)=\dot{c_{0x}}(y,z) \;\;,\;\; \dot{p}(1)=\dot{c_{1x}}(y,z) \;\;,\;\; p(t)=x_{\mathbf{p}}(x,y,z).
$$

Figure 3.32 gives an illustration. The reparametrization in $y-$ and $z-$direction, $y_{\mathbf{p}}(x,y,z)$ and $z_{\mathbf{p}}(x,y,z)$, are computed in a similar way.

The reparametrization introduced above is applied locally to all cells of the data volume. The global reparametrization $\mathbf{p}(x,y,z)$ obtained this way gives a globally G^1 scalar field $s_{\mathbf{p}}(x,y,z)$. Thus all contours of $s_{\mathbf{p}}$ are G^1 as well. See figure 3.27c for an example.

Also note that $s_{\mathbf{p}}$ preserves linearity: if s is linear in a certain cell (i.e. if the contours of s in this cell are plane segments), $s_{\mathbf{p}}$ is linear in this cell as well.

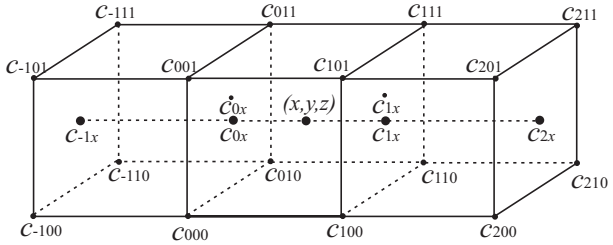


Figure 3.32: The reparametrization $x_{\mathbf{p}}(x, y, z)$ in x-direction is computed using $\dot{c_{0x}}$ and $\dot{c_{1x}}$. $\dot{c_{0x}}$ is estimated by c_{-1x}, c_{0x}, c_{1x} while $\dot{c_{1x}}$ is estimated by c_{0x}, c_{1x}, c_{2x}.

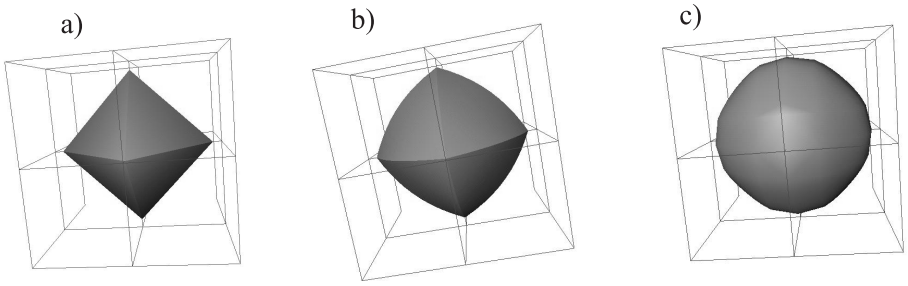


Figure 3.33: Scalar field $s(x, y, z) = x^2 + y^2 + z^2$, sampled by a $3 \times 3 \times 3$ grid in the domain $[-1, 1]^3$, $r = 0.9$; a) Marching cubes; b) exact contours; c) globally G^1 contours.

The isosurface extraction algorithm for the scalar field $s_{\mathbf{p}}$ is simple. We apply an isosurface extraction algorithm in s and apply $\mathbf{p}$ to all points of the contour.

Results:
We applied the algorithms to construct the exact contour of (3.4) and (3.5) (described in section 3.5.2.3) as well as its G^1-reparametrization (described in section 3.7) both to theoretical and practical data sets.

One result was already shown in figure 3.27. Figure 3.27a shows the MC triangulation of two adjacent cells for a certain threshold. Figure 3.27b shows its exact contour while figure 3.27c shows the result of the G^1 reparametrization. We can clearly see the higher continuity of the contour in figure 3.27c.

Figure 3.33 shows the scalar field $s(x, y, z) = x^2+y^2+z^2$ which is sampled by a $3\times3\times3$ grid in the domain $[-1, 1]^3$. Obviously the contours of s are concentric spheres. Figure 3.33a shows the result of the MC algorithm for $r = 0.9$: the sphere is approximated by 8 triangles. Figure 3.33b shows the exact contours of the piecewise trilinear interpolation. Although this shape comes closer to a sphere, we can still see discontinuities of the surface across the faces of the cell. Figure 3.33c shows the G^1 reparametrization of the exact contour of figure 3.33b. Note that this is not an exact sphere although it almost looks like one.

Figure 3.34 shows a $5 \times 5 \times 5$ hexahedral grid with random scalar values between 0 and 1 at the grid points, and $r = 0.5$. Figure 3.34a shows the result of the topologically exact Marching Cubes algorithm where the triangles inside a cell are Phong shaded. Figure 3.34b shows the exact contour of the scalar field.

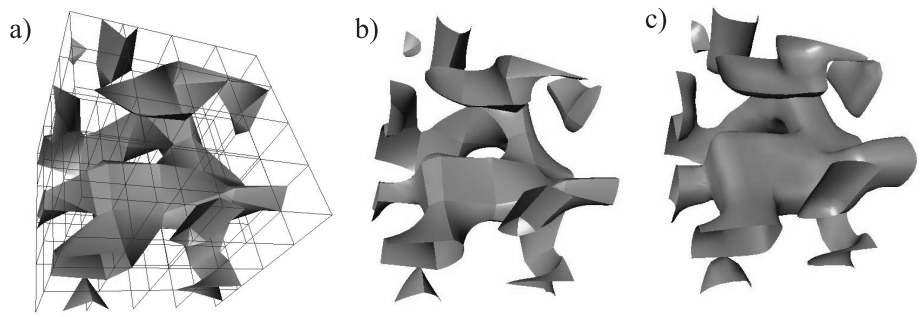


Figure 3.34: $5 \times 5 \times 5$ random volume data set; a) Marching cubes; b) exact contours; c) globally G^1 contours.

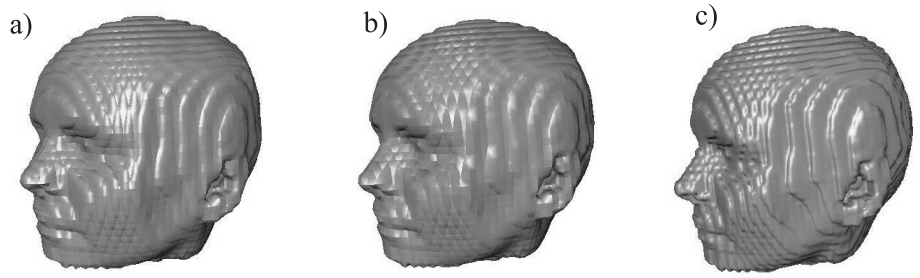


Figure 3.35: Downsampled data set of figure 3.25a ; a) Marching cubes; b) exact contours; c) globally G^1 contours.

We can clearly see the discontinuities of the surface across the cell faces. These discontinuities disappear in the G^1 reparametrization shown in figure 3.34c.

Figure 3.35 shows a downsampled version of the data set of figure 3.25a. Originally consisting of $256 \times 256 \times 109$ grid points, this version has only a $51 \times 51 \times 35$ grid resolution. The result of the MC algorithm shown in figure 3.35a consists of 93.636 triangles and shows clearly a "staircase effect" due to the low sample rate. The exact contour in figure 3.35b shows hardly any visual differences to figure 3.35a. The G^1 reparametrization of figure 3.35c looks smoother but still has the "staircase effects". In fact, figure 3.35c seems to emphasize the "staircase effects" more than 3.35a and 3.35b. This due to the fact that the "staircase effect" appears at locations of rapidly changing gradients of the scalar field. Figure 3.36 gives an illustration.

Figures 3.37 and 3.38 show inner details of the data set shown in figure 3.25a. There are only few visual differences between the MC results (figures 3.37a, 3.38a) and the exact contours (figures 3.37b, 3.38b). The G^1 reparametrizations (figures 3.37c, 3.38c) look significantly smoother.

Figure 3.39 shows a magnified detail of figure 3.38. Again the G^1 contour (figure 3.39c) looks smoother than the exact contour (figure 3.39b) and its MC approximation (figure 3.39a).

We conclude that in most cases the exact contour of a piecewise trilinear scalar field gives only slight visual improvements against the MC approximation if the MC triangles are rendered using Phong shading. The G^1 contour gives

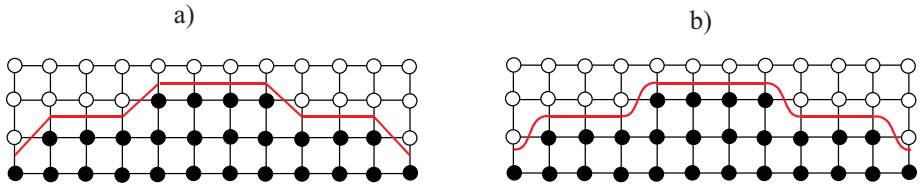


Figure 3.36: "Staircase effect": grid points marked with solid points have isovalue of 1, grid points marked with hollow points have isovalue of 0; relative to cells containing only scalar values of either 0 or 1, the gradient of the scalar field changes rapidly in cells which contain both vertices with 0 and 1 scalar values; a) piecewise linear approximation of isoline; b) the G^1 reparametrization smooths out the isosurface but does not eliminate the "staircase effect".

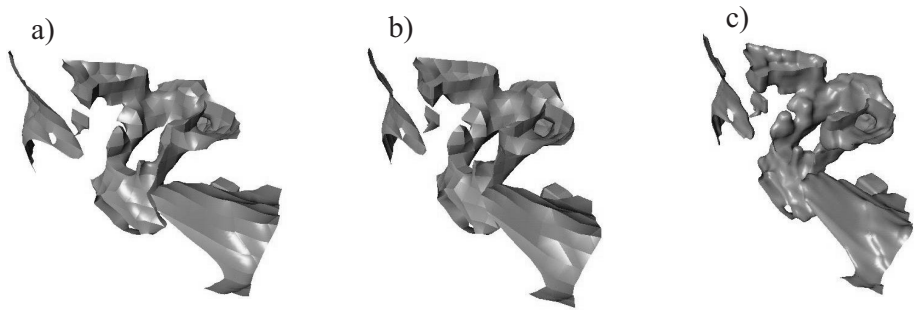


Figure 3.37: Inner detail of the data set of figure 3.25a; a) Marching cubes; b) exact contours; c) globally G^1 contours.

improved visual results for low resolution volume data.

Even if the G^1 reparametrization looks smoother, it is not able to remove the "staircase effects" in volume data (see figure 3.35). This is due to the fact that this effect appears at regions with high changes of the gradient of the scalar field, i.e. it is due to second order information of the scalar field. Since the G^1 reparametrization deals only with first order information, it cannot handle the "staircase effect".

The computation of the exact contours and their G^1 reparametrization is not useful for any volume data. In fact, the number of triangles produced by the Marching Cubes algorithm must not be too high because the exact contour increases the final number of triangles. Also, smoothness should be one of the desired quality criteria of the isosurfaces. This is the case for the example data set figure 3.35, 3.37-3.39 but may not appear for other data sets.

3.8 Piecewise Trilinear Interpolation of Larger Areas

In section 3.5.2.2 we treated algorithms to speed up the Marching Cubes algorithm and decrease the number of resulting triangles. The approaches introduced in this section have essentially the same goal. In contrary to the methods in section 3.5.2.2, this is achieved by applying different interpolation schemes of

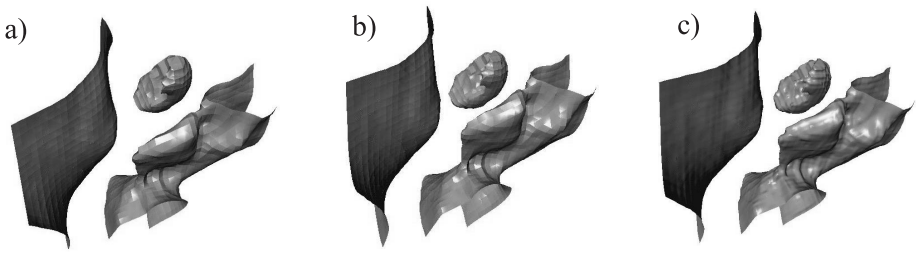


Figure 3.38: Inner detail of the data set of figure 3.25a; a) Marching cubes; b) exact contours; c) globally G^1 contours.

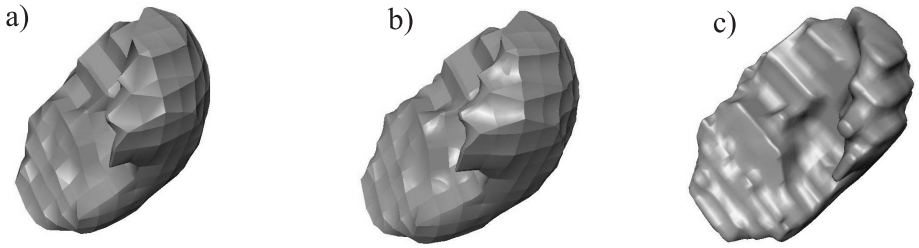


Figure 3.39: Detail of figure 3.38; a) Marching cubes; b) exact contours; c) globally G^1 contours.

the volume data before starting an isosurface extraction algorithm.

The main idea is to apply interpolations not on the cells but on certain larger areas.

One solution is to apply an adaptive tetrahedrization of the scalar field as introduced in [82] and [161]. There the scalar field is converted into an irregular tetrahedral grid. Different splitting criteria are discussed. The application of standard isosurface extraction algorithms like marching tetrahedra gives a coarser triangular representation of the scalar field than in the Marching Cubes case. Unfortunately, for the isosurface extraction on irregular tetrahedral grids, cracks in the approximated contour may appear. They can be prevented by a simultaneous subdivision of adjacent cells ([82]) or by using tetrahedral coons volumes instead of linearly interpolated tetrahedra ([92]).

In [136] and [179] a "splitting box algorithm" is introduced which performs the Marching Cubes algorithm not on the grid cells but on parallelepipedal collections of cells called boxes. Since a box generally consists of more than one cell, the number of boxes is smaller than the number of cells. Thus the application of the Marching Cubes algorithm to the boxes instead to the cells may give fewer triangles in a shorter time.

In [136] and [179] the structure of boxes was built top-down. Beginning with the whole scalar field as one box, bisections of the boxes were applied until a certain criterion was fulfilled or the cell level was reached. The bisection criterion there is a check that the isosurface computed in the box does not differ more than a cell distance from the isosurface computed by the cell-by-cell algorithm. Thus the subdivision criterion used there is threshold dependent. Changing the threshold, the structure of boxes has to be completely rebuilt. In [136] and [179] the performance of the splitting box algorithm is compared to the classical

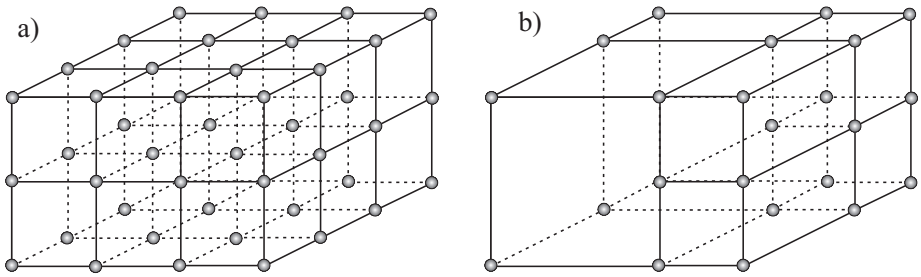


Figure 3.40: a) Trilinear interpolation in each cell for the MC algorithm; b) collect cells to bricks may speed up the MC algorithm.

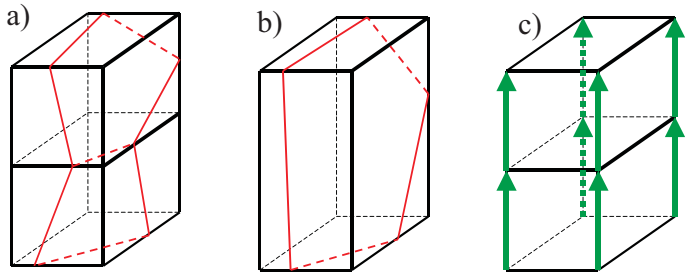


Figure 3.41: a) two cells and their contours for a certain threshold; b) merging the cells from a) and applying MC to the new brick gives a contour with the same topology as in a); c) monotony conditions for two cells to be mergeable.

Marching Cube algorithm. Unfortunately, no comparison was made with the improvements of the Marching Cubes algorithm discussed in section 3.5.2.2.

In [118] we introduced a similar approach to [136] and [179]. There the box structure is built bottom-up: cells are merged to boxes (called "bricks" in [118]). The merging criterion used there is threshold independent. Thus the brick structure is built in a preprocessing step and can be reused after changing the threshold. Figure 3.40 gives an illustration of a brick structure.

In [118] we allow a number of cells to be merged to a brick if the topology of the contour in the original cells and the brick coincides for every threshold. Figure 3.41a and b gives an example.

To give a geometric condition for two cells to be mergeable, we give the following

Theorem 2 *Let $s_1(x, y, z)$ be the piecewise trilinear scalar field defined in the domain $D = [0, 2] \times [0, 1] \times [0, 1]$ as illustrated in figure 3.42a. Furthermore, let $s_2(x, y, z)$ be the trilinear scalar field defined in D as illustrated in figure 3.42c. Then the contours of s_1 and s_2 have the same topology for any threshold if the following condition is satisfied:*

$$\begin{aligned} &(c_{000} < c_{100} < c_{200}) \ \wedge \ (c_{001} < c_{101} < c_{201}) \\ \wedge \quad &(c_{010} < c_{110} < c_{210}) \ \wedge \ (c_{011} < c_{111} < c_{211}). \end{aligned} \tag{3.22}$$

Proof: We have to find a reparametrization $r(x, y, z)$ of D which transforms the piecewise trilinear field of figure 3.42a into the scalar field illustrated in figure

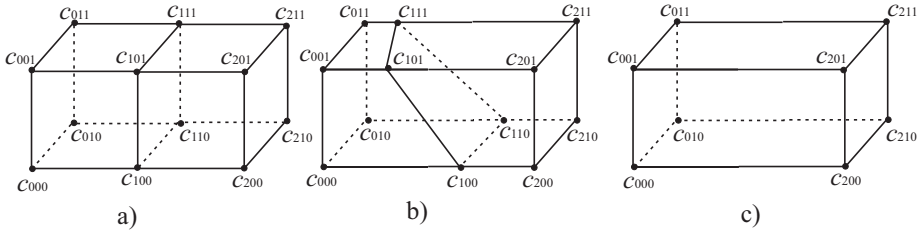


Figure 3.42: a) scalar field $s_1(x,y,z)$; b) $s_1(\mathbf{r}^{-1}(x,y,z))$; c) $s_2(x,y,z)$.

3.42b. Then we have to show that this scalar field coincides with the field s_2 shown in figure 3.42c. Introducing the auxiliary functions

$$\begin{aligned} c_0(y,z) &= (1-y)(1-z)c_{000} + y(1-z)c_{010} + (1-y)zc_{001} + yzc_{011} \\ c_1(y,z) &= (1-y)(1-z)c_{100} + y(1-z)c_{110} + (1-y)zc_{101} + yzc_{111} \\ c_2(y,z) &= (1-y)(1-z)c_{200} + y(1-z)c_{210} + (1-y)zc_{201} + yzc_{211} \\ c(y,z) &= 2\frac{c_1(y,z)-c_0(y,z)}{c_2(y,z)-c_0(y,z)} \end{aligned} \tag{3.23}$$

the scalar fields s_1 and s_2 can be expressed as

$$\begin{aligned} s_1(x,y,z) &= \begin{cases} (1-x)\cdot c_0(y,z) + x\cdot c_1(y,z) & \text{for } 0 \le x \le 1 \\ (2-x)\cdot c_1(y,z) + (x-1)\cdot c_2(y,z) & \text{for } 1 \le x \le 2 \end{cases} \\ s_2(x,y,z) &= (1-\frac{x}{2})\cdot c_0(u,z) + \frac{x}{2}\cdot c_2(y,z). \end{aligned} \tag{3.24}$$

It is sufficient to show that s_1 and s_2 can be transformed into each other by a regular one-to-one reparametrization $\mathbf{r}$ of the domain D. We choose

$$\begin{aligned} \mathbf{r}(x,y,z) &= (x_{\mathbf{r}}(x,y,z), y, z) \\ x_{\mathbf{r}}(x,y,z) &= \begin{cases} x\cdot c(y,z) & \text{for } 0 \le x \le 1 \\ (2-x)\cdot c(y,z) + (x-1)\cdot 2 & \text{for } 1 \le x \le 2 \end{cases} . \end{aligned} \tag{3.25}$$

From (3.22) and (3.23) we get that $c_0(y,z) < c_1(y,z) < c_2(y,z)$ for any $y,z \in [0,1]$. This and (3.23) gives that $0 < c(y,z) < 2$ for any $y,z \in [0,1]$. This makes sure that the reparametrization $\mathbf{r}$ defined by (3.25) is regular and one-to-one. In particular we have $\mathbf{r}(D) = \mathbf{r}^{-1}(D) = D$ where

$$\begin{aligned} \mathbf{r}^{-1}(x,y,z) &= (x_{\mathbf{r}}^{-1}(x,y,z), y, z) \\ x_{\mathbf{r}}^{-1}(x,y,z) &= \begin{cases} \frac{x}{c(y,z)} & \text{for } 0 \le x \le c(y,z) \\ \frac{x-2\cdot(c(y,z)-1)}{2-c(y,z)} & \text{for } c(y,z) \le x \le 2 \end{cases} \end{aligned} \tag{3.26}$$

is the inverse function of $\mathbf{r}(x,y,z)$. Then (3.24) and (3.26) give

$$s_1(\mathbf{r}^{-1}(x,y,z)) = s_2(x,y,z)$$

which proves the theorem.

Theorem 2 states that merging is possible when the scalar values increase monotonously in x-direction. A similar statement can be made for monotonously decreasing values in x-direction, and for monotonously increasing (decreasing)

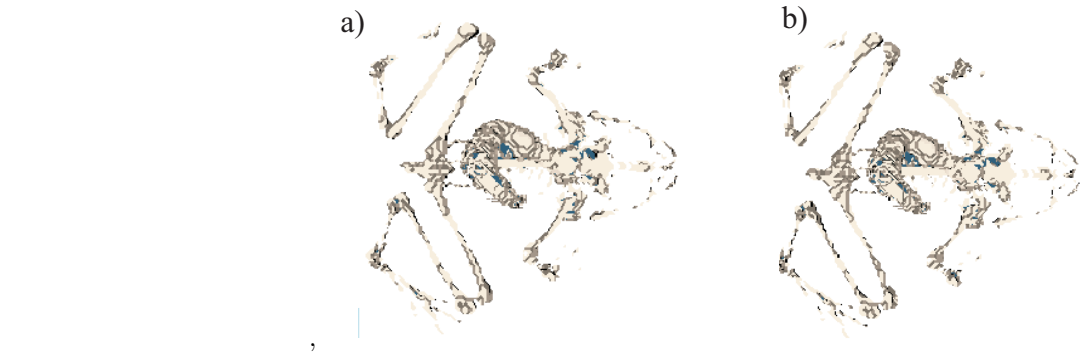


Figure 3.43: a) result of the MC algorithm; b) application of the MC algorithm after a threshold-independent simplification of the data set described in [118] gives a reduction of the number of triangles to 14% (from [118]).

values in y- or z-direction. Figure 3.41c illustrates the monotony condition which allows a merging as shown in figure 3.41b.

As a generalization of theorem 2 we get the following conditions to merge a box of cells:

Theorem 3 *Let $s_1(x,y,z)$ be the piecewise trilinear scalar field which is defined by the scalar values $c_{i,j,k}$ ($i = i_b,...,i_e,\ j = j_b,...,j_e,\ k = k_b,...,k_e$) and the regular normalized grid $\mathbf{x}_{i,j,k} = (i,j,k)^T$ in the domain $D = [i_b,i_e] \times [j_b,j_e] \times [k_b,k_e]$. Furthermore, let s_2 be the trilinear scalar field defined in D by the eight scalar values c_{i_b,j_b,k_b}, c_{i_e,j_b,k_b}, c_{i_b,j_e,k_b}, c_{i_e,j_e,k_b}, c_{i_b,j_b,k_e}, c_{i_e,j_b,k_e}, c_{i_b,j_e,k_e}, c_{i_e,j_e,k_e}. Then the contours of s_0 and s_1 have the same topology for any threshold r if the following conditions are satisfied:*

$$
\begin{aligned}
c_{i,j,k} \ \circ_x c_{i+1,j,k} \quad &\text{for} \quad i = i_b,...,i_e-1,\ j = j_b,...,j_e,\ k = k_b,...,k_e \\
c_{i,j,k} \ \circ_y c_{i,j+1,k} \quad &\text{for} \quad i = i_b,...,i_e,\ j = j_b,...,j_e-1,\ k = k_b,...,k_e \\
c_{i,j,k} \ \circ_z c_{i,j,k+1} \quad &\text{for} \quad i = i_b,...,i_e,\ j = j_b,...,j_e,\ k = k_b,...,k_e-1
\end{aligned}
$$

with $\circ_x, \circ_y, \circ_z \in \{>,<\}$.

In [118] a number of strategies for merging cells to a brick are discussed.

The result of the MC algorithm to the merged cells is a simplified contour which may contain cracks. After removing these cracks (as done in [118]), the original MC contour and the simplified one always have the same topology.

Figure 3.43 shows the result of the Marching Cubes algorithm to a brick structure built following [118]. The triangular mesh resulting from the brick structure (including crack removal) was 14% of the number of triangles produced by the original Marching Cubes algorithm. (The test data set used here is part of the VTK distribution - see [165])

In [118] there is a comparison of the performance of the brick structures to the improved Marching Cubes approaches discussed in section 3.5.2.2. It turned out that in most cases the algorithms in section 3.5.2.2 are superior concerning the mesh reduction rate, visual appearance of the mesh, and computing performance. Nevertheless [118] gives a first approach to simplifying volume data for isosurface extraction threshold independently.

Chapter 4

CAGD for Flow Visualization

Since the very beginning of Scientific Visualization, flow visualization has been one of its main topics. Flow data comes from numerical simulations (CFD - computational fluid dynamics), or from experiments and measurements.

Flow data consists of a finite number of sample points $\mathbf{x_i}$ in the 2D or 3D Euclidian space, and a number of velocity vectors $\mathbf{v_i}$ (2D or 3D) where each vector is assigned to exactly one sample point:

$$F = \{(\mathbf{x_i}, \mathbf{v_i}) \in \mathbb{E}^n \times \mathbb{R}^n : \mathbf{i} \in G \text{ and } G \text{ finite}\} \tag{4.1}$$

with $n \in \{2, 3\}$. The sample points $\mathbf{x_i}$ may lie on a certain grid, or they may be scattered in 2D (3D).

A 2D vector field on a regular grid is defined by $G = \{(0, ..., n_x) \times (0, ..., n_y)\}$. Figure 4.1a gives an illustration of this with $n_x = n_y = 4$. Figure 4.1b shows an example of scattered flow data with $G = \{1, ..., 6\}$.

Following the classification of scientific data in [23], flow data can be described as $E^{1V}_{[n]}$ with $n \in \{2, 3\}$.

The definition of flow data given in (4.1) can be generalized in some ways.

- The measurement or simulation of flow data may not only give one vector per sample point but also additional information (scalars, vectors) in each sample point. This additional information may be measures like pressure or temperature. For the investigations in this chapter we focus on the velocity vector in each point and omit the additional information.
- A flow data set may be considered as time dependent (unsteady). Here we additionally consider a finite number n_t of time steps. At each time step a (possibly different) vector $\mathbf{v}_{\mathbf{i},i_t}$ is measured or observed. Thus an unsteady flow field can be formally described as

  $$F = \{(\mathbf{x_i}, (\mathbf{v}_{\mathbf{i},0}, ..., \mathbf{v}_{\mathbf{i},n_t})) \in \mathbb{E}^n \times (\mathbb{R}^n)^{n_t+1} : \mathbf{i} \in G \text{ and } G \text{ finite}\} \tag{4.2}$$

  where $n \in \{2, 3\}$ is the dimensionality of the flow and $n_t + 1$ is the number of time steps. [1]

[1] Theoretically it is possible to also vary the location of the sample points over time. Since

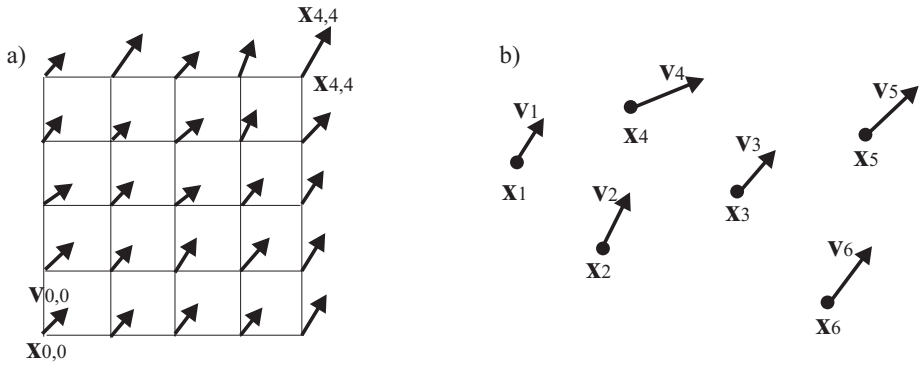


Figure 4.1: a) flow data on a regular grid; b) scattered flow data.

Figure 4.2 shows the pipeline for the process of flow visualization which we consider here. This pipeline follows [152] but emphasizes the interpolation step by considering it as a process of is own. This is justified by the fact that the interpolation of the flow data is a very significant data conversion step in the visualization process.

In figure 4.2 we recognize the usual steps of the visualization pipeline (see section 2.3.1). The filtering step works on the raw flow data. Here noise reduction, data selection or data completion may be done. Also data conversions (for instance the conversion of curvilinear grids to regular grids) may apply here. Overviews on filtering operations for flow data can be found in [152], [64] and [167].

The interpolation step of the visualization pipeline of flow data converts the flow data defined by (4.1) into a *vector field*. A vector field $\mathbf{v}$ can be defined as

$$\mathbf{v} : E \to \mathbb{R}^n \tag{4.3}$$

with $n \in \{2, 3\}$ is the dimensionality of the vector space and $E \subset \mathbb{E}^n$ is a closed, compact subset of $\mathbb{E}^n$. For converting flow data described by (4.1) into a vector field described by (4.3), E may be chosen as the convex hull of the sample points:

$$E = \text{conv}(\{\mathbf{x_i} : \mathbf{i} \in G\}). \tag{4.4}$$

The interpolation process can be formulated as searching a vector field $\mathbf{v}$ with

$$\mathbf{v}(\mathbf{x_i}) = \mathbf{v_i} \text{ for all } \mathbf{i} \in G. \tag{4.5}$$

In the mapping step of the visualization pipeline a suitable visualization technique for the vector field and a certain set of interpretation aims have to be selected, and an arrangement of the parameters of the technique has to be specified. The resulting geometric primitives are finally rendered in the rendering step of the pipeline. A detailed description of the visualization pipeline of flow data can be found in [152] and [167].

It is the purpose of this chapter to study the application of CAGD methods in flow visualization. We see these applications in three ways:

practical unsteady flow data sets are defined on a fixed grid, we do not consider this extension here.

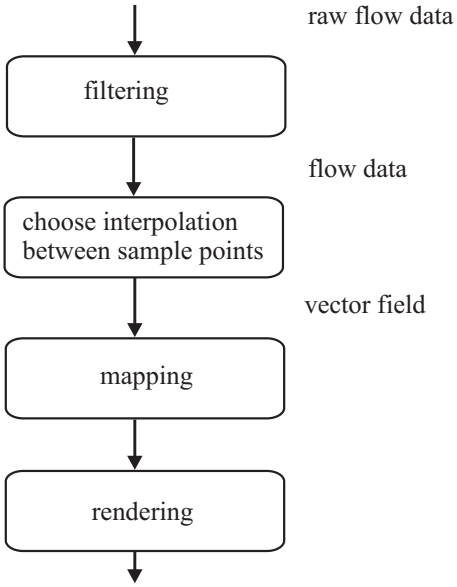


Figure 4.2: Pipeline for flow visualization.

- The interpolation of flow data is a critical part of the visualization process of flow data. Since interpolation issues are well studied in the CAGD context, the results obtained there can be used to study the interpolation problem of flow data. We do so in section 4.2.
- To visualize flow data, a number of techniques exist which are based on the application of curves and surfaces. We study these in section 4.3.
- Similar to the design of curves and surfaces in CAGD, flow data can be obtained not only by measurement and simulation but also by design. We discuss the design process of flow data and its usefulness for flow visualization in section 4.4.

Before starting to study these three applications of CAGD methods in flow visualization, we summarize important properties of vector fields in section 4.1 to build a base for the sections 4.2-4.4.

4.1 Properties of Vector Fields

Collecting properties of a vector field, we start with the simplest case: a steady 2D vector field

$$\begin{aligned} \mathbf{v}: \quad E_2 &\rightarrow \mathbb{R}^2 \\ (x,y) &\rightarrow \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix} \end{aligned} \tag{4.6}$$

where E_2 is a closed and compact subset of $\mathbb{E}^2$. Furthermore we assume $\mathbf{v}$ to be continuous and differentiable. Then the partial derivatives of $\mathbf{v}$ can be written

as

$$\mathbf{v}_x(x,y) = \begin{pmatrix} \frac{du}{dx}(x,y) \\ \frac{dv}{dx}(x,y) \end{pmatrix} = \begin{pmatrix} u_x(x,y) \\ v_x(x,y) \end{pmatrix}$$
$$\mathbf{v}_y(x,y) = \begin{pmatrix} \frac{du}{dy}(x,y) \\ \frac{dv}{dy}(x,y) \end{pmatrix} = \begin{pmatrix} u_y(x,y) \\ v_y(x,y) \end{pmatrix}. \tag{4.7}$$

Higher order partials can be computed similarly. The *Jacobian matrix* $\mathbf{J_v}$ is a 2×2 matrix which is defined in every point of the domain of the vector field by

$$\mathbf{J_v}(x,y) = \begin{pmatrix} u_x(x,y) & u_y(x,y) \\ v_x(x,y) & v_y(x,y) \end{pmatrix}. \tag{4.8}$$

The determinant of $\mathbf{J_v}$ is called *Jacobian* of $\mathbf{v}$.

A point $\mathbf{x}_0 \in E_2$ is called a *critical point* iff $\mathbf{v}(\mathbf{x}_0) = (0,0)^T = \mathbf{0}$ and $\mathbf{v}(\mathbf{x}) \neq \mathbf{0}$ for any $\mathbf{x} \neq \mathbf{x}_0$ in a certain neighborhood of $\mathbf{x}_0$.

A *tangent curve* $\mathbf{s}(t)$ of the vector field $\mathbf{v}$ is a curve in E_2 with

$$\dot{\mathbf{s}}(t) = \mathbf{v}(\mathbf{s}(t)) \tag{4.9}$$

for any t of the domain of $\mathbf{s}$. In (4.9), $\dot{\mathbf{s}}$ denotes the tangent vector of $\mathbf{s}$. Considering the vector field $\mathbf{v}$ as the velocity field of a steady flow, a tangent curve describes the path of a massless particle set out at a certain location in the flow. Thus the tangent curve in a steady vector field is also called *stream line*.

Tangent curves do not intersect each other (except for critical points of $\mathbf{v}$). Given a point in the flow, there is one and only one tangent curve through it (except for critical points of $\mathbf{v}$).

4.1.1 Classification of critical points

To classify a critical point in a 2D steady vector field, sectors of different flow behavior around it have to be considered. Three kinds of sectors can be distinguished ([57]):

- In a *parabolic sector* either all tangent curves end, or all tangent curves originate, in the critical point. Figure 4.3a shows an example.
- In a *hyperbolic sector* all tangent curves go by the critical point, except for two tangent curves making the boundaries of the sector. One of these two tangent curves ends in the critical point while the other one originates in it. Figure 4.3b shows an example.
- In an *elliptic sector* all tangent curves originate and end in the critical point. Figure 4.3c shows an example.

A critical point in a 2D vector field is completely classified by specifying number and order of all sectors around it. Consider figure 4.4a[2] for an example. This critical point consists of 7 sectors in the following order: hyperbolic, elliptic, hyperbolic, elliptic, parabolic, hyperbolic, hyperbolic.

[2]The visualization technique used for this (and the following) illustrations is called *Integrate&Draw* and is described in section 4.3.3.2. For now it is sufficient to mention that the behavior of the tangent curves can be detected quite well in this visualization.

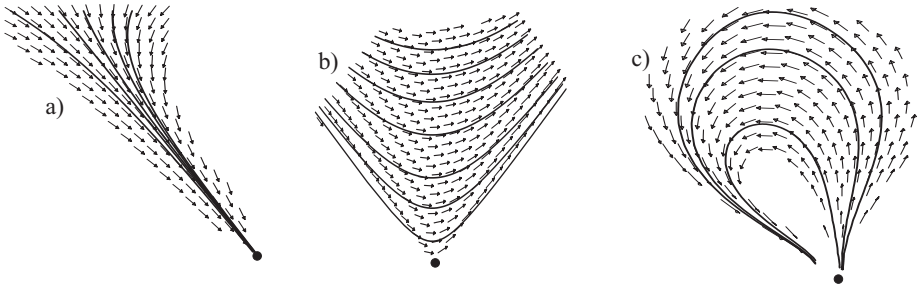


Figure 4.3: Sectors of a critical point; a) parabolic sector; b) hyperbolic sector; c) elliptic sector (from [205]).

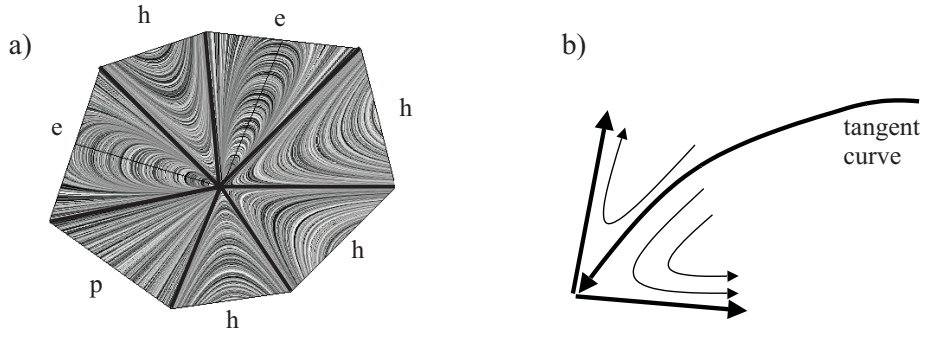


Figure 4.4: a) general critical point; b) tangent curve separating two hyperbolic sectors.

The different sectors are delimited by tangent curves originating or ending in the critical point. Figure 4.4b shows such a tangent curve delimiting two hyperbolic sectors.

Each critical point can be assigned an *index*:

$$\text{index} = 1 + \frac{n_e - n_h}{2} \tag{4.10}$$

where n_e is the number of elliptic sectors and n_h is the number of hyperbolic sectors. The index can also be interpreted as the number of counterclockwise revolutions made by the vectors of $\mathbf{v}$ while traveling counterclockwise on a closed curve around the critical point (the closed curve must be so tight to the critical point that no other critical points are inside it).

The index can be considered as an overview of the complexity of a critical point but does not cover the complete classification: there are critical points with different sectors but the same index. For instance, both critical points in figures 4.4a and 4.21 have an index of 0.

An further introduction to the classification of 2D critical points and their indices can be found in [57].

A critical point $\mathbf{x}_0$ in the vector field $\mathbf{v}$ is called a *first order critical point* iff the Jacobian does not vanish in $\mathbf{x}_0$; otherwise the critical point is called *higher order critical point*. As shown in [87] and [88], the classification of critical points

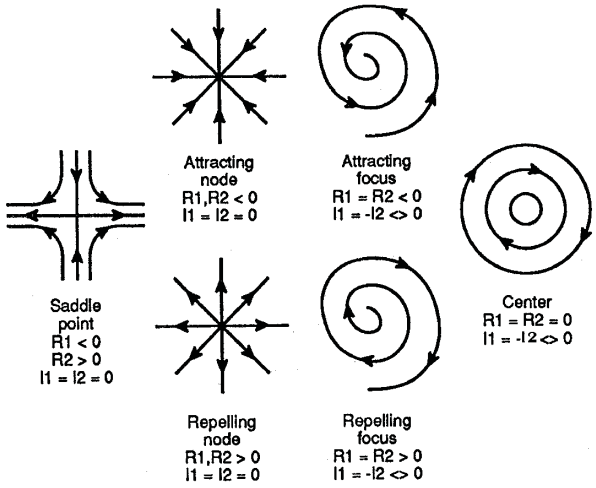


Figure 4.5: Classification of first order critical points; R_1, R_2 denote the real parts of the eigenvalues of the Jacobian matrix while I_1, I_2 denotes its imaginary parts (from [87]).

$\mathbf{x}_0 = (x_0, y_0)$ in the vector field $\mathbf{v}$ simplifies if $\mathbf{x}_0$ is a first order critical point. In this case a first order Taylor expansion

$$\mathbf{v}_{T1,\mathbf{x}_0} = \begin{pmatrix} u_x(\mathbf{x}_0) & u_y(\mathbf{x}_0) \\ v_x(\mathbf{x}_0) & v_y(\mathbf{x}_0) \end{pmatrix} \cdot \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} \tag{4.11}$$

of the flow around $\mathbf{x}_0$ is sufficient to obtain the complete classification of it. (4.11) ensures that

$$\mathbf{J}_\mathbf{v}(\mathbf{x}_0) = \mathbf{J}_{\mathbf{v}_{T1,\mathbf{x}_0}}(\mathbf{x}_0). \tag{4.12}$$

It turns out that for $\det(\mathbf{J}_\mathbf{v}(\mathbf{x}_0)) < 0$, the critical point $\mathbf{x}_0$ consists of 4 hyperbolic sectors and therefore has an index of -1. A critical point of this classification is called a *saddle point*. In this case the eigenvectors of $\mathbf{J}_\mathbf{v}(\mathbf{x}_0)$ denote the delimiters of the hyperbolic areas around $\mathbf{x}_0$. For $\det(\mathbf{J}_\mathbf{v}(\mathbf{x}_0)) > 0$, the critical point $\mathbf{x}_0$ consists of one parabolic sector and therefore has an index of +1.

This classification of a first order critical point $\mathbf{x}_0$ with an index of +1 can be refined by considering the eigenvalues of $\mathbf{J}_\mathbf{v}(\mathbf{x}_0)$. Let R_1, R_2 be the real parts of the eigenvalues of $\mathbf{J}_\mathbf{v}(\mathbf{x}_0)$, and let I_1, I_2 be the imaginary parts of the eigenvalues of $\mathbf{J}_\mathbf{v}(\mathbf{x}_0)$. Then the refined classification following [87] is shown in figure 4.5. Note that positive real parts denote a repelling behavior of the flow while negative real parts indicate an attracting behavior. Non-zero imaginary parts denote a circulating behavior of the flow.

4.1.2 Separatrices

Separatrices are tangent curves that divide the vector field into areas of different flow behavior. Different types of separatrices are possible:

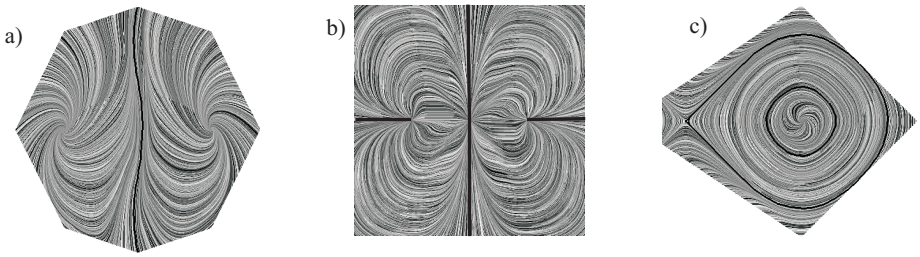


Figure 4.6: Types of separatrices; a) separatrix not touching a critical point (type 2); b) separatrix not separating different sectors in the critical points (type 3); c) inner separatrix is closed curve (type 4).

1. Each tangent curve originating/ ending in the critical point and separating two sectors there is a separatrix. Figure 4.4b illustrates a separatrix which separates two hyperbolic sectors of a critical point.

2. Separatrices may not touch any critical point. They may go "from infinity to infinity" (or from one border line of the vector field to another one). Figure 4.6a shows a vector field which consists of two attracting foci (see classification in [87] and figure 4.5). The separatrix between the two critical points does not touch any of them.

3. Separatrices may originate/ end in a critical point without separating sectors there. Figure 4.6b gives an example of this. Here we have one repelling node (middle) and two attracting nodes (left, right). Since each node consists of only one parabolic sector, the separatrices shown in the figure do not separate different sectors there.

4. Separatrices may be closed curves which do not touch any critical point. Figure 4.6c gives an example. Here we have two critical points: a saddle point and an attracting focus. The outer separatrix originates and ends in the saddle point and is therefore a separatrix of type 1. The inner separatrix of type 4 separates a region of inflow into the attracting focus and a region of circulation around the attracting focus.

Separatrices of the type 1 were already treated in [87]. In [112] it was shown that there exist more general separatrices. [195] gives an approach to extract some of them by introducing and treating critical points at infinity. However, the classification made above seems to cover all kinds of non-trivial separatrices treated in [112] and [195].

4.1.3 Topology of a 2D vector field

The topology of a 2D vector field denotes one of its most important features. It is completely described by detecting and classifying all critical points, and finding all separatrices. The topology of a vector field describes the behavior of the whole vector field in terms of only a small number of items. Thus it is a useful tool for analyzing and visualizing vector fields. Vector field visualization techniques which make use of the topology are discussed in section 4.3.2.1. The

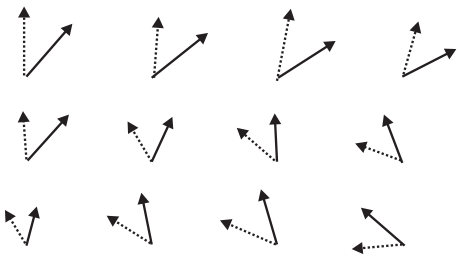


Figure 4.7: Rotated vector fields; if the solid arrows denote the vector field $\mathbf{v}$, the dashed arrows denote the vector field $\mathbf{v}^{[\frac{\pi}{4}]}$ (from [185]).

topology of vector fields can also be used to define distance functions of vector fields. This will be discussed in section 4.1.6.

In the following we call a vector field topology which consist only of first order critical points and separatrices of the type 1 a *simple topology*. Simple topologies can extracted automatically using the approaches in [87].

4.1.4 Rotated and domain rotated vector fields

This section introduces two ways of obtaining a new vector field from a given one: rotation and domain rotation. Both concepts will later be used to define distance functions on vector fields (see section 4.1.6).

The concept of *rotated vector fields* was introduced in [185]. Given a vector field $\mathbf{v}$, a new vector field $\mathbf{v}^{[\gamma]}$ can be obtained in the following way: for every point (x, y) in the domain, the direction of $\mathbf{v}(x, y)$ is rotated counterclockwise by the angle γ while the magnitude remains unchanged. Figure 4.7 gives an illustration of $\mathbf{v}^{[\frac{\pi}{4}]}$.

The rotated vector field $\mathbf{v}^{[\gamma]}$ can be computed from $\mathbf{v}$ by

$$\mathbf{v}^{[\gamma]}(x, y) = \begin{pmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{pmatrix} \cdot \mathbf{v}(x, y). \tag{4.13}$$

A special rotated vector field is the perpendicular vector field $\mathbf{v}^{\perp}$ of $\mathbf{v}$ which is defined as

$$\mathbf{v}^{\perp} = \mathbf{v}^{[\frac{\pi}{2}]} = \begin{pmatrix} -v(x, y) \\ u(x, y) \end{pmatrix}. \tag{4.14}$$

The rotation of a vector field keeps locations and indices of the critical points unchanged. All other components of vector field topology may change under rotation.

The concept of *domain rotation* of a vector field describes the rotation of the domain of the vector field - including the vectors - around a critical point. This way the tangent curves are rotated around the critical point as well.

Given the vector field $\mathbf{v}(x, y)$ with a critical point $\mathbf{x}_0 = (x_0, y_0)$, the domain rotated vector field $\mathbf{v}^{\langle \delta, \mathbf{x}_0 \rangle}$ which is obtained by a counterclockwise domain rotation around $\mathbf{x}_0$ by the angle δ can be written as

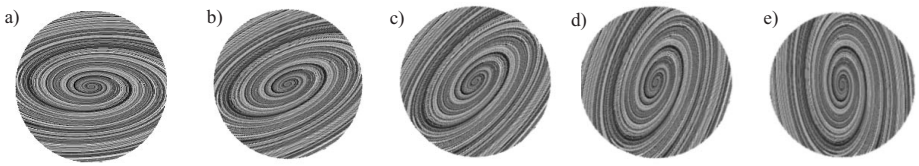


Figure 4.8: Domain rotated vector fields around a critical point $\mathbf{x}_0$; a) $\mathbf{v} = \mathbf{v}^{\langle 0,\mathbf{x}_0\rangle}$; b) $\mathbf{v}^{\langle \frac{\pi}{8},\mathbf{x}_0\rangle}$; c) $\mathbf{v}^{\langle \frac{\pi}{4},\mathbf{x}_0\rangle}$; d) $\mathbf{v}^{\langle \frac{3\pi}{8},\mathbf{x}_0\rangle}$; e) $\mathbf{v}^{\langle \frac{\pi}{2},\mathbf{x}_0\rangle}$.

$$\mathbf{v}^{\langle \delta,\mathbf{x}_0\rangle} = \begin{pmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{pmatrix} \cdot \tag{4.15}$$
$$\cdot \quad \mathbf{v}\left(((x,y)-(x_0,y_0)) \cdot \begin{pmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{pmatrix} + (x_0,y_0)\right).$$

Figure 4.8 shows example of domain rotated vector fields.

A domain rotation of a vector field $\mathbf{v}$ around the critical point $\mathbf{x}_0$ keeps only the location of $\mathbf{x}_0$ while the locations of other critical points change. Nevertheless the classification of the critical points (see section 4.1.1) remains unchanged.

4.1.5 Derived measures

Given a 2D vector field, a variety of measures can be derived from it which may be used for visualization purposes ([152], [64]):

- magnitude $\|\mathbf{v}\| = \sqrt{u^2+v^2}$
- velocity gradient $\nabla\mathbf{v} = (u_x, v_y)^T$
- divergence $\mathrm{div}(\mathbf{v}) = \nabla \cdot \mathbf{v} = u_x + v_y$

where ∇ denotes the Nabla operator $\nabla = (\frac{d}{dx}, \frac{d}{dy})^T$. An introduction to the concepts of ∇ and div can be found in [38].

Vector fields $\mathbf{v}$ with $\mathrm{div}(\mathbf{v}) \equiv 0$ are of special interest: they describe an *incompressible flow*. For such a vector field, a scalar field $s(x,y)$ can be found in such a way that tangent curves of $\mathbf{v}$ coincide with the equipotential lines of s, i.e.

$$\mathbf{v}(x,y) = \begin{pmatrix} -s_y(x,y) \\ s_x(x,y) \end{pmatrix} \tag{4.16}$$

The scalar field s is sometimes called *stream function*.

Another derived measure of a vector field is its *curvature*, which was studied in [185]. Starting from the observation that for each point in the flow there is exactly one tangent curve through it (except for critical points), we compute the curvature of the tangent curve in each domain point of the vector field.

Given a (non-critical) point (x_0,y_0) in $\mathbf{v}$, let $\mathbf{s}$ be the tangent curve through (x_0,y_0). Furthermore, let $\mathbf{s}$ be parameterized in such a way that

$$\mathbf{s}(t_0) = (x_0,y_0) \tag{4.17}$$
$$\dot{\mathbf{s}}(t_0) = \mathbf{v}(\mathbf{s}(x_0,y_0)). \tag{4.18}$$

($\dot{\mathbf{s}}(t)$ denotes the tangent vector of $\mathbf{s}(t)$). Then we can compute the second derivative vector $\ddot{\mathbf{s}}$ of $\mathbf{s}$ at t_0 by applying the chain rule to (4.18):

$$\ddot{\mathbf{s}}(t_0) = (u \cdot \mathbf{v}_x + v \cdot \mathbf{v}_y)(x_0, y_0). \tag{4.19}$$

Now we can easily compute the signed curvature of $\mathbf{s}$ in (x_0, y_0):

$$\kappa(t_0) = \frac{\det[\dot{\mathbf{s}}(t_0), \ddot{\mathbf{s}}(t_0)]}{\|\dot{\mathbf{s}}(t_0)\|^3}. \tag{4.20}$$

(4.18), (4.19) and (4.20) have the following consequence: in order to compute the curvature of a tangent curve in a certain point of a vector field it is not necessary to know the tangent curve itself. It is sufficient to know the vector field $\mathbf{v}$ and its first order partials.

Inserting (4.18) and (4.19) into (4.20), we obtain a simple formula for the curvature of the tangent curve in every point of the vector field:

$$\kappa(\mathbf{v}) = \frac{u \cdot \det[\mathbf{v}, \mathbf{v}_x] + v \cdot \det[\mathbf{v}, \mathbf{v}_y]}{\|\mathbf{v}\|^3}. \tag{4.21}$$

(4.21) describes a scalar field in the domain of the vector field $\mathbf{v}$. This scalar field describes the curvature of the tangent curve in every point of the domain. In [185], this scalar field $\kappa(\mathbf{v})$ is called the *curvature of the vector field* $\mathbf{v}$. $\kappa(\mathbf{v})$ is only defined for non-critical points. It does not depend on the magnitudes of the vectors in $\mathbf{v}$. For the perpendicular vector field $\mathbf{v}^\perp$ we can compute its curvature by inserting (4.14) into (4.21). This way we obtain

$$\kappa(\mathbf{v}^\perp) = \frac{u \cdot \det[\mathbf{v}, \mathbf{v}_y] - v \cdot \det[\mathbf{v}, \mathbf{v}_x]}{\|\mathbf{v}\|^3}. \tag{4.22}$$

In section 4.3.3.3 the application of the curvature of vector fields for visualization is discussed.

4.1.6 Metrics on 2D vector fields

In this section we study distance functions on vector fields. This issue recently became evident for the assessment of compression algorithms for vector fields. To evaluate a compression algorithm, the distance between the original and the compressed vector field has to be considered.

One approach for a distance function on vector fields is to consider the local deviation of direction and magnitude of the flow vectors in a certain number of sample points. The vector field compression algorithms in [86] and [183] are based on this approach. These distance functions give a fast comparison of vector fields but do not take their topologies into consideration. In fact, two vector fields with a significant different topology (and therefore different flow behavior) may have a short distance to each other.

A first approach to find a distance function which is based on the topology of vector fields is introduced in [122]. Here the critical points of the vector fields to be compared are detected and matched: for each critical point in the first vector field a corresponding critical point in the second vector field has to be found, and vice versa. Then the distances between all corresponding critical points are compared: their summation gives the distance of the two vector fields. This way the computation of the distance of two vector fields is reduced to the computation of the distance of critical points.

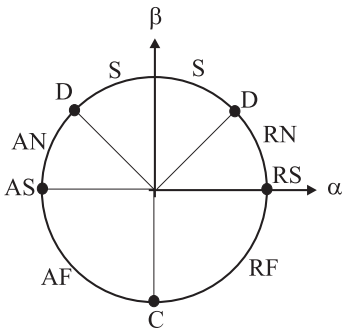


Figure 4.9: Classification of first order critical points in (α, β) phase plane (following [122]): RS (repelling star), RN (repelling node), D (degenerate - not a first order critical point), S (saddle), AN (attracting node), AS (attracting star), AF (attracting focus), C (center), RF (repelling star).

4.1.6.1 The (α, β) phase plane

The conceptional idea of how to compute the distance of two critical points in [122] is to compute the amount of work which must be performed to transform one critical point into the other. In [122] only first order critical points $\mathbf{x}_0$ in a vector field $\mathbf{v}$ are considered. Based on the Jacobian matrix $\mathbf{J}_\mathbf{v}(\mathbf{x}_0)$, the critical point $\mathbf{x}_0$ is mapped into an (α, β) phase plane by

$$\begin{aligned} p &= \operatorname{div}(\mathbf{v})(\mathbf{x}_0) = (u_x + v_y)(\mathbf{x}_0) \\ q &= \det(\mathbf{J}_\mathbf{v}(\mathbf{x}_0)) \\ \hat{\alpha} &= p \\ \hat{\beta} &= \operatorname{sign}(p^2 - 4q) \cdot \sqrt{\|(p^2 - 4q\|} \\ \alpha &= \frac{\hat{\alpha}}{\sqrt{\hat{\alpha}^2 + \hat{\beta}^2}} \\ \beta &= \frac{\hat{\beta}}{\sqrt{\hat{\alpha}^2 + \hat{\beta}^2}}. \end{aligned} \tag{4.23}$$

This way the first order critical point $\mathbf{x}_0$ is mapped onto the unit circle in the (α, β) plane.

Figure 4.9 shows the relation between the classification of first order critical points in [87] (shown in figure 4.5) and the location in the (α, β) phase plane. Note that the additionally introduced classes of critical points, attracting star and repelling star, correspond to the conditions

$$\begin{aligned} \text{attracting star:} \quad & R_1 = R_2 < 0 \ , \ I_1 = I_2 = 0 \\ \text{repelling star :} \quad & R_1 = R_2 > 0 \ , \ I_1 = I_2 = 0 \end{aligned}$$

in [87]. Then the distance of two first order critical points is simply the Euclidian distance of their corresponding points in the (α, β) plane. This distance is called EMD (earth mover's distance) in [122].

The (α, β) phase plane in [122] has a number of useful properties which correspond to the intuition of the distance of critical points:

- Invariance under scaling of the vector field. The critical point $\mathbf{x}_0$ of the vector field $\lambda \cdot \mathbf{v}$ with $\lambda > 0$ has the same (α, β) coordinates as $\mathbf{x}_0$ in $\mathbf{v}$.
- Invariance under domain rotation of the vector field around the critical point. The (α, β) coordinates of the critical point $\mathbf{x}_0$ in the domain rotated vector field $\mathbf{v}^{\langle \delta, \mathbf{x}_0 \rangle}$ does not depend on the angle δ.

However, the (α, β) phase plane of [122] also has properties which do not correspond to intuition:

- Inconsistent treatment of inverted vector fields. Given a first order critical point $\mathbf{x}_0$ in a vector field $\mathbf{v}$, a certain amount of work is necessary to convert this critical point into the critical point of the vector field $-\mathbf{v}$. Figure 4.10 shows an example of inverting a center and a repelling star. The inverse of the center is a center as well and has therefore the same (α, β) coordinates of $(0, -1)$. The inversion of the repelling star (coordinates $(1, 0)$ in (α, β) space) is an attracting star with the (α, β) coordinates of $(-1, 0)$.
- Collapsing of critical points with different flow behavior (but similar topology concerning [87]) into the same location in (α, β) space. To illustrate this, figure 4.11 shows the critical point $(0, 0)$ of the linear vector field

$$\mathbf{v}(x, y) = \begin{pmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{pmatrix} \cdot \begin{pmatrix} x \\ -\frac{1-2\sqrt{r(1-r)}}{1-2r}\, y \end{pmatrix}$$

  for 18 different choices of γ and r. In particular, γ and r have been chosen as

  a) : $\gamma = \frac{\pi}{2}, r = 1$; b) : $\gamma = -\frac{\pi}{2}, r = 1$;
  c) : $\gamma = \frac{\pi}{2}, r = 0.8$; d) : $\gamma = -\frac{\pi}{2}, r = 0.8$;
  e) : $\gamma = \frac{\pi}{2}, r = 0.6$; f) : $\gamma = -\frac{\pi}{2}, r = 0.6$;

  g) : $\gamma = 0, r = \frac{1}{1+\sin^2\gamma}$; h) : $\gamma = -\frac{\pi}{8}, r = \frac{1}{1+\sin^2\gamma}$;
  i) : $\gamma = \frac{\pi}{8}, r = \frac{1}{1+\sin^2\gamma}$; j) : $\gamma = -\frac{\pi}{4}, r = \frac{1}{1+\sin^2\gamma}$;
  k) : $\gamma = \frac{\pi}{4}, r = \frac{1}{1+\sin^2\gamma}$; l) : $\gamma = -\frac{3\pi}{8}, r = \frac{1}{1+\sin^2\gamma}$;

  m) : $\gamma = \frac{\pi}{4}, r = \frac{1}{2\sin^2\gamma}$; n) : $\gamma = -\frac{\pi}{4}, r = \frac{1}{2\sin^2\gamma}$;
  o) : $\gamma = \frac{\pi}{3}, r = \frac{1}{2\sin^2\gamma}$; p) : $\gamma = -\frac{\pi}{3}, r = \frac{1}{2\sin^2\gamma}$;
  q) : $\gamma = \frac{5\pi}{12}, r = \frac{1}{2\sin^2\gamma}$; r) : $\gamma = -\frac{5\pi}{12}, r = \frac{1}{2\sin^2\gamma}$.

  This way the critical points in figures 4.11a-f have (α, β) coordinates of $(0, -1)$; the critical points in figures 4.11g-l have (α, β) coordinates of $(1, 0)$; and the critical points in figures 4.11m-r have (α, β) coordinates of $(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$. This contradicts the observation that for instance the figures 4.11j and 4.11p are visually more similar than the figures 4.11m and 4.11p.

In [13] the approach of [122] is extended by considering not only the critical points but also their connectivity. This way the distance of two vector fields is not the sum of the distances of the critical points but the distance of two graphs, in which the nodes describe the critical points, and the edges describe the connectivity.

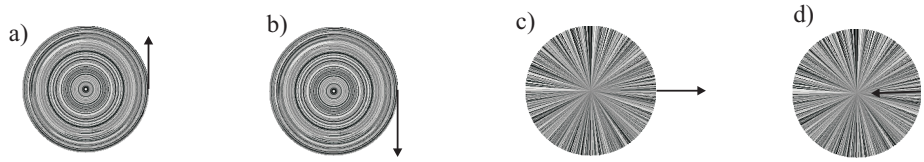


Figure 4.10: Inverted vector fields in (α, β) plane; a) center with (α, β) coordinates $(0, -1)$; b) inverse vector field of a) has the same (α, β) coordinates; c) repelling star with (α, β) coordinates $(1, 0)$; d) inverse vector field of c) has the (α, β) coordinates $(-1, 0)$.

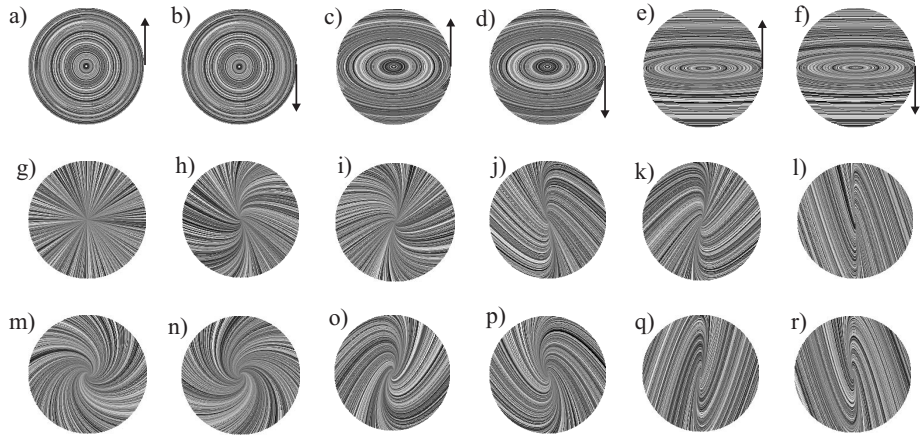


Figure 4.11: Different critical point with the same (α, β) coordinates; a-f: (α, β) coordinates $(0, -1)$; g-l: (α, β) coordinates $(1, 0)$; m-r: (α, β) coordinates $(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$.

4.1.6.2 The (γ, r) phase plane

In this section we introduce a new metric for first order critical points which does not have the disadvantages of the (α, β) phase plane described above. We follow the conceptional idea of [122] that the distance of two critical points is the amount of work which must be performed to transform one critical point into the other. We base our approach on two concepts: the rotation of a vector field (see section 4.1.4) and the scaling of the vector field into one direction. Both operations perform a parameterized transformation between different critical points and can therefore be interpreted as the amount of work to transform one critical point into another.

The idea is to distinguish only between first order critical points which cannot be transformed into each other by scaling and domain rotation. To do so, we introduce the following definitions.

Given are two vector fields $\mathbf{v}$ and $\mathbf{w}$ which both have a first order critical point in $\mathbf{x}_0$. The vector fields $\mathbf{v}$ and $\mathbf{w}$ are *domain rotation equivalent* in $\mathbf{x}_0$ (written $(\mathbf{v}, \mathbf{x}_0) \sim_{dre} (\mathbf{w}, \mathbf{x}_0)$) iff their first order Taylor expansions around $\mathbf{x}_0$ (see (4.11)) can be transformed into each other by scaling and domain rotation, i.e.

$$(\mathbf{v},\mathbf{x}_0) \sim_{dre} (\mathbf{w},\mathbf{x}_0) \iff \exists\, \delta \in [0,2\pi] \quad \exists \lambda > 0 \;:\; \mathbf{v}_{T1,\mathbf{x}_0} = \lambda\, (\mathbf{w}_{T1,\mathbf{x}_0})^{\langle \delta, \mathbf{x}_0 \rangle}. \tag{4.24}$$

Furthermore we introduce the concept of *normalized Jacobian*: given a vector field $\mathbf{v}$ with a first order critical point $\mathbf{x}_0$, we define the normalized Jacobian d_{norm} in $\mathbf{x}_0$ as

$$d_{norm}(\mathbf{v}(\mathbf{x}_0)) \;=\; 2\frac{\det(\mathbf{J}_\mathbf{v})}{u_x^2+v_x^2+u_y^2+v_y^2}(\mathbf{x}_0) \;=\; 2\frac{u_x\,v_y - v_x\,u_y}{u_x^2+v_x^2+u_y^2+v_y^2}(\mathbf{x}_0). \tag{4.25}$$

The value d_{norm} can be interpreted as a scaling independent version of the Jacobian. For any first order critical point, d_{norm} ranges between -1 and 1. Furthermore, d_{norm} is invariant under scaling and domain rotation around $\mathbf{x}_0$ of the vector field. Thus d_{norm} is constant for domain rotation equivalent vector fields: for two vector fields $\mathbf{v}$ and $\mathbf{w}$ with a first order critical point in $\mathbf{x}_0$ we get

$$(\mathbf{v},\mathbf{x}_0) \sim_{dre} (\mathbf{w},\mathbf{x}_0) \implies d_{norm}(\mathbf{v}(\mathbf{x}_0)) = d_{norm}(\mathbf{w}(\mathbf{x}_0)). \tag{4.26}$$

In a similar way to d_{norm} we introduce the *normalized divergence* div_{norm} of the vector field $\mathbf{v}$ in the critical point $\mathbf{x}_0$ as

$$\begin{aligned} \mathrm{div}_{norm}(\mathbf{v}(\mathbf{x}_0)) &= \frac{\mathrm{div}(\mathbf{v})}{\sqrt{2\,(u_x^2+v_x^2+u_y^2+v_y^2)}}(\mathbf{x}_0) \\ &= \frac{u_x+v_y}{\sqrt{2\,(u_x^2+v_x^2+u_y^2+v_y^2)}}(\mathbf{x}_0). \end{aligned} \tag{4.27}$$

The value div_{norm} can be interpreted as a scaling independent version of the divergence. For any first order critical point, div_{norm} ranges between -1 and 1. Furthermore, div_{norm} is invariant under scaling and domain rotation around $\mathbf{x}_0$ of the vector field. Thus div_{norm} is constant for domain rotation equivalent vector fields: for two vector fields $\mathbf{v}$ and $\mathbf{w}$ with a first order critical point in $\mathbf{x}_0$ we get

$$(\mathbf{v},\mathbf{x}_0) \sim_{dre} (\mathbf{w},\mathbf{x}_0) \implies \mathrm{div}_{norm}(\mathbf{v}(\mathbf{x}_0)) = \mathrm{div}_{norm}(\mathbf{w}(\mathbf{x}_0)). \tag{4.28}$$

The phase plane we use here to classify first order critical points is the area inside the unit circle where (γ, r) are the polar coordinates ($\gamma \in [0,2\pi], r \in [0,1]$). To characterize this (γ, r) phase plane, we define a *reference critical point* for each point of it. This is the critical point $(0,0)$ of the following vector field:

$$\mathbf{v}_{\gamma,r}(x,y) = \begin{pmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{pmatrix} \cdot \begin{pmatrix} x \\ -\frac{1-2\sqrt{r(1-r)}}{1-2r}\, y \end{pmatrix}. \tag{4.29}$$

$\mathbf{v}_{\gamma,r}$ defines a vector field with a first order critical point in $(0,0)$ for each point (γ, r) of the phase plane ($\gamma \in [0,2\pi], r \in [0,1]$). Figure 4.12 gives an illustration of the reference critical points in the (γ, r) phase plane.

The system of reference critical points in the (γ, r) phase plane has the following properties:

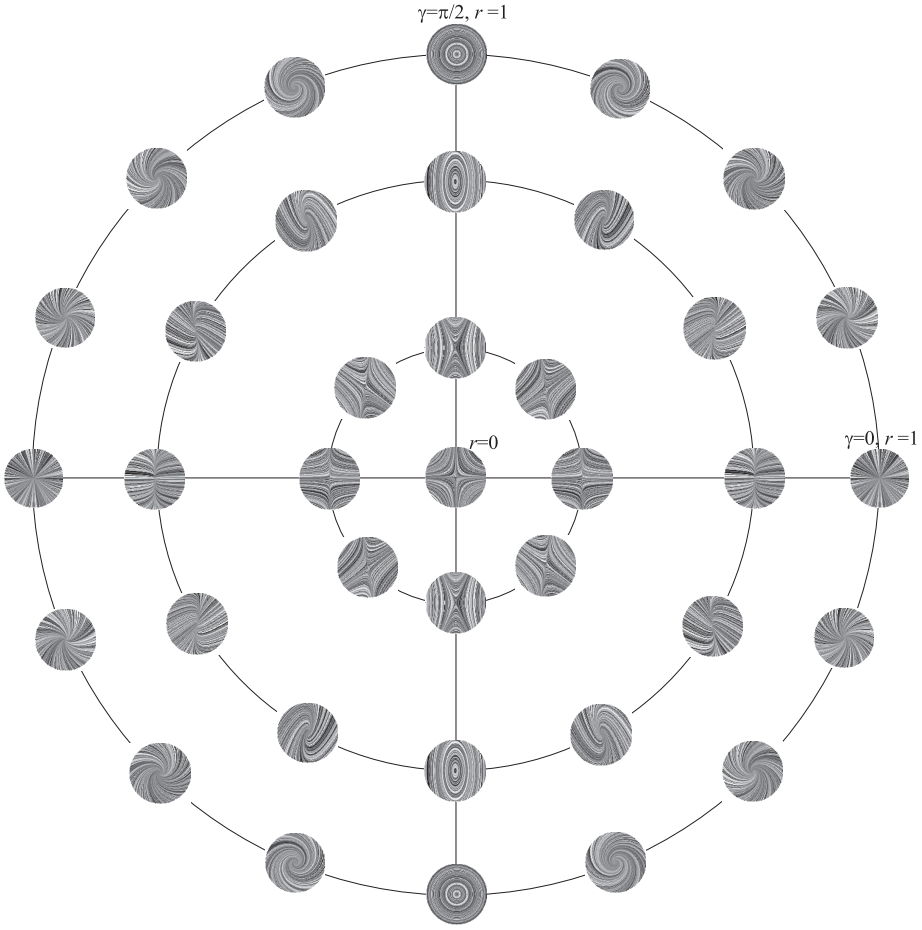


Figure 4.12: Reference critical points in the (γ, r) phase plane.

- Critical points which lie on a circle $r = \text{const}$ in the (γ, r) plane can be transformed into each other by rotation:

$$\mathbf{v}_{\gamma+\alpha,r} = \mathbf{v}_{\gamma,r}{}^{[\alpha]}. \tag{4.30}$$

  This follows directly from (4.29) and the definition (4.13) of rotated vector fields.

- Critical points which lie on a ray through the origin $r = 0$ in the (γ, r) plane can be transformed into each other by scaling of the y-component:

$$\mathbf{v}_{\gamma,r_2}(x, y) = \mathbf{v}_{\gamma,r_1}(x, \lambda\, y) \tag{4.31}$$

  with

$$\lambda = \frac{(1 - 2\, r_2)\left(1 - 2\sqrt{r_1\,(1 - r_1)}\right)}{(1 - 2\, r_1)\left(1 - 2\sqrt{r_2\,(1 - r_2)}\right)}. \tag{4.32}$$

  This is a straightforward deduction from (4.29).

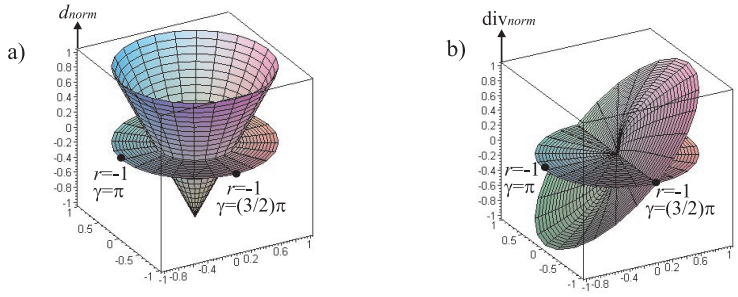


Figure 4.13: a) normalized Jacobian d_{norm} of the reference critical points as height field over the (γ, r) plane; b) normalized divergence div_{norm} of the reference critical point as height field over the (γ, r) plane.

- The critical point in the center $r = 0$ of the (γ, r) plane deserves special attention. For this point, a rotation of the corresponding vector field gives only a domain rotated version of itself:

$$\mathbf{v}_{\gamma,0} = \mathbf{v}_{0,0}{}^{\langle \frac{\gamma}{2},(0,0)\rangle}. \tag{4.33}$$

  This follows as a straightforward exercise in algebra from (4.29) and (4.15). This property can also be written as $\mathbf{v}_{\gamma_1,0} \sim_{dre} \mathbf{v}_{\gamma_2,0}$ for any $\gamma_1, \gamma_2 \in [0, 2\pi]$.

- Considering the normalized Jacobian d_{norm} of the reference critical points in the (γ, r) plane , we obtain

$$d_{norm}(\mathbf{v}_{\gamma,r}(0,0)) = 2\,r - 1. \tag{4.34}$$

  This follows from (4.25) and (4.29). Figure 4.13a illustrates the normalized Jacobian of the reference critical point as a height field over the (γ, r) plane.

- Considering the normalized divergence div_{norm} of the reference critical points in the (γ, r) plane, we obtain

$$\mathrm{div}_{norm}(\mathbf{v}_{\gamma,r}(0,0)) = \sqrt{r} \cdot \cos\gamma. \tag{4.35}$$

  This follows from (4.27) and (4.29). Figure 4.13b illustrates the normalized divergence of the reference critical point as a height field over the (γ, r) plane.

- The normalized divergence of the perpendicular vector field of $\mathbf{v}_{\gamma,r}$ is

$$\mathrm{div}_{norm}(\mathbf{v}_{\gamma,r}{}^{\perp}(0,0)) = \mathrm{div}_{norm}(\mathbf{v}_{\gamma+\frac{\pi}{2},r}(0,0)) = -\sqrt{r} \cdot \sin\gamma. \tag{4.36}$$

  This follows from (4.27), (4.29) and (4.14).

- The reference critical points in the (γ, r) plane yield the following classification of first order critical points following [87] and [122]: the critical point $(0, 0)$ of the reference vector field $\mathbf{v}_{\gamma,r}$ is a
  - saddle point (Sa) iff ($\gamma = \frac{\pi}{2}$ and $r < \frac{1}{2}$) or ($\gamma = -\frac{\pi}{2}$ and $r < \frac{1}{2}$) or $r = 0$,
  - repelling saddle (RSa) iff $-\frac{\pi}{2} < \gamma < \frac{\pi}{2}$ and $0 < r < \frac{1}{2}$,
  - attracting saddle (ASa) iff $\frac{\pi}{2} < \gamma < \frac{3}{2}\pi$ and $0 < r < \frac{1}{2}$,
  - degenerate (D) - not a critical point - iff $r = \frac{1}{2}$,
  - center 1 (C1) iff $\gamma = \frac{\pi}{2}$ and $\frac{1}{2} < r \leq 1$,
  - center 2 (C2) iff $\gamma = -\frac{\pi}{2}$ and $\frac{1}{2} < r \leq 1$,
  - repelling focus 1 (RF1) iff $0 < \gamma < \frac{\pi}{2}$ and $\frac{1}{2} < r < \frac{1}{1+\sin^2\gamma}$,
  - repelling focus 2 (RF2) iff $-\frac{\pi}{2} < \gamma < 0$ and $\frac{1}{2} < r < \frac{1}{1+\sin^2\gamma}$,
  - attracting focus 1 (AF1) iff $\frac{\pi}{2} < \gamma < \pi$ and $\frac{1}{2} < r < \frac{1}{1+\sin^2\gamma}$,
  - attracting focus 2 (AF2) iff $\pi < \gamma < \frac{3}{2}\pi$ and $\frac{1}{2} < r < \frac{1}{1+\sin^2\gamma}$,
  - repelling star 1 (RS1) iff $0 < \gamma < \frac{\pi}{2}$ and $r = \frac{1}{1+\sin^2\gamma}$,
  - repelling star 2 (RS2) iff $-\frac{\pi}{2} < \gamma < 0$ and $r = \frac{1}{1+\sin^2\gamma}$,
  - attracting star 1 (AS1) iff $\frac{\pi}{2} < \gamma < \pi$ and $r = \frac{1}{1+\sin^2\gamma}$,
  - attracting star 2 (AS2) iff $\pi < \gamma < \frac{3}{2}\pi$ and $r = \frac{1}{1+\sin^2\gamma}$,
  - repelling node (RN) iff $-\frac{\pi}{2} < \gamma < \frac{\pi}{2}$ and $\frac{1}{1+\sin^2\gamma} < r \leq 1$,
  - attracting node (AN) iff $\frac{\pi}{2} < \gamma < \frac{3}{2}\pi$ and $\frac{1}{1+\sin^2\gamma} < r \leq 1$.

  This classification of critical points has extensions to the classifications of [87] and [122] in the following way:
  - We distinguish between three kinds of saddle points. A saddle point (in the sense of [87] and [122]) is a first order critical point which has both inflow and outflow. A repelling saddle (RSa) has more outflow than inflow, i.e. a positive divergence. An attracting saddle (ASa) has more inflow than outflow and therefore a negative divergence. A saddle point (Sa) has a zero divergence.
  - The classes of points RF, RS, C, AF, AF are each subdivided into two subclasses 1 and 2. Subclass 1 means that in a neighborhood of the critical point all tangent curves turn to the left, i.e. they have non-negative curvature (see section 4.1.5). In critical points of subclass 2, all tangent curves in a neighborhood turn to the right, i.e. have non-positive curvature.

  Figure 4.14 illustrates the location of the different classes of critical points in the (γ, r) phase plane. Note that the curve $r = \frac{1}{1+\sin^2\gamma}$ defining attracting and repelling stars is not an ellipse.

After showing that the system of reference critical points in the (γ, r) phase plane has a number of useful properties, we still have to show that it describes all first order critical points uniquely (except for domain rotation and scaling). To do so, we formulate

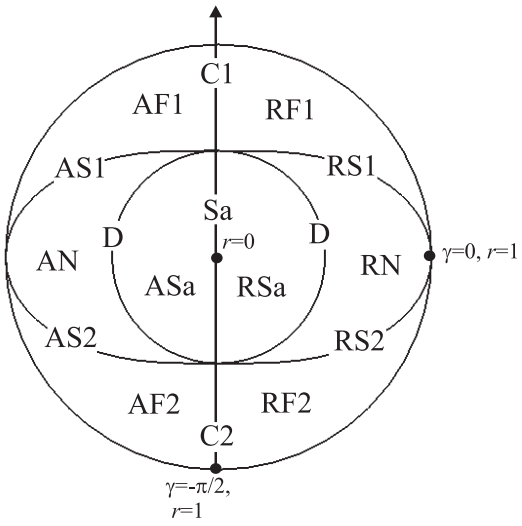


Figure 4.14: Classification of critical points in the (γ, r) phase plane.

Theorem 4 *Given is a vector field* $\mathbf{v}(x, y)$ *with a first order critical point at* $\mathbf{x}_0 = (x_0, y_0)$. *Then there exists one and only one reference critical point in the* (γ, r) *phase plane which is domain rotation equivalent to* $\mathbf{v}$. *This reference critical point is the critical point of* $\mathbf{v}_{\gamma,r}$ *with*

$$\cos\gamma = \frac{u_x + v_y}{\sqrt{(u_x + v_y)^2 + (v_x - u_y)^2}}(\mathbf{x}_0) \tag{4.37}$$

$$\sin\gamma = \frac{v_x - u_y}{\sqrt{(u_x + v_y)^2 + (v_x - u_y)^2}}(\mathbf{x}_0) \tag{4.38}$$

$$r = \frac{1}{2} + \frac{u_x\, v_y - v_x\, u_y}{u_x^2 + v_x^2 + u_y^2 + v_y^2}(\mathbf{x}_0). \tag{4.39}$$

(4.37) and (4.38) determine γ uniquely except for the case $(u_x + v_y)(\mathbf{x}_0) = (v_x - u_y)(\mathbf{x}_0) = 0$. Since in this case we obtain $r = 0$ from (4.39), γ is of no importance there.

To prove theorem 4 we assume that $\mathbf{x}_0 = (0, 0)$. This is not a restriction because $\mathbf{v}(x, y)$ can be transformed to $\mathbf{v}(x - x_0, y - y_0)$ which moves the critical point to $(0, 0)$.

Since $\mathbf{v}_{\gamma,r}$ has to fulfill $\mathbf{v}(0, 0) \sim_{dre} \mathbf{v}_{\gamma,r}(0, 0)$, we obtain (4.39) from (4.25), (4.26) and (4.34). Similarly, (4.37) is obtained from (4.27), (4.28) and (4.35). (4.38) follows from (4.27), (4.28), (4.36) and (4.14). Thus the only reference critical point which is a candidate for being domain rotation equivalent to $\mathbf{v}$ is $\mathbf{v}_{\gamma,r}$ with γ, r described by (4.37)-(4.39). To show that this reference critical point is indeed domain rotation equivalent to $\mathbf{v}$, we have to find a domain rotation angle δ and a scaling factor $\lambda > 0$ in such a way that

$$\lambda\,(\mathbf{v}_{T1,\mathbf{x}_0})^{\langle\delta,\mathbf{x}_0\rangle} = \mathbf{v}_{\gamma,r}. \tag{4.40}$$

Choosing δ and λ as

$$\cos(2\delta) = \frac{u_x^2 + v_x^2 - u_y^2 - v_y^2}{\sqrt{(u_x^2 + v_x^2 - u_y^2 - v_y^2)^2 + 4(u_x u_y + v_x v_y)^2}}(\mathbf{x}_0) \quad (4.41)$$

$$\sin(2\delta) = \frac{-2(u_x u_y + v_x v_y)}{\sqrt{(u_x^2 + v_x^2 - u_y^2 - v_y^2)^2 + 4(u_x u_y + v_x v_y)^2}}(\mathbf{x}_0) \quad (4.42)$$

$$\lambda = \frac{\mathrm{div}(\mathbf{v}_{\gamma,r}(\mathbf{x}_0))}{\mathrm{div}(\mathbf{v}(\mathbf{x}_0))}, \quad (4.43)$$

(4.40) follows from (4.15), (4.37), (4.38), (4.39), (4.41), (4.42) and (4.43) [3]. If $\mathrm{div}(\mathbf{v}(\mathbf{x}_0)) = 0$, (4.43) has to be replaced by $\lambda = \frac{\mathrm{div}(\mathbf{v}_{\gamma,r}{}^{\perp}(\mathbf{x}_0))}{\mathrm{div}(\mathbf{v}^{\perp}(\mathbf{x}_0))}$ which yields (4.40) as well. Thus theorem 4 is proven.

Theorem 4 shows that the γ, r phase plane gives a continuous one-to-one representation of all first order critical points. Thus it can be used to compute the distance of two first order critical points by mapping them into the γ, r phase plane and computing their Euclidian distance there.

4.1.7 Unsteady vector fields

Up to here, section 4.1 has only treated steady 2D vector fields. This section 4.1.7 is for studying properties of unsteady 2D vector fields.

In (4.2) unsteady flow data was introduced. In the visualization pipeline, this data has to be converted to *unsteady vector fields*.

An unsteady 2D vector field can be described as a 3D vector field

$$\mathbf{v}(x, y, t) = \begin{pmatrix} u(x, y, t) \\ v(x, y, t) \\ a(x, y, t) \end{pmatrix} \quad (4.44)$$

where

$$a(x, y, t) \equiv 1 \quad , \quad a_x = a_y = a_t \equiv 0. \quad (4.45)$$

The auxiliary dimension $a(x, y, t)$ can be interpreted as the time component of the vector field. Since time passes at a constant rate, we have $a(x, y, t) \equiv 1$. The vector field $\mathbf{v}$ has a critical point iff $u^2 + v^2 = 0$. In the rest of section 4.1.7, $\mathbf{v}$ stands only for an unsteady vector field described by (4.44).

Projecting $\mathbf{v}$ into the planes $t = \mathrm{const}$, we obtain another description of an unsteady 2D vector field:

$$\mathbf{v}_p(x, y, t) = \begin{pmatrix} u(x, y, t) \\ v(x, y, t) \end{pmatrix}. \quad (4.46)$$

For steady vector fields we have introduced the concept of tangent curves. Its extension to unsteady vector fields splits into four classes of curves: stream lines, streak lines, path lines, and time lines. In the following we define all these

[3] Ones δ and λ are specified by (4.41)–(4.43), this is just a straightforward computation for which one can use formula manipulation programs like Mathematica or Maple.

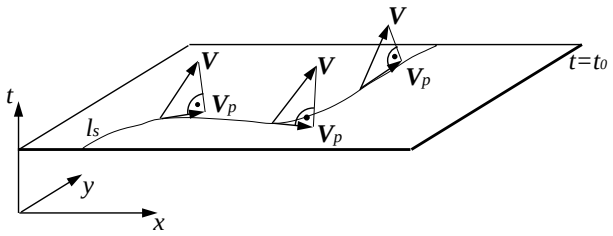


Figure 4.15: Stream line $\mathbf{l}_s$ of an unsteady flow.

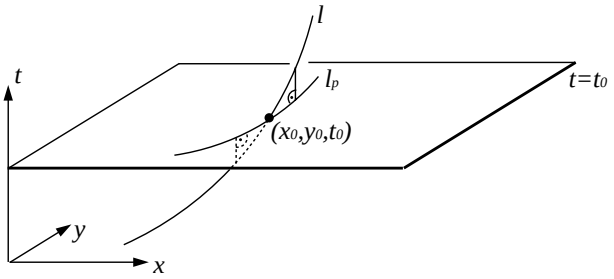


Figure 4.16: Path line $\mathbf{l}_p$ of an unsteady flow.

curves and show (following [188]) how to compute their curvatures as a local property.

Stream lines are the tangent curves of $\mathbf{v}_p$. For every time and every location there is one and only one stream line through it (except for critical points).

Figure 4.15 shows the computation of the stream lines for the time $t = t_0$. We consider the tangent curves of $\mathbf{v}_p$ at this time. We obtain the curvature of the stream lines by computing the first and second derivative vectors of the stream line for every point of the domain of $\mathbf{v}_p$:

$$\dot{\mathbf{x}}_{stream}(x,y,t) = \mathbf{v}_p(x,y,t) \tag{4.47}$$

$$\ddot{\mathbf{x}}_{stream}(x,y,t) = (u \cdot \mathbf{v}_{p_x} + v \cdot \mathbf{v}_{p_y})(x,y,t) \tag{4.48}$$

$$\kappa_{stream}(x,y,t) = \frac{\det[\dot{\mathbf{x}}_{stream}, \ddot{\mathbf{x}}_{stream}]}{\|\dot{\mathbf{x}}_{stream}\|^3}(x,y,t). \tag{4.49}$$

Path lines are obtained by setting out a particle and tracing its path in the unsteady vector field. Therefore, path lines are projections of the tangent curves of $\mathbf{v}$ into a plane $t = \text{const}$. For every location and every time there is one and only one path line through it (except for critical points).

Considering figure 4.16, the curve $\mathbf{l}$ is the tangent curve of $\mathbf{v}$ through the point (x_0, y_0, t_0). The curve $\mathbf{l}_p$ is the projection of $\mathbf{l}$ into the plane $t = t_0$. The curvature of the path line through (x_0, y_0, t_0) is the curvature of $\mathbf{l}_p$ in this point. To compute it, we express the first and second derivative vectors of $\mathbf{l}$ in (x_0, y_0, t_0):

$$\dot{\mathbf{x}}_l = \mathbf{v} \quad , \quad \ddot{\mathbf{x}}_l = u \cdot \mathbf{v}_x + v \cdot \mathbf{v}_y + a \cdot \mathbf{v}_t. \tag{4.50}$$

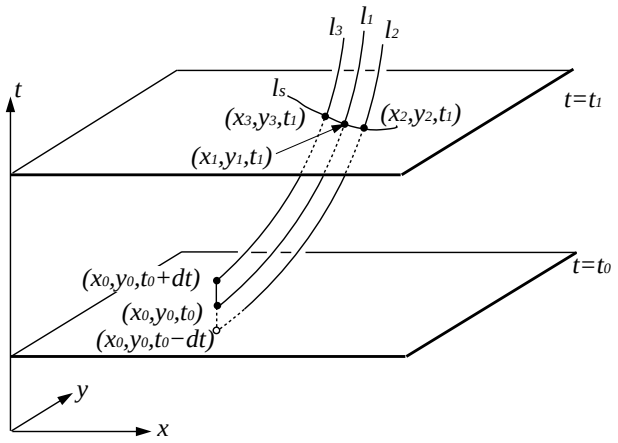


Figure 4.17: Streak line $\mathbf{l}_s$ of an unsteady flow.

Projecting $\dot{\mathbf{x}}_l$ and $\ddot{\mathbf{x}}_l$ into the plane $t = t_0$ and taking (4.45) into consideration we obtain the first and second derivatives of the path line in (x_0, y_0, t_0):

$$\dot{\mathbf{x}}_{path}(x, y, t) = \mathbf{v}_p(x, y, t) \tag{4.51}$$

$$\ddot{\mathbf{x}}_{path}(x, y, t) = (u \cdot \mathbf{v}_{p_x} + v \cdot \mathbf{v}_{p_y} + \mathbf{v}_{p_t})(x, y, t). \tag{4.52}$$

Then the curvature of the path line through (x_0, y_0, t_0) is

$$\kappa_{path}(x, y, t) = \frac{\det[\dot{\mathbf{x}}_{path}, \ddot{\mathbf{x}}_{path}]}{\|\dot{\mathbf{x}}_{path}\|^3}(x, y, t). \tag{4.53}$$

A *streak line* is the location of all particles set out at one point at different times. To illustrate this, consider figure 4.17. Suppose a particle is set out at (x_0, y_0, t_0). The path of the particle is the tangent curve $\mathbf{l}_1$ of the vector field $\mathbf{v}$. $\mathbf{l}_1$ might pass the location (x_1, y_1) at the time t_1 $(t_0 \leq t_1)$. We consider two more particles set out at (x_0, y_0) but a short time before and after t_0, i.e. we set out particles at $(x_0, y_0, t_0 - dt)$ and $(x_0, y_0, t_0 + dt)$. They follow the tangent curves $\mathbf{l}_2$ and $\mathbf{l}_3$ of $\mathbf{v}$. Suppose $\mathbf{l}_2$ passes the location (x_2, y_2) at the time $t = t_1$, and l_3 passes the location (x_3, y_3) at the time $t = t_1$. Then (x_1, y_1, t_1), (x_2, y_2, t_1) and (x_3, y_3, t_1) lie on a streak line $\mathbf{l}_s$ through (x_1, y_1, t_1). Converging dt to 0, we might compute tangent direction and curvature of $\mathbf{l}_s$ in (x_1, y_1, t_1).

Computing the curvature of streak lines gives the following two problems:

- A streak line through (x_1, y_1, t_1) is not uniquely defined. Another choice of t_0 might lead to another streak line through (x_1, y_1, t_1).
- Computing a streak line through (x_1, y_1, t_1), we have to compute the tangent curve $\mathbf{l}_1$ of $\mathbf{v}$. This is in general only possible by integrating $\mathbf{l}_1$ numerically - a procedure we want to avoid.

To deal with these problems there are two solutions:

1. We consider only the special case $t_0 = t_1$. This way a streak line through (x_1, y_1, t_1) is uniquely defined, and we do not have to trace the tangent

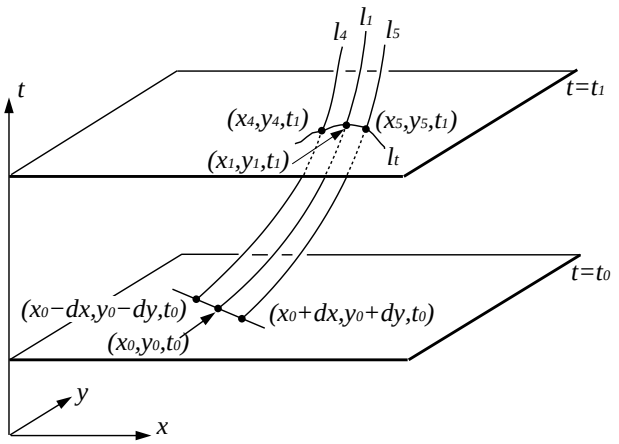


Figure 4.18: Time line $\mathbf{l}_t$ of an unsteady flow.

curves. Unfortunately, in this case the streak line through (x_1, y_1, t_1) coincides with the stream line through (x_1, y_1, t_1) computed above. So this case is of less interest.

2. Setting $t_0 = t_1$, the direction of the streak lines coincides with the direction of the stream lines: $\dot{\mathbf{x}}_{streak} = \dot{\mathbf{x}}_{stream} = \mathbf{v}_p$. Setting $t_0 = t_1 - dt$, the direction of the streak lines might be $\dot{\mathbf{x}}_{dt}$, which usually differs from $\dot{\mathbf{x}}_{streak}$. Then we want to define the "curvature" of streak lines as a measure of how much the directions of $\dot{\mathbf{x}}_{streak}$ and $\dot{\mathbf{x}}_{dt}$ differ. In other words: the "curvature" of streak lines is a measure of how "strongly" the directions of the streak lines change while varying the time t_0 (when the particles are set out) around t_1 (when the streak lines are considered). The choice of the concept "curvature" is justified in the following similarity to the usual curvature concept of curves: The curvature of a curve can be considered as a measure of how much the tangent direction changes while varying the location on the curve.

   To compute the "curvature" of streak lines, we have to compute

$$\ddot{\mathbf{x}}_{streak} = \lim_{dt \to 0} \frac{\dot{\mathbf{x}}_{streak} - \dot{\mathbf{x}}_{dt}}{dt}. \tag{4.54}$$

   From (4.54) we obtain

$$\dot{\mathbf{x}}_{streak}(x, y, t) = \mathbf{v}_p(x, y, t) \quad , \quad \ddot{\mathbf{x}}_{streak}(x, y, t) = \begin{pmatrix} u_t(x, y, t) \\ v_t(x, y, t) \end{pmatrix} \tag{4.55}$$

   and can compute the curvature of the streak lines by

$$\kappa_{streak}(x, y, t) = \frac{\det[\dot{\mathbf{x}}_{streak}, \ddot{\mathbf{x}}_{streak}]}{\|\dot{\mathbf{x}}_{streak}\|^3}(x, y, t). \tag{4.56}$$

Time lines are obtained by setting out particles located on a line at a fixed time and tracing them in the unsteady flow.

Consider figure 4.18. Suppose a particle is set out at (x_0, y_0, t_0). The path of the particle is the tangent curve $\mathbf{l}_1$ of $\mathbf{v}$. The curve $\mathbf{l}_1$ might pass the location

(x_1, y_1) at the time t_1 $(t_0 \leq t_1)$. We consider two more particles set out at the time $t = t_0$: $(x_0 - dx, y_0 - dy, t_0)$ and $(x_0 + dx, y_0 + dy, t_0)$. These points and (x_0, y_0, t_0) are located on a straight line in the plane $t = t_0$. Let these particles follow the tangent curves $\mathbf{l}_4$ and $\mathbf{l}_5$ of $\mathbf{v}$. Suppose $\mathbf{l}_4$ passes the location (x_4, y_4) at the time $t = t_1$, and $\mathbf{l}_5$ passes the location (x_5, y_5) at the time $t = t_1$. Then (x_1, y_1, t_1), (x_4, y_4, t_1) and (x_5, y_5, t_1) lie on a time line $\mathbf{l}_t$ through (x_1, y_1, t_1).

The choice of a particular time line through (x_1, y_1, t_1) depends on two parameters: the choice of t_0 and the choice of the straight line in the plane $t = t_0$. Thus a time line through (x_1, y_1, t_1) is not uniquely defined. We therefore cannot compute its curvature as a local property.

Since topology has been proven to be an important feature for the analysis and visualization of steady 2D vector fields, it seems obvious to also consider the topology of unsteady vector fields. However, the treatment of the topology of unsteady vector fields seems not to be defined and studied yet in the context of scientific visualization. The reasons for that are the following:

- To classify a critical point of a steady flow, we examined the behavior of the tangent curves around it. Due to the existence of different classes of tangent curves for unsteady vector fields, the approach of steady vector fields cannot be directly applied here.
- Critical points in unsteady vector fields may change their location, collapse or appear/disappear over time. These phenomena have to be considered in defining the topology of unsteady vector fields.
- The concept of separatrices of steady vector fields cannot be directly extended to unsteady vector fields because of the different classes of tangent curves.

However, the definition and visualization of the topology of 2D unsteady vector fields seems to be a challenging future research subject.

4.1.8 3D vector fields

For a 3D vector field, many of its properties can be obtained by a straightforward generalization of the 2D case (sections 4.1.1–4.1.7). In addition there are some properties which appear only for 3D vector fields and do not have 2D counterparts.

Given a 3D vector field

$$\begin{array}{rccl} \mathbf{v}: & E_3 & \rightarrow & \mathbb{R}^3 \\ & (x,y,z) & \rightarrow & \begin{pmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{pmatrix} \end{array} \tag{4.57}$$

where E_3 is a closed and compact subset of $\mathbb{E}^3$, the Jacobian matrix here is a 3×3 matrix

$$\mathbf{J}_{\mathbf{v}}(x,y,z) = \begin{pmatrix} u_x(x,y,z) & u_y(x,y,z) & u_z(x,y,z) \\ v_x(x,y,z) & v_y(x,y,z) & v_z(x,y,z) \\ w_x(x,y,z) & w_y(x,y,z) & w_z(x,y,z) \end{pmatrix}. \tag{4.58}$$

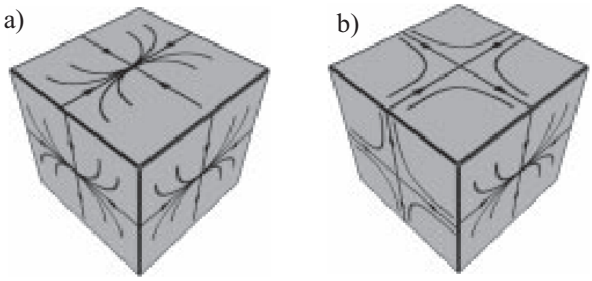


Figure 4.19: 3D first order critical point visualized by showing the classification of the 2D critical points in the eigenplanes; a) type (RN,RN,RN), b) type (Sa,Sa,AN); (from [12]).

The definition of tangent curves and critical points is similar to the 2D case. Given any (non-critical) point in the 3D vector field, there is one and only one tangent curve through it.

A general classification of 3D critical points, similar to the 2D case in section 4.1.1, seems not to exist because the sectors of different flow behavior (parabolic, elliptic, hyperbolic) which define the 2D classifications do not have counterparts in 3D. Only for the case of 3D first order critical points (i.e. critical points $\mathbf{x}_0$ with $\det(\mathbf{J}_\mathbf{v}(\mathbf{x}_0)) \neq 0$), extensions from the 2D classification of [87] exist. To show them, the eigenvectors of $\mathbf{J}_\mathbf{v}(\mathbf{x}_0)$ are computed. These vectors define three eigenplanes which intersect each other in $\mathbf{x}_0$. Then the vector field in the eigenplanes are classified as 2D vector fields. This way a first order 3D critical point $\mathbf{x}_0$ is characterized by three 2D critical points in the eigenplanes of $\mathbf{x}_0$. Figure 4.19a shows an example of a critical point of the type (RN,RN,RN): in all three eigenplanes the resulting vector field has a repelling node (RN). To visualize the three eigenplanes, a cube-like 3D icon is placed at the locations of the critical points. Figure 4.19b shows a critical point of the type (Sa, Sa, AN): two eigenplanes show a saddle point while the remaining eigenplane gives an attracting node.

A classification of 3D first order critical point which is based on the eigenplane approach can be found in [29], [148] and [12].

The concept of separatrices of 3D vector fields also seems not to have been used for visualization purposes. While separatrices for 2D vector fields are certain tangent curves, separatrices for 3D vector fields are tangent surfaces which makes their treatment far more complicated.

The concept of rotated vector fields also seems not to exist for the 3D case because to specify a rotation in 3D we do not have to determine a rotational center point but a center axis.

Most of the derived measures of 2D vector fields have direct extensions into 3D. In addition there are measures of 3D vector fields which do not have 2D counterparts:

- rotation $\mathrm{rot}(\mathbf{v}) = \nabla \times \mathbf{v} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix}$
- vorticity $\omega = \mathbf{v} \cdot \mathrm{rot}(\mathbf{v})$.

A nice explanation of the concepts of rot($\mathbf{v}$) can be found in [38].

Similarly to the curvature of 2D vector fields, the curvature of 3D vector fields can be computed as a local property of the vector field. Given a (non-critical) point (x_0, y_0, z_0) in the 3D vector field $\mathbf{v}$ defined by (4.57), let $\mathbf{s}$ be the tangent curve through (x_0, y_0, z_0). Furthermore, let $\mathbf{s}$ be parameterized in such a way that

$$\begin{aligned} \mathbf{s}(t_0) &= (x_0, y_0, z_0) && (4.59)\\ \dot{\mathbf{s}}(t_0) &= \mathbf{v}(\mathbf{s}(t_0)) && (4.60) \end{aligned}$$

where $\dot{\mathbf{s}}(t)$ denotes the tangent vector of $\mathbf{s}(t)$. Then we can compute the second derivative vector $\ddot{\mathbf{s}}$ of $\mathbf{s}$ at t_0 by applying the chain rule to (4.60):

$$\ddot{\mathbf{s}}(t_0) = (u\,\mathbf{v}_x + v\,\mathbf{v}_y + w\,\mathbf{v}_z)(x_0, y_0, z_0). \tag{4.61}$$

Applying the chain rule one more time to (4.61), we can compute the third derivative $\dddot{\mathbf{s}}$ of $\mathbf{s}$ as

$$\begin{aligned} \dddot{\mathbf{s}}(t_0) &= (\, u\,(u\,\mathbf{v}_x + v\,\mathbf{v}_y + w\,\mathbf{v}_z)_x \,+\, v\,(u\,\mathbf{v}_x + v\,\mathbf{v}_y + w\,\mathbf{v}_z)_y \\ &\quad + w\,(u\,\mathbf{v}_x + v\,\mathbf{v}_y + w\,\mathbf{v}_z)_z\,)(x_0, y_0, z_0) \\ &= (\; u\,(u_x\,\mathbf{v}_x + u\,\mathbf{v}_{xx} + v_x\,\mathbf{v}_y + v\,\mathbf{v}_{yx} + w_x\,\mathbf{v}_z + w\,\mathbf{v}_{zx}) \\ &\quad + v\,(u_y\,\mathbf{v}_x + u\,\mathbf{v}_{xy} + v_y\,\mathbf{v}_y + v\,\mathbf{v}_{yy} + w_y\,\mathbf{v}_z + w\,\mathbf{v}_{zy}) \\ &\quad + w\,(u_z\,\mathbf{v}_x + u\,\mathbf{v}_{xz} + v_z\,\mathbf{v}_y + v\,\mathbf{v}_{yz} + w_z\,\mathbf{v}_z + w\,\mathbf{v}_{zz})\;)(x_0, y_0, z_0). \end{aligned} \tag{4.62}$$

Then we can compute the curvature of $\mathbf{s}$ in (x_0, y_0, z_0) as

$$\kappa(t_0) = \frac{\|\dot{\mathbf{s}}(t_0) \times \ddot{\mathbf{s}}(t_0)\|}{\|\dot{\mathbf{s}}(t_0)\|^3}. \tag{4.63}$$

Inserting (4.60) and (4.61) into (4.63) gives the formula for the curvature $\kappa(\mathbf{v})$ at any point of the vector field $\mathbf{v}$. Note that (4.63) describes the curvature of a 3D curve and is therefore always non-negative.

For a 3D tangent curve we can not only compute its curvature as a local property; its torsion (see [55]) can also be computed as local property. The torsion at a point (x_0, y_0, z_0) in the vector field $\mathbf{v}$ can be computed by inserting (4.60), (4.61), (4.62) into

$$\tau(t_0) = \frac{\det[\dot{\mathbf{s}}(t_0), \ddot{\mathbf{s}}(t_0), \dddot{\mathbf{s}}(t_0)]}{\|\dot{\mathbf{s}}(t_0) \times \ddot{\mathbf{s}}(t_0)\|^2}. \tag{4.64}$$

This way the torsion $\tau(\mathbf{v})$ of the 3D vector field $\mathbf{v}$ is defined as a scalar field which describes the torsion of the tangent curve at any point of the flow.

For the 2D case we also introduced the curvature of the perpendicular vector field. The analogue to the perpendicular vector field in 2D is *normal surfaces* in 3D. A normal surface of a 3D vector field is a surface with the property that for any point on the surface the surface normal and the vector of the vector field have the same direction. From this definition follows that for every point in the flow there exists one and only one normal surface through it. Computing the Gaussian curvature $K(\mathbf{v})$ and the Mean curvature $H(\mathbf{v})$ of the normal surface

in every point of the vector field $\mathbf{v}$, we obtain two more derived scalar fields from a 3D vector field. For the Gaussian Curvature $K(\mathbf{v})$ we obtain (see [185], [205]):

$$K(\mathbf{v}) = \frac{k}{4 \cdot \|\mathbf{v}\|^4} \tag{4.65}$$

with

$$\begin{aligned} k &= (4\,v_y\,w_z - v_z^2 - w_y^2 - 2\,v_z\,w_y)\,u^2 \\ &\quad +(4\,u_x\,w_z - 2\,w_x\,u_z - w_x^2 - u_z^2)\,v^2 \\ &\quad +(4\,u_x\,v_y - v_x^2 - 2\,v_x\,u_y - u_y^2)\,w^2 \\ &\quad +(2\,(u_z\,v_z + u_z\,w_y + w_x\,v_z + w_x\,w_y) - 4\,(u_y\,w_z + v_x\,w_z))\,u\,v \\ &\quad +(2\,(u_y\,w_y + v_x\,w_y + u_y\,v_z + v_x\,v_z) - 4\,(v_y\,u_z + v_y\,w_x))\,u\,w \\ &\quad +(2\,(v_x\,u_z + u_y\,u_z + v_x\,w_x + u_y\,w_x) - 4\,(u_x\,v_z + u_x\,w_y))\,v\,w. \end{aligned}$$

For the Mean Curvature $H(\mathbf{v})$ we obtain (see [185], [205]):

$$H(\mathbf{v}) = \frac{h}{2 \cdot \|\mathbf{v}\|^3} \tag{4.66}$$

with

$$\begin{aligned} h &= (-v_y - w_z)\,u^2 + (-u_x - w_z)\,v^2 + (-u_x - v_y)\,w^2 \\ &\quad +(u_y + v_x)\,u\,v + (u_z + w_x)\,u\,w + (v_z + w_y)\,v\,w. \end{aligned}$$

$H(\mathbf{v})$ can also be written as

$$H(\mathbf{v}) = -\frac{\mathrm{div}(\bar{\mathbf{v}})}{2}$$

with $\bar{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$ is the normalized vector field of $\mathbf{v}$. The application of $K(\mathbf{v})$ and $H(\mathbf{v})$ for visualization purposes is discussed in section 4.3.3.4.

A first approach to defining a topology based metric on 3D vector fields can be found in [12]. There the distance of two first order critical points is obtained by computing the distances of the corresponding 2D critical points in the eigenplanes. To compute these 2D distances, the distance approach of [122] (described in section 4.1.6.1 of this work) was used. Approaches to incorporate higher order topologies or connectivity information of the critical points seem not to exist yet.

Similarly to 2D unsteady vector fields, a 3D unsteady vector field can be defined as

$$\mathbf{v}_p(x, y, z, t) = \begin{pmatrix} u(x, y, z, t) \\ v(x, y, z, t) \\ w(x, y, z, t) \end{pmatrix}. \tag{4.67}$$

Also similarly to 2D unsteady vector fields, the concepts of stream lines, streak lines, path lines and time lines can be defined. We obtain for the curvatures $\kappa_{stream}, \kappa_{path}, \kappa_{streak}$ of stream lines, path lines and streak lines:

$$
\begin{aligned}
\dot{\mathbf{x}}_{stream} = \dot{\mathbf{x}}_{path} = \dot{\mathbf{x}}_{streak} &= \mathbf{v} \\
\ddot{\mathbf{x}}_{stream} &= u\,\mathbf{v}_x + v\,\mathbf{v}_y + w\,\mathbf{v}_z \\
\ddot{\mathbf{x}}_{path} &= u\,\mathbf{v}_x + v\,\mathbf{v}_y + w\,\mathbf{v}_z + \mathbf{v}_t \\
\ddot{\mathbf{x}}_{streak} &= \mathbf{v}_t \\
\kappa_{stream}(x,y,z,t) &= \frac{\|\dot{\mathbf{x}}_{stream} \times \ddot{\mathbf{x}}_{stream}\|}{\|\dot{\mathbf{x}}_{stream}\|^3}(x,y,z,t) \\
\kappa_{path}(x,y,z,t) &= \frac{\|\dot{\mathbf{x}}_{path} \times \ddot{\mathbf{x}}_{path}\|}{\|\dot{\mathbf{x}}_{path}\|^3}(x,y,z,t) \\
\kappa_{streak}(x,y,z,t) &= \frac{\|\dot{\mathbf{x}}_{streak} \times \ddot{\mathbf{x}}_{streak}\|}{\|\dot{\mathbf{x}}_{streak}\|^3}(x,y,z,t).
\end{aligned}
$$

However, the curvature of 3D unsteady vector field seems not to have been applied for visualization purposes yet.

4.2 Interpolating Flow Data

It is the purpose of this section to introduce the most common interpolation schemes for flow data and discuss their applicability for certain visualization problems.

Interpolation is a well-studied issue in CAGD. Given a set of points in $\mathbb{E}^3$, there is a variety of approaches to construct an interpolating curve or surface through these points. To choose a particular interpolation scheme, the following demands may be considered for curves and surfaces:

- continuity of the interpolation
  The interpolant is required to have a certain algebraic continuity (C^i) or geometric continuity (G^i). (See [55] for an introduction on C^i and G^i continuity)
- minimization of certain measures (bending energy, arc length, area)
- preservation of certain geometric properties (convexity, shape)
- fairness/ aesthetic look of the interpolant.

Among the existing interpolation schemes, the class of piecewise (bi)polynomial interpolations is the most popular one. Bézier- or B-spline curves and surfaces are applied both for sample points on a grid structure and for scattered data points.

For flow data, principally the same schemes as for surfaces exist, but the demands on the interpolant may differ. Thus also the criteria for the choice of an appropriate interpolation scheme for flow data are different to the curve and surface case. The demands for flow data are

- continuity of the interpolation
  Here the concept of C^i continuity of the obtained vector field exists similar to the curve and surface case. A concept of geometric continuity for vector fields seems not to exist. Instead, a certain algebraic or geometric continuity of the tangent curves may be demanded.

- topology of the interpolant
  A certain topology of the vector field should be preserved or obtained.

(Theoretically one might also think of interpolation schemes which minimize certain properties, but this seems not to have been done yet in the context of scientific visualization.)

As we will see in the following, studying the topological behavior of certain interpolation schemes is rather complicated .In general we can make the following statement about the interpolation of flow data: as long as only a certain continuity of the interpolant matters, the interpolation schemes from the CAGD context can be directly applied for flow data interpolation. If the topology of the obtained vector field is additionally considered, new research on the applicability of the interpolation scheme has to be done, since the concept of vector field topology does not have a direct counterpart in the surface context[4].

The rest of this section 4.2 is organized as follows: sections 4.2.1 and 4.2.2 introduces the most common interpolation schemes for 2D flow data: piecewise linear or bilinear interpolation. Section 4.2.3 discusses the application of higher order (bi)polynomial interpolation schemes. In section 4.2.4, interpolation schemes for 3D flow data are discussed. Finally, section 4.2.5 treats the question of which interpolation scheme is appropriate for a given flow data set, especially under consideration of topological aspects.

4.2.1 Piecewise linear interpolation of 2D flow data

A piecewise linear interpolation of a 2D vector field is the most popular interpolation for 2D scattered flow data. The scattered sample points of the flow data are triangulated; inside each triangle a linear interpolation of the vectors at its vertices is applied.

Given the sample points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ with the assigned vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, the interpolated vector field inside the triangle $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ can be written in barycentric coordinates as

$$\mathbf{v}(x_1, x_2, x_3) = x_1\ \mathbf{x}_1 + x_2\ \mathbf{x}_2 + x_3\ \mathbf{x}_3 \tag{4.68}$$

with $x_1 + x_2 + x_3 = 1$. Figure 4.20a gives an illustration. An introduction to the concept of barycentric coordinates can be found in [55].

There is a one-to-one correlation between a piecewise linear interpolation and first order critical points. A linearly interpolated vector field has up to one non-degenerate critical point; this critical point is always a first order critical point. Conversely, each first order critical point can be constructed from a linear vector field.

Joining two piecewise linear vector fields along a common line, we get in general a globally C^0 interpolated vector field. Note that the tangent curves of a C^0 vector field are always C^1 continuous. To show that a piecewise linear interpolation does not generally yield higher order continuities for the vector field or the tangent curves, consider figure 4.20c. This figure shows the curvature plot[5] of the vector field shown in figure 4.20b. The color discontinuities across

[4] There exist concepts of surface topology but they do not directly correlate to the topology of vector fields.

[5] The curvature plot of a vector field is discussed in section 4.3.3.3. Here it is sufficient to know that for each point of the domain of the vector field the curvature of the tangent curve through it is computed and continuously color coded.

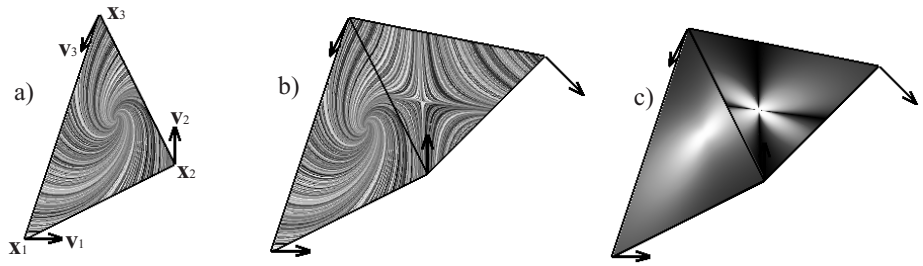


Figure 4.20: a) linear vector field inside the triangle $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$; b) piecewise linear vector field consisting of two domain triangles; c) curvature plot of b) reveals that the tangent curves in b) are not curvature continuous.

the boundaries of the domain triangles denote that the tangent curves are not globally curvature continuous, thus also not C^2 continuous. From this we can deduce that the piecewise linear vector field shown in figure 4.20b is not C^1 continuous.

4.2.2 Piecewise bilinear interpolation of 2D flow data

A piecewise bilinear interpolation is the standard for interpolating 2D flow data on rectangular grids: given the grid points $\mathbf{x}_{i,j}, \mathbf{x}_{i+1,j}, \mathbf{x}_{i+1,j+1}, \mathbf{x}_{i,j+1}$ and their assigned vectors $\mathbf{v}_{i,j}, \mathbf{v}_{i+1,j}, \mathbf{v}_{i+1,j+1}, \mathbf{v}_{i,j+1}$ on a rectangular grid, the vectors inside the rectangle $\mathbf{x}_{i,j}, \mathbf{x}_{i+1,j}, \mathbf{x}_{i+1,j+1}, \mathbf{x}_{i,j+1}$ are computed as

$$\begin{aligned} \mathbf{v}(x,y) = \quad & (1-x)(1-y)\,\mathbf{x}_{i,j} \;+\; x\,(1-y)\,\mathbf{x}_{i+1,j} \\ + \quad & (1-x)\,y\,\mathbf{x}_{i,j+1} \;+\; x\,y\,\mathbf{x}_{i+1,j+1} \end{aligned} \tag{4.69}$$

where $(x,y) \in [0,1]^2$ are the local coordinates inside the rectangle.

To compute the critical points of a bilinear vector field, we have to solve $\mathbf{v}(x,y) = (0,0)^T$ where $\mathbf{v}(x,y)$ is given by (4.69). This ends with solving a quadratic equation, the solution of which gives up to two critical points. If we obtain two distinct critical points $\mathbf{x}_{01}, \mathbf{x}_{02}$, it can be shown[6] that

$$\det(\mathbf{J}_\mathbf{v}(\mathbf{x}_{01})) = -\det(\mathbf{J}_\mathbf{v}(\mathbf{x}_{02})). \tag{4.70}$$

This gives that both critical points are of first order; one is a saddle while the other has either attracting-, center-, or repelling behavior[7] .

If the solutions $\mathbf{x}_{01}, \mathbf{x}_{02}$ of (4.69) collapse to a non-degenerate critical point, we obtain a critical point with an index of 0, i.e. this is not a first order critical point. Figure 4.21a shows the bilinear vector field $\mathbf{v}(x,y) = \begin{pmatrix} x - 2\,y + 10\,x\,y \\ x - 2\,y - 5\,x\,y \end{pmatrix}$ in the domain $(x,y) \in [-1,1]^2$. This vector field has a critical point of an index 0 at $(0,0)$ which consists of two hyperbolic sectors. The critical point $(0,0)$ of the bilinear vector field $\mathbf{v}(x,y) = \begin{pmatrix} x + 2\,y + 10\,x\,y \\ x + 2\,y - 5\,x\,y \end{pmatrix}$ in the domain

[6]This is a straightforward computation for which a formula manipulation program like Mathematica or Maple can be used.

[7]Note that (4.70) is in general not true for the normalized Jacobian defined in (4.25); i.e. in general we have $d_{norm}(\mathbf{J}_\mathbf{v}(\mathbf{x}_{01})) \neq -d_{norm}(\mathbf{J}_\mathbf{v}(\mathbf{x}_{02}))$.

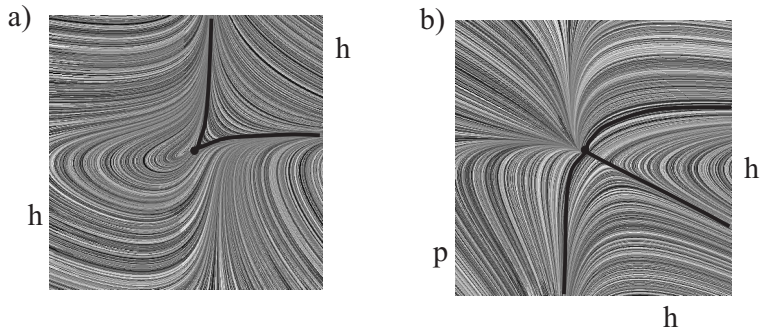


Figure 4.21: a) critical point with index 0 of a bilinear vector field with two hyperbolic sectors; b) critical point with index 0 of a bilinear vector field with two hyperbolic and one parabolic sectors.

$(x, y) \in [-1, 1]^2$ which is shown in figure 4.21b has also the index 0 but consists of two hyperbolic and one parabolic sectors.

Joining two piecewise bilinear vector fields along a common line gives in general a global C^0 interpolant; the tangent curves of the vector field are globally C^1.

4.2.3 Piecewise higher order polynomial interpolation of 2D flow data

Given a scattered flow data set, instead of a piecewise linear interpolation on a triangulation, a piecewise quadratic, cubic, or higher order polynomial interpolation may be applied. Strategies to achieve a higher order continuity of the vector field can be directly taken from the CAGD context. For instance, to obtain a globally C^1 continuous vector field (with globally C^2 tangent curves), a quintic Clough-Tocher interpolant (see [11] and [55]), a quadratic Powell-Sabin interpolant (see [153], [55], and [164] especially for vector fields), or Nielson's C^1 interpolant (see [140], and [164] especially for vector fields) may be applied.

Unfortunately, applying higher order polynomial interpolations to scattered vector data may create unwanted changes in the topology of the vector field: the number of critical points may increase, and collapsing them may lead to new higher order critical points.

A quadratic vector field may have up to four critical points. For example, figure 4.22a shows the quadratic vector field

$$\mathbf{v}(x, y) = \begin{pmatrix} (5\,x - 10\,y + 1)(5\,x - 10\,y + 4) \\ (10\,x - 5\,y - 1)(10\,x - 5\,y - 4) \end{pmatrix}$$

in the domain $[0, 1]^3$. This vector field has the four critical points $\left(\frac{1}{5}, \frac{1}{5}\right)$, $\left(\frac{2}{5}, \frac{3}{5}\right)$, $\left(\frac{3}{5}, \frac{2}{5}\right)$, $\left(\frac{4}{5}, \frac{4}{5}\right)$.

A cubic vector field may have up to 9 critical points. For example, figure 4.22b shows the cubic vector field

$$\mathbf{v}(x, y) = \begin{pmatrix} (14\,x - 40\,y + 3)(14\,x - 40\,y + 13)(14\,x - 40\,y + 23) \\ (42\,x - 20\,y - 1)(42\,x - 20\,y - 11)(42\,x - 20\,y - 21) \end{pmatrix}$$

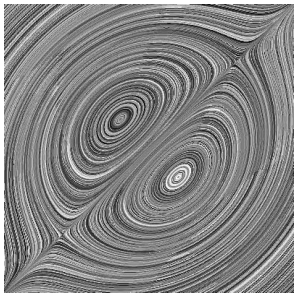
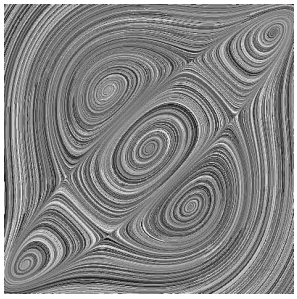
a)  b) 

Figure 4.22: a) quadratic 2D vector field with 4 critical points; b) cubic 2D vector field with 9 critical points.

in the domain $[0,1]^3$. This vector field has the 9 critical points $\left(\frac{1}{14},\frac{1}{10}\right)$, $\left(\frac{5}{14},\frac{1}{5}\right)$, $\left(\frac{9}{14},\frac{3}{10}\right)$, $\left(\frac{3}{14},\frac{2}{5}\right)$, $\left(\frac{1}{2},\frac{1}{2}\right)$, $\left(\frac{11}{14},\frac{3}{5}\right)$, $\left(\frac{5}{14},\frac{7}{10}\right)$, $\left(\frac{9}{14},\frac{4}{5}\right)$, $\left(\frac{13}{14},\frac{9}{10}\right)$.

In general, a polynomial vector field of degree n has up to n^2 critical points.

Applying higher order polynomial vector fields may not only create new first order critical points, their collapsing may also lead to the appearance of higher order critical points.

If the flow data set is on a rectangular grid, one can apply a higher order bipolynomial interpolation instead of the bilinear interpolation. Here similar statements to the case of polynomial interpolation apply: the construction of higher order continuous vector fields is well-understood while the controlled treatment of higher order critical points is rarely investigated. Applying a biquadratic interpolation, up to 8 critical points may appear. Figure 4.23a shows the biquadratic vector field

$$\mathbf{v}(x,y) = \begin{pmatrix} (x-\frac{1}{5})(y-\frac{2}{5})(x-\frac{3}{5})(y-\frac{4}{5}) \\ \\ (x-\frac{2}{5})(y-\frac{1}{5})(x-\frac{4}{5})(y-\frac{3}{5}) \end{pmatrix}$$

in the domain $[0,1]^3$. This vector field has the 8 critical points $\left(\frac{1}{5},\frac{1}{5}\right)$, $\left(\frac{3}{5},\frac{1}{5}\right)$, $\left(\frac{3}{5},\frac{3}{5}\right)$, $\left(\frac{1}{5},\frac{3}{5}\right)$, $\left(\frac{2}{5},\frac{2}{5}\right)$, $\left(\frac{4}{5},\frac{2}{5}\right)$, $\left(\frac{4}{5},\frac{4}{5}\right)$, $\left(\frac{2}{5},\frac{4}{5}\right)$.

A bicubic vector field may produce up to 18 critical point. Figure 4.23b shows the bicubic vector field

$$\mathbf{v}(x,y) = \begin{pmatrix} (x-\frac{1}{7})(y-\frac{2}{7})(x-\frac{3}{7})(y-\frac{4}{7})(x-\frac{5}{7})(y-\frac{6}{7}) \\ \\ (x-\frac{2}{7})(y-\frac{1}{7})(x-\frac{4}{7})(y-\frac{3}{7})(x-\frac{6}{7})(y-\frac{5}{7}) \end{pmatrix}$$

in the domain $[0,1]^3$. This vector field has the 18 critical points $\left(\frac{1}{7},\frac{1}{7}\right)$, $\left(\frac{3}{7},\frac{1}{7}\right)$, $\left(\frac{5}{7},\frac{1}{7}\right)$, $\left(\frac{1}{7},\frac{3}{7}\right)$, $\left(\frac{3}{7},\frac{3}{7}\right)$, $\left(\frac{5}{7},\frac{3}{7}\right)$, $\left(\frac{1}{7},\frac{5}{7}\right)$, $\left(\frac{3}{7},\frac{5}{7}\right)$, $\left(\frac{5}{7},\frac{5}{7}\right)$, $\left(\frac{2}{7},\frac{2}{7}\right)$, $\left(\frac{4}{7},\frac{2}{7}\right)$, $\left(\frac{6}{7},\frac{2}{7}\right)$, $\left(\frac{2}{7},\frac{4}{7}\right)$, $\left(\frac{4}{7},\frac{4}{7}\right)$, $\left(\frac{6}{7},\frac{4}{7}\right)$, $\left(\frac{2}{7},\frac{6}{7}\right)$, $\left(\frac{4}{7},\frac{6}{7}\right)$, $\left(\frac{6}{7},\frac{6}{7}\right)$. In general, a bipolynomial interpolation of degree n produces up to $2\,n^2$ critical points.

A globally C^1 or C^2 continuous vector field can be achieved by applying an interpolating Bézier- or B-spline surface approach (see [55]).

Generally we can make the following statement about higher order (bi)polynomial interpolation: principally they are able to represent higher order critical points, but it is non-trivial (or even impossible) to control this process.

a) b)

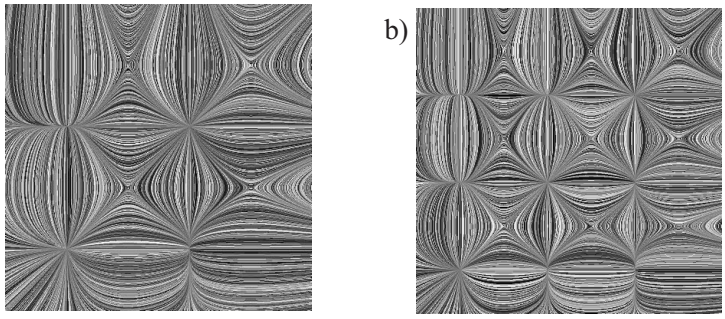


Figure 4.23: a) biquadratic 2D vector field with 8 critical points; b) bicubic 2D vector field with 18 critical points.

In fact it is rather hard to detect higher order critical points in a higher order (bi)polynomial interpolation.

4.2.4 Interpolation of 3D flow data

For 3D scattered flow data, the standard interpolation technique is a piecewise linear interpolation over a tetrahedrization of the sample points. Given the sample points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4 \in \mathbb{E}^3$ with the vectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, $\mathbf{v}_4 \in \mathbb{R}^3$, the vector field inside the tetrahedron $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$ is given by

$$\mathbf{v}(x_1, x_2, x_3, x_4) = x_1\,\mathbf{v}_1 + x_2\,\mathbf{v}_2 + x_3\,\mathbf{v}_3 + x_4\,\mathbf{v}_4 \tag{4.71}$$

where (x_1, x_2, x_3, x_4) are the barycentric coordinates of a location relative to the vertices $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, i.e. $x_1 + x_2 + x_3 + x_4 = 1$.

As in the 2D case, there is a one-to-one relation between linear vector fields and first order critical points. A linear 3D vector field has up to one non-degenerate critical point which is of first order. Conversely, every 3D first order critical point can be constructed by a first order linear vector field.

Instead of a piecewise linear interpolation, piecewise higher order polynomial interpolation may be applied to obtain a higher order continuity of the vector field and its tangent curves. A quadratic vector field may have up to 8 critical points. In general, a polynomial vector field of the degree n has up to n^3 distinct first order critical points.

For 3D flow data on a 3D rectangular grid, a piecewise trilinear interpolation over each grid cell is the most common approach. Doing so, up to 6 distinct first order critical points may appear inside a grid cell. For example, the trilinear vector field

$$\mathbf{v}(x, y, z) = \begin{pmatrix} (x - \frac{1}{10})(y - \frac{2}{10})(z - \frac{3}{10}) \\ (x - \frac{4}{10})(y - \frac{5}{10})(z - \frac{6}{10}) \\ (x - \frac{7}{10})(y - \frac{8}{10})(z - \frac{9}{10}) \end{pmatrix}$$

in the domain $[0, 1]^3$ has the 6 critical points $(\frac{1}{10}, \frac{5}{10}, \frac{9}{10})$, $(\frac{1}{10}, \frac{8}{10}, \frac{6}{10})$, $(\frac{4}{10}, \frac{2}{10}, \frac{9}{10})$, $(\frac{4}{10}, \frac{8}{10}, \frac{3}{10})$, $(\frac{7}{10}, \frac{5}{10}, \frac{3}{10})$, $(\frac{7}{10}, \frac{2}{10}, \frac{6}{10})$. This fact has the following consequences:

- Since the detection of critical points in a piecewise trilinear vector field ends in solving polynomials of degree 6, no closed solution of this problem exists. Instead, numerical solutions have to be applied.
- The first order critical points of a trilinear vector field may collapse to higher order critical points.

In general, a trilinear interpolation gives a globally C^0 vector field (with globally C^1 tangent curves). A higher order continuity of the vector field can be achieved by applying a piecewise higher order tripolynomial interpolation. As in the 2D case, this increases the number of possible critical points (and therefore the number of possible higher order critical points by collapsing the first order critical points). In general, a tripolynomial vector field of degree n has up to $6\,n^3$ critical points.

4.2.5 Choosing the appropriate interpolation

After introducing different kinds of interpolation schemes on flow data, this chapter treats the problem of choosing an appropriate one for a given flow data set. The choice depends on the following aspects:

1. The desired continuity of the vector field

   If a globally C^0 continuous vector field is sufficient, linear or bilinear (or trilinear) interpolations are appropriate. The desired continuity depends on the grid resolution and on the visualization technique to be chosen. If the grid cells have approximately pixel size, the interpolation issue does not play an important role. In this case a piecewise (bi/tri)linear interpolation (or even a piecewise constant interpolation) is sufficient. If the visualization technique to be used is based on first order approximations of the vector field (for instance numerical integration of tangent curves, see section 4.3.1.1), a piecewise (bi/tri)linear interpolation is also sufficient: more accurate interpolation results will be destroyed by the visualization process in this case.

2. Additional information about the topology of the vector field to be constructed

   If for a given flow data set it is known that higher order critical points appear in the flow, interpolation methods which can treat these critical points should be applied. Such additional information about the topology can be obtained in two ways:

   - The appearance of higher order critical points may be predicted due to symmetry reasons. For instance, the ideal flow around a ship propeller consisting of 6 symmetric segments might produce a critical point with 6 similar regions. Another example for the appearance of higher order critical points are magnetic fields inside coils or transformers (see [162]).
   - Higher order critical points may be predicted by estimations on the usual piecewise (bi)linear interpolation. If the (bi-)linear interpolation contains clusters of first order critical points, it may be assumed that the original flow has a higher order critical point in this cluster.

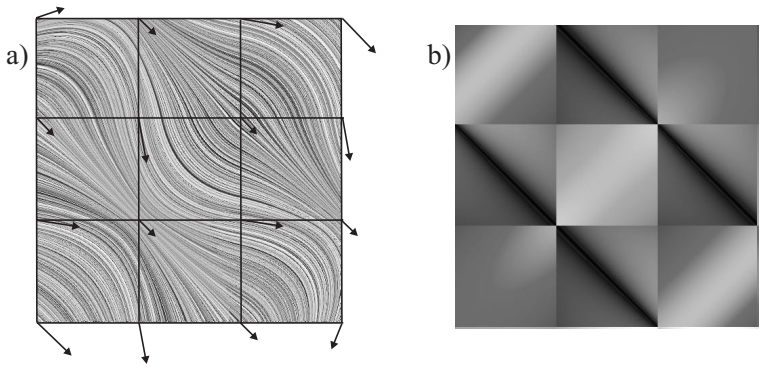


Figure 4.24: Piecewise bilinear vector field on a regular 4×4 grid; a) Integrate and Draw; b) curvature plot.

3. Information about the history of the flow data

   If the flow data is obtained by a numerical simulation based on a finite element analysis, higher order critical points are already excluded in the step of creating the flow data. Thus higher order critical points do not have to be considered for the visual analysis. If the flow data is obtained by measuring a real flow at certain sample points, the appearance of higher order critical points has to be taken into consideration.

4. Computing costs

   Higher order interpolations are more computing intensive than (bi)linear interpolations. The enhancement of their application has to be weighted against the additional computing costs.

Taking the points mentioned above into consideration, the interpolation problem for flow data can be formulated as follows: Given a flow data set, an interpolation of a certain demanded continuity has to be specified which is able to reproduce topologies of a certain order in a controlled way. Furthermore, this interpolation should not exceed a certain amount of computing cost.

Obviously, we have to find a compromise between these demands. To illustrate that this is not trivial, consider the flow data set on a regular 4×4 data set shown in figure 4.24a. The application of a piecewise bilinear interpolation gives a globally C^0 vector field with C^1 tangent curves. The curvature plot of this vector field (figure 4.24b) shows discontinuities between the grid cells. This shows that the tangent curves are not curvature continuous, thus they are not C^2 continuous, thus the vector field is not globally C^1.

To obtain a globally C^1 continuous vector field, a piecewise biquadratic interpolation may be applied, as shown in figure 4.25. The smooth curvature plot in figure 4.25b is a (necessary) indicator that the vector field is indeed C^1 continuous. Unfortunately, the biquadratic interpolation creates a number of new critical points (figure 4.25a). In fact, the cell in the middle of the grid now has 8 critical points while the piecewise linear interpolation did not have any critical point.

To remove these unwanted critical points and preserve the C^1 continuity of the vector field, a piecewise bicubic interpolation may be applied, as illustrated

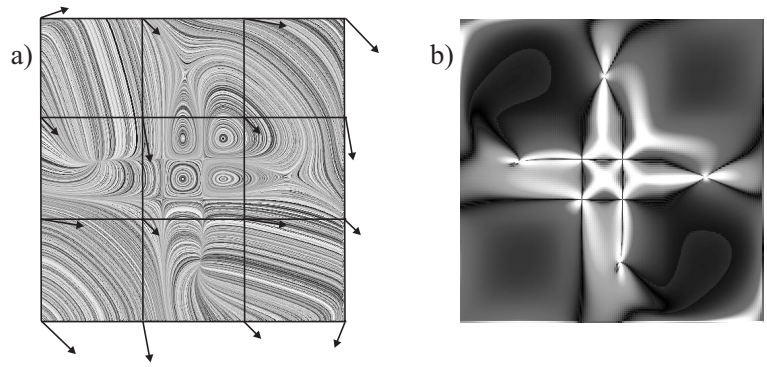


Figure 4.25: Piecewise biquadratic vector field on a regular 4×4 grid; a) Integrate and Draw; b) curvature plot.

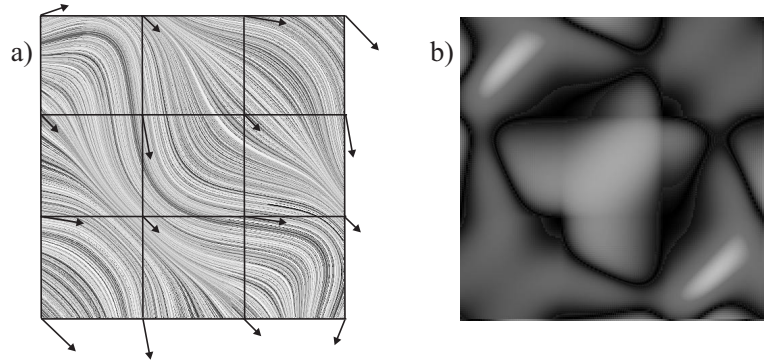


Figure 4.26: Piecewise bicubic vector field on a regular 4×4 grid; a) Integrate and Draw; b) curvature plot.

in figure 4.26. Here we can clearly see that this vector field has the same topology as the bilinear one in figure 4.24, but the smooth curvature plot indicates the C^1 continuity.

Up to now, little research has been done on the issue of finding an appropriate interpolation of given flow data. The first approach to use higher order polynomial interpolation for a controlled representation of higher order topologies can be found in [162] and [163]. There the polynomial vector field is described in terms of Clifford Algebras. This way the approach is able to represent the index of a higher order critical point exactly. The exact order of the segments around the critical point cannot be controlled as well as the joint of the higher order vector field with adjacent linear interpolations is not C^0 continuous. So the approach in [162] and [163] follows the strategy of focusing on an exact representation of the index of the critical points at the expense of the global continuity of the vector field.

The converse approach is introduced in [164]. There a higher order global continuity of the interpolant is emphasized; for two well-known C^1 interpolation schemes on a triangulation, the Sabin-Powell interpolant and Nielson's C^1 interpolant, the impact of their application to the topology is studied. In [164]

it was shown that these interpolation schemes produce better representations for higher order critical points than a piecewise linear interpolation. However, an exact and controlled representation is still not possible.

One approach to giving an exact and controlled representation of higher order critical points is in [194]. There a piecewise linear interpolant is chosen with the special property that the critical point lies on a vertex of the underlying triangulation. Originally introduced for the simplification of vector fields, this approach can be used to deal with higher order critical points of any topology.

Another approach to dealing with vector fields of any topology is described in section 4.4 of this work. Here for any given topology a piecewise linear vector field of exactly this topology is constructed. Approaches to considering general 2D topologies and a general smoothness of a vector field are still unknown.

4.3 Curves and Surfaces for Flow Visualization

Considering the pipeline for flow visualization in figure 4.2 again, we now focus on the mapping step. In this step a suitable visualization technique for the vector field has to be chosen.

In the past decade a variety of visualization techniques for vector fields have been developed. It is not the purpose of this chapter to survey these techniques; surveys of vector field visualization techniques can be found in [152] and [167]. Instead we want to focus on techniques which use curves and surfaces for visualization. Since curves and surfaces are able to encode higher amounts of information, they are candidates for such large data sets like flow data and the vector fields derived from them.

The existing visualization techniques can be divided into three classes (see [167]): elementary methods, local methods, and global methods. Each of these classes of methods is treated in one of the sections 4.3.1 – 4.3.3.

4.3.1 Elementary methods

Elementary methods show properties of the vector field at a number of selected locations. The simplest representatives of these class of methods are arrow plots. Here we treat tangent curves and stream surfaces.

4.3.1.1 Visualizing tangent curves

In section 4.1 we introduced the concept of tangent curves for vector fields. Since they describe the path of a massless particle in a steady flow, the drawing of a certain number of tangent curves may give an intuitive impression of the flow. Indeed, techniques which visualize tangent curves or their properties are widespread and common in vector field visualization. To draw tangent curves, two problems have to be solved:

1. The tangent curves have to be integrated
2. A selection of which tangent curves to draw has to be made.

To 1.: Tangent curves are usually given in an implicit representation of (4.9). For a visualization, an explicit representation as a parametric curve would be optimal. Unfortunately, for sufficiently complicated vector fields, no explicit

Figure 4.27: Stream line of a flow of a Bay area of the Baltic Sea near Greifswald (Greifswalder Bodden); data set provided by department of Mathematics of the University of Rostock.

description of the tangent curves exists. In fact, an explicit representation of the tangent curves exists only for piecewise linear vector fields. In [141] a representation of tangent curves of a linear vector field as parametric exponential curve is described. In section 4.4 of this work we use the fact that under certain conditions a tangent curve of a linear vector field is a quadratic curve segment to construct vector fields of a given topology.

If the vector field is more complicated than piecewise linear, numerical integration methods have to be applied. Here the standard method is a fourth order Runge-Kutta method (see [167]). [64] gives a comparison of other known integration techniques.

Numerical integration techniques of tangent curves are based on a local lower order Taylor expansion of the vector field. Thus their numerical integration involves a certain inaccuracy of the tangent curves. Unfortunately, these inaccuracies are increased as the integration proceeds. Especially in areas around higher order critical points the inaccuracies of the numerical integration methods may destroy the exact topology of these critical points. A study on the accuracy of numerical tangent curve integration methods can be found in [129].

To 2.: Tangent curves in a vector field are dense. To visualize them, a selection of the tangent curves to be visualized has to be done. Here two extreme cases have to be avoided: if the tangent curves are too close to each other, they tend to be not distinguishable any more. On the other hand, if the tangent curves are too far away from each other, important information may be missed. There are several strategies for placing an appropriate number of stream lines. In figure 4.27, all stream lines passing the grid points of a certain rectangular grid[8] are drawn. As a result we have an overview of the flow behavior in most parts of the flow. Nevertheless there are regions where the stream lines are too

[8]This grid is not necessarily the grid of the original flow data set.

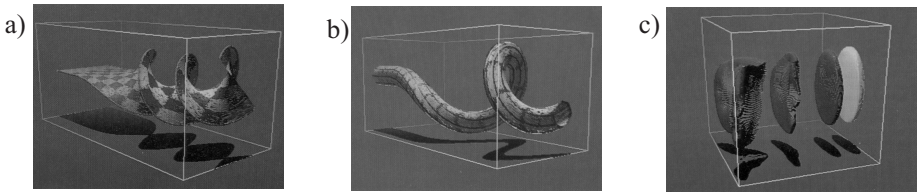


Figure 4.28: a) stream surface; b) stream tube; c) stream objects; (from [180]).

dense or too coarse.

More involved tangent curve selection methods exist. In [104] an approach is introduced to control the distance of adjacent tangent curves. In [196] the stream line selection is guided by visual attributes.

For 3D vector fields the treatment of tangent curves works principally in the same way. Due to ambiguous projections of the 3D curves onto the 2D screen the problem of getting an appropriate selection of tangent curves becomes even more important. In [213], 3D illuminated stream lines are used to enable the user to recognize the location in space.

For unsteady vector fields, stream lines, streak lines, path lines or time lines may be visualized. Doing so, the ideas and problems are principally the same as for tangent curves in a steady vector field.

An approach similar to the visualization of tangent curves is *particle tracing*. Here a tangent curve is represented by the trajectory of a small particle over time. The problems of computing the path of the particle coincide with the problems of computing tangent curves.

4.3.1.2 Stream surfaces, stream objects

Stream surfaces are generalizations of tangent curves in a 3D vector field. The idea is to consider not only the path of one particle but the paths of all particles set out on an initial curve. This curve may be a line segment (resulting in a stream surface) or a circle (resulting in a stream tube). In a similar way, the approach of particle tracing can be extended to stream objects by considering the location of a particle set originally located on a certain surface over time. Figure 4.28 gives an illustration of stream surfaces, stream tubes, and stream objects.

Similarly to tangent curves, stream surfaces and stream objects are generally computed by numerically integrating a certain number of tangent curves and connecting them. If two adjacent tangent curves on a stream surface move too far away from each other, the tracing of an additional tangent curve between them may be started. Approaches to finding an explicit parametric description of stream surfaces do not seem to exist in the context of scientific visualization.

4.3.2 Local methods

As elementary methods, local methods show properties of the vector field in selected points. In addition, information from the neighborhood of the selected points is visualized as well. From the variety of existing local techniques we treat

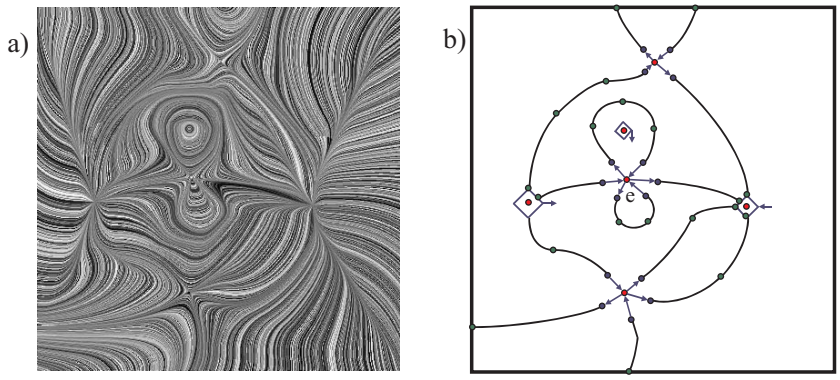


Figure 4.29: a) 2D vector field; b) topological skeleton.

the following which make use of curves/surfaces and their properties: topological skeletons, flow ribbons, and local probes.

4.3.2.1 Visualizing the topological skeleton

In section 4.1.3 we introduced the topology of a vector field as consisting of critical points and separatrices. Visualizing these two features of a vector field gives a topological skeleton which allows the user to infer the behavior of the vector field in any point. This idea was introduced in [87] and [88] to scientific visualization. For 2D vector fields with a rather simple topology (i.e. the number of critical points and separatrices is not too large), this method gives a very effective and intuitive graphical representation of the vector field. However, if the topology is more complicated, topology simplification algorithms ([43], [194]) may be applied. Except for the computing costs and overlaying effects in the visualization for vector fields with a rich topology, the visualization of topological skeletons suffers from two more drawbacks:

- If the vector field contains higher order critical points, they are hardly detectable in such a way that the correct order of the different areas of flow behavior around a critical points is obtained. Thus the separatrices originating and ending in this higher order critical point can generally not be detected and visualized correctly.

- If a separatrix does not originate or end in a critical point and separate two regions of different flow behavior there (i.e. if the separatrix is not of type 1 - see section 4.1.2), it may be missed out by the topology extraction step.

However, the topological skeleton has been proven to be a successful technique for vector fields containing only first order critical points and separatrices of the type 1 (see section 4.1.2). Figure 4.29b illustrates the topological skeleton of the vector field shown in figure 4.29a.

Due to the still undefined concept of topology for unsteady 2D vector fields, extensions of the concept of topological skeleton to this class of vector fields seem not to exist.

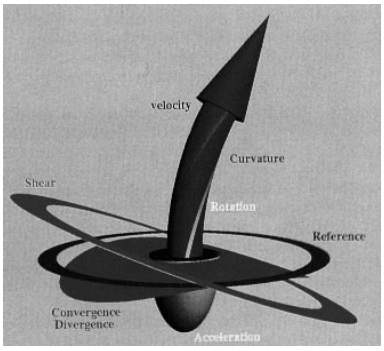


Figure 4.30: A local probe which represents – among other measures – the curvature of the tangent curve in the selected locations of the vector field (from [44]).

To extend the method to 3D vector fields, [69] gives an approach to represent the topology of a first order critical point while the concept of separatrices seems not to have been used yet for visualization purposes.

4.3.2.2 Flow ribbons

Flow ribbons are based on tangent curves in a 3D vector field. There the tangent curves are not visualized as line segments but as narrow ruled surfaces following the flow. (See [55] for an introduction to ruled surfaces). This way additional information can be coded into size and orientation of the ruled surface. One common measure to be encoded by flow ribbons is the rotation (a vector field - see section 4.1.8) in each point of the tangent curve. This vector defines orientation and size of the ribbon in every point (see [96]).

Since flow ribbons are based on the integration of tangent curves, all advantages and disadvantages of tangent curves (described above) apply as well. The rotation of the vector field can be directly computed in each point of the tangent curve; no further numerical problems in computing the ruled surface appear.

4.3.2.3 Probes

The idea of probes is to place certain 3D icons at selected locations of the vector field. These icons encode local properties of the vector field in this point. These properties may be geometric properties of curves and surfaces. [44] introduces a local probe which encodes – among others – the curvature of the tangent curve in the selected points. See figure 4.30 for an illustration. Probes which encode other curve and surface properties like torsion of the tangent curve or Gaussian and Mean curvature of the perpendicular surfaces (see section 4.1.8) are possible as well.

4.3.3 Global methods

Global methods show the behavior of the entire vector field. Hence they do not focus on certain locations in the vector field. The general approach is to map

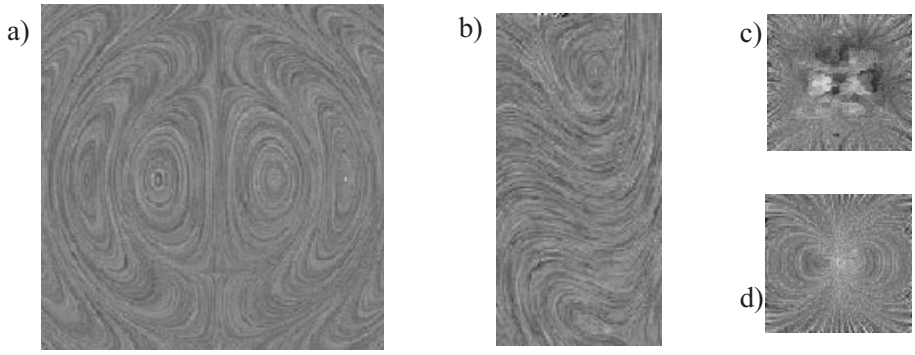


Figure 4.31: Examples of LIC images; a) electrostatic field of a dipole; b) flow around a cylinder; c) electrostatic field of a Benzene molecule; d) electrostatic field of a water molecule; data sets courtesy of Konrad-Zuse-Institut Berlin (Germany); (images from [156]).

relevant properties of the vector field to a scalar field which can be rendered using standard techniques.

Among the existing global methods there are a number of methods which use curves and surfaces – directly or indirectly by visualizing their properties – for encoding the important information on the vector field. Here we treat the techniques LIC and Spot Noise, Integrate&Draw, and curvature plots.

4.3.3.1 LIC and Spot Noise

LIC (Line Integral Convolution, [26]) and Spot Noise ([198]) are techniques which map a 2D vector field into a 2D scalar field which emphasizes the recognition of the behavior of all tangent curves. Following the fact that the recognition of tangent curves gives an intuitive visual impression of the vector field, LIC and Spot Noise create scalar fields which show as many tangent curves as possible without overlaying them.

The idea of Spot Noise is to transform a certain input texture in the vector field into the direction of the flow.

LIC uses a grey-valued noisy input texture and convolutes this into the flow direction (see [26] and [167] for details). This way the resulting texture changes its color only slightly in flow direction while rapid changes appear in the direction perpendicular to the flow. Although the resulting texture may look rather blurry, it gives a good impression of the behavior of the vector field. Figure 4.31 shows LIC images for a number of vector fields.

To apply LIC, a local numerical tangent curve integration has to be applied for each pixel. This procedure makes LIC rather time consuming. The FAST LIC algorithm introduced in [178] speeds up the LIC algorithm significantly. Using the fact that most of the information computed to obtain the color of a certain point in the vector field can be reused to compute the color of the adjacent points into flow direction, [178] applies the convolution not pixelwise but along the stream lines. To do so, [178] provides strategies to select tangent curves in such a way that each pixel is covered by at least one tangent curve. If a pixel is covered by more than one tangent curve, its final color is obtained by

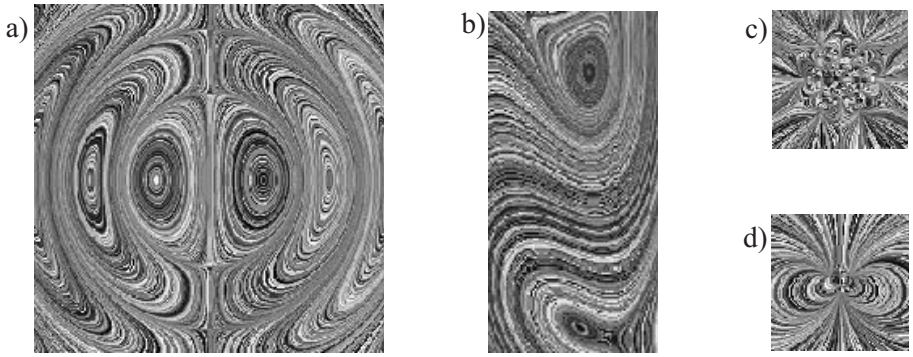


Figure 4.32: Examples of Integrate&Draw images show a visual improvement to the LIC images of figure 4.31; a) electrostatic field of a dipole; b) flow around a cylinder; c) electrostatic field of a Benzene molecule; d) electrostatic field of a water molecule; data sets courtesy of Konrad-Zuse-Institut Berlin (Germany); (images from [156]).

averaging the colors of the tangent curves through it.

The LIC concept has been extended in various ways. In order to additionally visualize the orientation of the flow, [203] uses asymmetric filter kernels. [60], [14] and [131] apply LIC to vector fields on surfaces. [61] and [173] use LIC for unsteady flows, while [101] and [155] study LIC for 3D vector fields. LIC and Spot Noise are compared in [42].

4.3.3.2 Integrate&Draw

Starting from the idea of LIC, [156] goes a step further to create scalar fields of a 2D vector field, which enables the user to recognize the behavior of the tangent curves. As in the FAST LIC case, a set of tangent curves has to be found which covers all pixels at least once. Instead of using these tangent curves to convolute an input textures, [156] simply draws each of them in a random grey color. If more than one tangent curve passes a pixel, a weighted average of their grey values is computed as

$$\bar{g} = \frac{g_1 + 2\,g_2 + ... + n\,g_n}{1 + 2 + ... + n}$$

where $\bar{g}$ is the final grey value of a pixel covered by n tangent curves with the grey values $g_1, ..., g_n$. This weighted average ensures that for high numbers of tangent curves in a pixel the average grey value does not converge to 0.5.

The application of LIC gives images where the tangent curves are clearer visible than in the LIC case. Figure 4.32 illustrates this. [156] states that the computation cost of Integrate&Draw is also lower than for FAST LIC.

4.3.3.3 2D curvature plots

In section 4.1.5 we introduced the curvature $\kappa(\mathbf{v})$ of a 2D vector field $\mathbf{v}$ as a derived scalar field. A logical next step is to try to use this scalar field as a global visualization technique for vector fields.

Following [185], we want to visualize the curvature κ of a 2D vector field in the following way: compute κ for every point of the domain and color code these

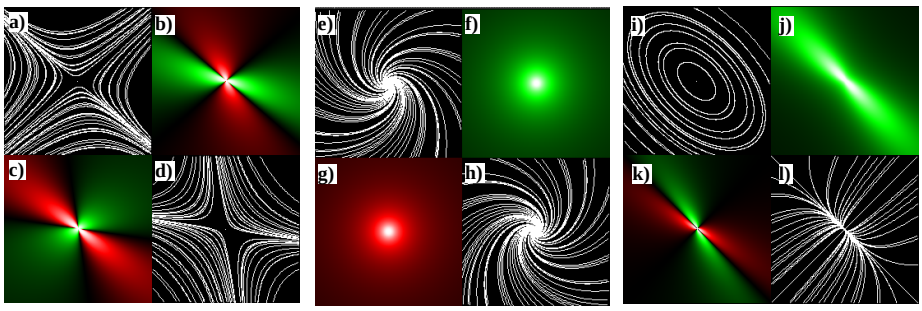


Figure 4.33: Linear vector field with saddle point (a..d); linear vector field with repelling focus (e..h); linear vector field with center (i..l).

values. To do this we use a continuous color coding map with the following properties: a negative value is mapped to a green color, a positive value is mapped to a red color. The higher the magnitude of the value the lighter the color gets. A zero value gives black; if the value diverges to plus (minus) infinity the red (green) color tends to white.

The pictures a-d of figure 4.33 give an example of the vector field $\mathbf{v}(x, y) = \begin{pmatrix} 1 \\ 3 \end{pmatrix} x + \begin{pmatrix} 4 \\ 1 \end{pmatrix} y$. This linear vector field has a critical point at $(0, 0)$ - a repelling saddle. Figure 4.33a shows its numerical tangent curve integration. Figure 4.33b is the visualization of its curvature. Figures 4.33d and 4.33c show the same for the perpendicular vector field $\mathbf{v}^\perp$. In this case, $\mathbf{v}^\perp$ has a repelling saddle at $(0, 0)$ as well.

The reason for visualizing the curvature of both $\mathbf{v}$ and $\mathbf{v}^\perp$ is shown by considering the following visualization properties:

In the curvature visualization b) of figure 4.33 the critical point appears as highlight. Considering (4.21), $\kappa(\mathbf{v})$ tends to infinity only if the denominator of κ tends to 0. This occurs only at critical points. Therefore, a highlight in the curvature visualization always indicates a critical point in the vector field. The reverse question arises: does every critical point produce a highlight in the curvature visualizations? The answer is yes, if we exclude certain degenerate points. A *degenerate critical point* of a vector field $\mathbf{v}$ is a critical point where the directions of the vectors of $\mathbf{v}$ do not change in the neighborhood of the critical point. For non-degenerate critical points, we have the following

Theorem 5 *In the neighborhood of a non-degenerate critical point of a 2D vector field* $\mathbf{v}$*, the curvature of* $\mathbf{v}$ *or* $\mathbf{v}^\perp$ *(or both curvatures) tend to infinity.*

An exact definition of a degenerate critical point and the proof of this theorem can be found in [185]. The same theorem can be formulated in the following way: non-degenerate critical points in a vector field $\mathbf{v}$ always produce highlights in the visualization of the curvature of $\mathbf{v}$ or $\mathbf{v}^\perp$.

Considering the curvature visualizations b) and c) of figure 4.33 again, another question arises: Do the curvature visualizations of $\mathbf{v}$ and $\mathbf{v}^\perp$ contain all information about $\mathbf{v}$? The answer is given by

Theorem 6 *Given are two 2D vector fields* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *which have non-constant direction fields. If* $\kappa(\mathbf{v}_1) = \kappa(\mathbf{v}_2)$ *and* $\kappa({\mathbf{v}_1}^\perp) = \kappa({\mathbf{v}_2}^\perp)$ *then the directions of the vectors of* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *coincide in every point.*

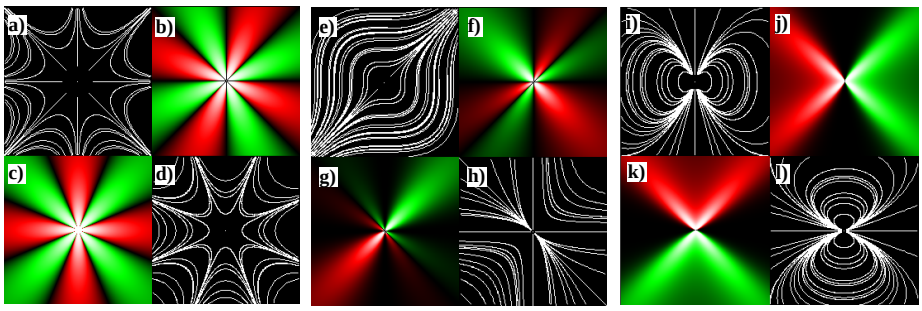


Figure 4.34: Higher order saddle point (a..d); critical point with two elliptic sectors (e..h); dipole (i..l).

See [185] for a proof. Theorem 6 has an interesting consequence: the curvatures of $\mathbf{v}$ and $\mathbf{v}^\perp$ together contain all information about the directions of the vectors in $\mathbf{v}$. Therefore, the curvatures of $\mathbf{v}$ and $\mathbf{v}^\perp$ contain all information about the topology of $\mathbf{v}$. This statement is true for vector fields of general topology.

Pictures e-h of figure 4.33 show a linear vector field with a repelling focus. Figure 4.33e is the numerical stream line integration, figure 4.33f is the curvature visualization. Figures 4.33h and 4.33g show the same for the perpendicular vector field. The repelling focus appears completely green around the highlight in the curvature visualization and completely red in the curvature visualization of the perpendicular vector field. Figures 4.33 i-l show the visualization of a center. It appears completely green around the highlight in the curvature visualization (figure 4.33j) and has 4 different areas (colored red or green) each of 90 degrees in the perpendicular curvature visualization (figure 4.33k).

Figure 4.34 shows a collection of higher order critical points. None of these points can be treated using the topology methods of [87] but their curvature visualization gives a fairly good impression of them. Figures 4.34 a-d show a saddle point with 4 pairs of tangent curves through it. In the curvature visualization (figure 4.34b) we have eight differently colored sections around the critical point. The perpendicular field (figure 4.34c) has eight different sections as well. Figure 4.34 e-h shows the visualization of the vector field $\mathbf{v}(x, y) = (y^2, x^2)^T$ in the range $[-1, 1] \times [-1, 1]$. This vector field has a critical point with two elliptic sections in $(0, 0)$. Observing the stream line integration (figure 4.34e), this critical point may be missed. The curvature visualization (figure 4.34f) shows it clearly as a highlight with six differently colored sections around it. Here the visualization of the perpendicular curvature has two differently colored areas (figure 4.34g). Figures 4.34 i-l show the visualization of a vector field describing a dipole. Both the visualization of its curvature (figure 4.34j) and its perpendicular curvature (figure 4.34k) show two differently colored sections around the highlighted critical point.

A general algorithm which infers the topology of higher order critical points from the curvature visualizations is still unknown. Nevertheless, the higher order critical points of figure 4.34 can be clearly distinguished from the first order critical points of figure 4.33 by their curvature visualizations.

An implementation of the curvature of 2D vector fields (as well as Integrate & Draw) has been realized in the system CurVis. This system also uses adaptive image generation to provide flow visualizations to remote clients over the

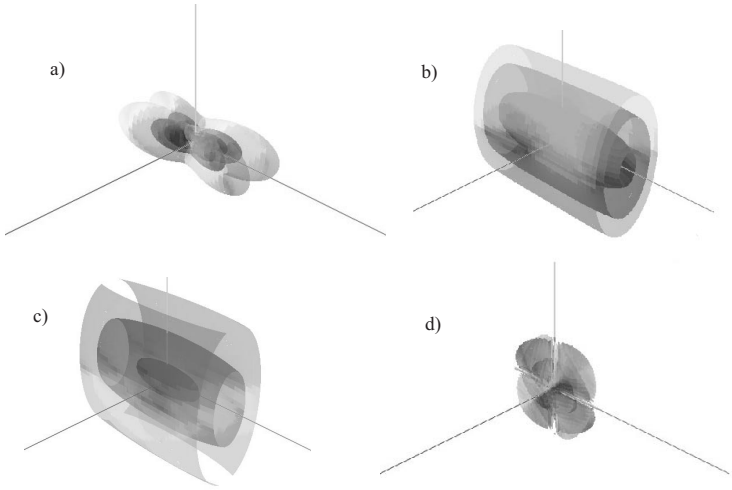


Figure 4.35: Linear vector field with a first order critical point of the type (RN,RN,RN); a) curvature $\kappa(\mathbf{v})$; b) Gaussian curvature $K(\mathbf{v})$; c) Mean curvature $H(\mathbf{v})$; a) torsion $\tau(\mathbf{v})$; (from [205]).

internet[9].

Another concept of curvature of vector fields is used [207]. There the vector field is converted into a scaler field called level set. The local curvature of the isosurfaces of this level set is computed and used as input of an optimization process of the vector field.

4.3.3.4 3D curvature plots

In [205] the usage of curvature plots was extended to the visualization of 3D vector fields. As shown in the previous section, in the 2D case the curvature of a vector field $\mathbf{v}$ and the curvature of its perpendicular vector field $\mathbf{v}^{\perp}$ were useful features for visualization. The scalar fields we consider for a 3D vector field are the curvature $\kappa(\mathbf{v})$ of the tangent curves, the torsion $\tau(\mathbf{v})$ of the tangent curves, the Gaussian curvature $K(\mathbf{v})$ of the normal surfaces, and the Mean curvature $M(\mathbf{v})$ of the normal surfaces (see section 4.1.8).

In [205] the behavior of these scalar fields around first order critical points is studied. The results are somewhat similar to the 2D case: around first order critical points at least one of the scalar fields κ, τ, K, H tends to infinity. Thus the visualization of these scalar fields enables the user to visually detect critical points. The topological classification of the 3D critical points can also be obtained from these scalar fields. To visualize these 3D scalar fields, [205] extracts a number of isosurfaces using a Marching Cubes-like algorithm on an adapted octree structure. Depending on the shapes of the isosurfaces around a 3D critical point, its topological classification can be inferred. Figure 4.35 shows the visualization of curvature, torsion, Gaussian curvature and Mean curvature of a vector field around around a first order critical point of the type (RN,RN,RN)[10] .

[9]Curvis can be accessed at
`http://www.informatik.uni-rostock.de/Projekte/movi/IIS/curvisrdr.html`

[10]This abbreviation means that in each eigenplane of the Jacobian matrix in the critical

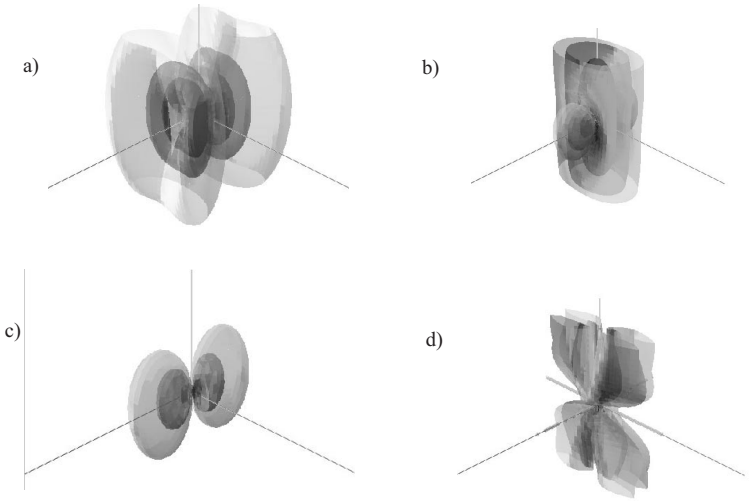


Figure 4.36: Linear vector field with a first order critical point of the type (AN,Sa,Sa); a) curvature $\kappa(\mathbf{v})$; b) Gaussian curvature $K(\mathbf{v})$; c) Mean curvature $H(\mathbf{v})$; a) torsion $\tau(\mathbf{v})$; (from [205]).

Figure 4.36 shows the same features around a first order critical point of another type (AN,Sa,Sa)[11] .

Although [205] shows the usefulness of the visualization of the scalar fields for detecting and classifying first order critical points, their behavior around general 3D critical points is still unknown. This is mainly due to the fact that a classification of general 3D critical points does not seem to exist in the context of scientific visualization.

In [160] the curvature and torsion of a vector field is used to detect vortex core lines.

4.4 Design of Vector Fields

Up to here, the vector fields we have considered for visualization were obtained from measured or simulated flow data sets. This section introduces a new way of obtaining vector fields: by design. The design of vector fields is strongly related to the ideas of curve and surface design in the CAGD context.

In CAGD, a curve/surface is designed by interactively moving a skeleton of control points. Out of this skeleton of control points, the resulting curve/surface is constructed. The skeleton is supposed to contain the essential information of the curve/surface in an intuitive way.

Transferring these ideas to the construction of vector fields, we have to find a skeleton of a vector field which describes the essential information of the vector field. As introduced in section 4.1.3, the topological skeleton is a good candidate for this.

point the 2D vector field has a repelling node behavior.

[11]This abbreviation means that in one eigenplane of the Jacobian matrix in the critical point the 2D vector field has an attracting node behavior, while in the two remaining eigenplanes the 2D vector fields have saddle points.

The construction of vector fields out of their topological skeletons has the following applications:

- As shown in section 4.3.2.1, for sufficiently complicated vector fields of general topology it is impossible to determine their exact topology. Hence it is impossible to judge if a new visualization technique represents the topology of a given vector field correctly. The topology-based construction of vector fields gives vector fields of a known topology of any complexity. These vector fields may serve as test data to evaluate the topological behavior of flow visualization techniques.
- For simple vector fields, the topological skeleton can be extracted automatically; based on this skeleton a new vector field can be constructed, which may be a compressed version of the original one consisting of exactly the same topology. Thus a construction of vector fields can be used for vector field compression.

The rest of this section 4.4 is organized in the following way: section 4.4.1 introduces how to describe the topological skeleton of a vector field as a set of control points and -polygons. Section 4.4.2 constructs a vector field out of this set of control points. Section 4.4.3 applies this method for the compression of vector fields.

4.4.1 Control polygons to describe the topological skeleton

In CAGD, the skeleton of a curve/surface is described by a set of control polygons. In a similar way we want to describe the topological skeleton of a 2D vector field as a certain set of control polygons.

Since the topology of a 2D vector field consists of critical points and separatrices, we have to find control polygons both for critical points and separatrices. As introduced in section 4.1.1, a critical point is topologically classified by the sectors of different flow behavior. These sectors are separated by separatrices.

To describe a critical point consisting of n different sectors, we use a convex closed polygon $(\mathbf{p}_0, ..., \mathbf{p}_{n-1})$ and a point $\mathbf{p}$ inside this polygon. Then $\mathbf{p}$ denotes the location of the critical point while the n separatrices are denoted by the n line segments $(\mathbf{p}, \mathbf{p}_0), (\mathbf{p}, \mathbf{p}_1), ..., (\mathbf{p}, \mathbf{p}_{n-1})$. Since each separatrix has either an inflow or outflow behavior, each of the line segments $(\mathbf{p}, \mathbf{p}_0), (\mathbf{p}, \mathbf{p}_1), ..., (\mathbf{p}, \mathbf{p}_{n-1})$ has to be marked either as inflow or outflow. Then the n areas of different flow behavior are defined by the n triangles $(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{(i+1) \bmod n})$ for $i = 0, ..., n-1$. If for an area $(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{(i+1) \bmod n})$ both separatrices $(\mathbf{p}, \mathbf{p}_i)$ and $(\mathbf{p}, \mathbf{p}_{(i+1) \bmod n})$ denote inflow (or both areas denote outflow), the triangle $(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{(i+1) \bmod n})$ describes a parabolic sector. Otherwise it has to be additionally specified whether the area should describe a hyperbolic or elliptic sector. Figure 4.37a illustrates the control polygon for a critical point consisting of 7 areas of different flow behavior.

Special treatment is necessary for first order critical points of an index of +1. Considering the (γ, r) phase plane classification of critical points introduced in section 4.1.6.2, these are all critical points with $r > \frac{1}{2}$. These critical points consist of only one parabolic sector; thus the general treatment described above

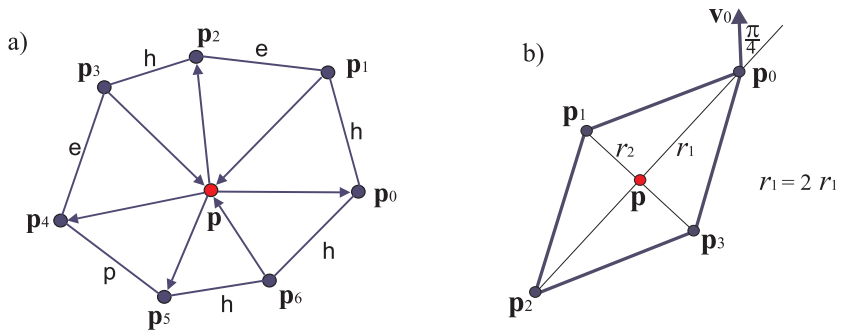


Figure 4.37: a) control polygon of a critical point consisting of 7 areas of different flow behavior: hyperbolic, elliptic, hyperbolic, elliptic, parabolic, hyperbolic, hyperbolic; b) control polygon of a first order critical point of index +1 with (γ, r) coordinates of $\left(\frac{\pi}{4}, \frac{9}{10}\right)$.

fails. To describe them we use a rectangle $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ with $\|\mathbf{p}_1 - \mathbf{p}_0\| = \|\mathbf{p}_2 - \mathbf{p}_1\| = \|\mathbf{p}_3 - \mathbf{p}_2\| = \|\mathbf{p}_0 - \mathbf{p}_3\|$. Then the location $\mathbf{p}$ of the critical point is the center of the rectangle. In addition, for one of the vertices (for instance $\mathbf{p}_0$) the flow direction vector $\mathbf{v}_0$ has to be specified. Then the rectangle $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ shall describe a first order critical point in $\mathbf{p}$ with the following (γ, r) coordinates:

$$\begin{aligned} \gamma &= \text{angle}(\mathbf{p}_0 - \mathbf{p}, \mathbf{v}_0) \\ r &= \frac{1}{2} + \frac{r_1\, r_2}{{r_1}^2 + {r_2}^2} \end{aligned}$$

with $r_1 = \|\mathbf{p}_2 - \mathbf{p}_0\|$ and $r_2 = \|\mathbf{p}_3 - \mathbf{p}_1\|$. The orientation of the whole rectangle denotes the domain rotation component of the critical point. Figure 4.37b illustrates the control polygon of a first order critical point with the (γ, r) coordinates of $\left(\frac{\pi}{4}, \frac{9}{10}\right)$.

To design separatrices, we search for a curve scheme which has to fulfill two conditions. On the one hand we need a curve scheme of a high flexibility and smoothness which can model even complicated shapes by smooth curves. On the other hand we have to keep the curve scheme simple enough to construct vector fields with exactly these curves as tangent curves.

We have chosen a piecewise G^1 (tangent direction) continuous quadratic Bézier spline curve approach. As we will see later in section 4.4.2, this class of curves can nicely be incorporated into a piecewise linear vector field. The Bézier polygons of the curves are the control polygons of the separatrices. Figure 4.38 illustrates an example. Note that these control polygons must not intersect any of the control polygons of the critical points.

To illustrate the complete construction of the topological control polygon, we construct the topological skeleton of a certain vector field of higher order topology as shown in figure 4.39. First we construct the control polygons of 3 higher order critical points as shown in figure 4.39a. As we can see in this figure, the upper critical point consists of 4 hyperbolic sectors, the critical point in the middle consists of 5 hyperbolic and one elliptic sectors, while the lower critical

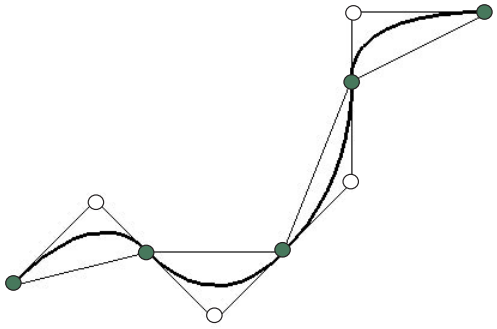


Figure 4.38: Separatrix as a piecewise G^1 continuous Bézier spline curve.

point consists of 4 hyperbolic and one parabolic sectors. In the next step, three more first order critical points of index +1 are constructed as shown in figure 4.39b. Then the separatrices are designed as piecewise G^1 quadratic Bézier spline curves, as shown in figure 4.39c. The result is a complete topological skeleton as shown in figure 4.39d.

The example above also gives an answer to the question of what size the control polygons for the critical points should have: they should be as large as possible, but sufficiently small not to intersect each other and to leave enough space to construct the separatrices in an appropriate resolution.

4.4.2 Constructing a vector field from a topological skeleton

In this section we describe how to convert the topological skeleton described in section 4.4.1 into a vector field of exactly the specified topology.

The vector field we obtain will be a *piecewise linear vector field*. To apply this class of vector fields, two problems have to be solved:

1. How to describe higher order critical points using piecewise linear vector fields?

2. How to describe piecewise quadratic separatrices using piecewise linear vector fields?

Problem 1: To solve this problem, we adapt the main idea of [194]. Given the closed polygon $(\mathbf{p}_0, ..., \mathbf{p}_{n-1})$ and the the critical point $\mathbf{p}$ inside this polygon, we construct the following initial triangulation: $\mathbf{p}$ is assigned with the zero vector $\mathbf{v} = (0,0)^T$; $\mathbf{p}_i$ is assigned with the vector $\mathbf{v}_i = \lambda_i (\mathbf{p}_i - \mathbf{p})$ with $\lambda_i \neq 0$ for $i = 0, ..., n-1$. The sign of λ_i depends on the inflow/outflow behavior of the separatrix $(\mathbf{p}, \mathbf{p}_i)$. A positive λ_i gives an outflow separatrix while a negative λ_i gives an inflow separatrix. The magnitudes of λ_i can be freely chosen at this stage of the modeling process. To avoid numerical problems, it

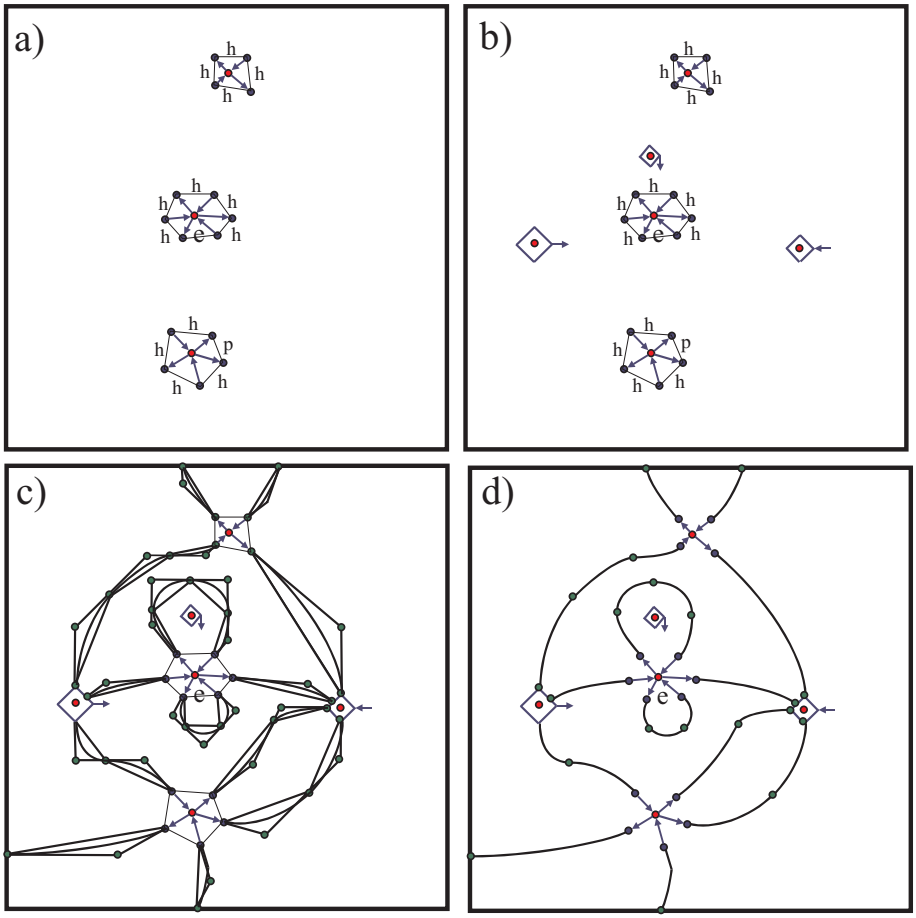


Figure 4.39: Constructing a topological skeleton: a) design control polygons of general critical points; b) design control polygons of first order critical points of index +1; c) design control polygons of separatrices; d) the final topological skeleton.

is recommended that all λ_i have approximately the same magnitude. For the following applications we have chosen $\lambda_i = \pm 1$. Then the initial triangulation to describe the critical point is given by the triangles $(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{(i+1) \bmod n})$ with the assigned vectors $(\mathbf{v}, \mathbf{v}_i, \mathbf{v}_{(i+1) \bmod n})$ for $i = 0, ..., n-1$. Figure 4.40a illustrates this initial triangulation for the example shown in figure 4.37a.

If a sector $(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{(i+1) \bmod n})$ describes a hyperbolic sector, a linear interpolation of the vectors $(\mathbf{v}, \mathbf{v}_i, \mathbf{v}_{(i+1) \bmod n})$ inside the triangle $(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{(i+1) \bmod n})$ is applied. Figure 4.40b illustrates this for all hyperbolic sectors of the example. Also a linear interpolation is applied for all parabolic sectors, as shown in figure 4.40c.

If the sector $(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{(i+1) \bmod n})$ describes an elliptic sector, the triangle $(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{(i+1) \bmod n})$ has to be refined by inserting an auxiliary point $\mathbf{q}_i$ with an assigned auxiliary vector $\mathbf{w}_i$, and considering the two new triangles $(\mathbf{p}, \mathbf{p}_i, \mathbf{q}_i)$ and $(\mathbf{p}, \mathbf{q}_i, \mathbf{p}_{(i+1) \bmod n})$. Describing $\mathbf{q}_i$ and $\mathbf{w}_i$ by

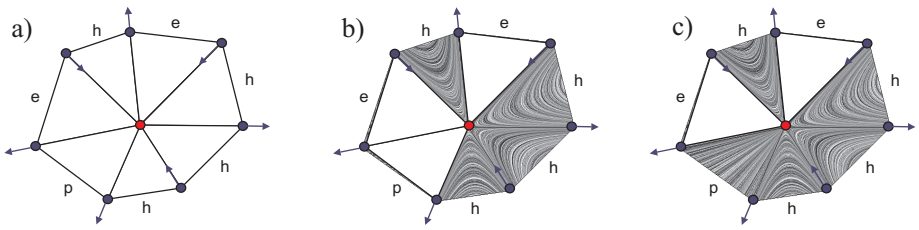


Figure 4.40: a) initial triangulation for the example shown in figure 4.37a; b) linear interpolation in all hyperbolic sectors; c) linear interpolation in all parabolic sectors.

$$\begin{aligned} \mathbf{q}_i &= \alpha\,\mathbf{p} + \beta\,\mathbf{p}_i + \gamma\,\mathbf{p}_{(i+1) \bmod n} \\ \mathbf{w}_i &= -\left(\delta\,\mathbf{v}_i + \varepsilon\,\mathbf{v}_{(i+1) \bmod n}\right) \end{aligned}$$

with $\alpha + \beta + \gamma = 1$ and $\beta, \gamma, \delta, \varepsilon > 0$, we formulate the following constraints for $\mathbf{q}_i, \mathbf{w}_i$:

- The triangles $(\mathbf{p}, \mathbf{p}_i, \mathbf{q}_i)$ and $(\mathbf{p}, \mathbf{q}_i, \mathbf{p}_{(i+1) \bmod n})$ must not create new separatrices by applying a linear interpolation inside them, i.e.

$$\begin{aligned} \det(\,(1-t)\,\mathbf{p}_i + t\,\mathbf{q}_i - \mathbf{p}\;,\;(1-t)\,\mathbf{v}_i + t\,\mathbf{w}_i\,) \neq 0 \\ \det(\,(1-t)\,\mathbf{q}_i + t\,\mathbf{p}_{(i+1) \bmod n} - \mathbf{p}\;,\;(1-t)\,\mathbf{w}_i + t\,\mathbf{v}_{(i+1) \bmod n}\,) \neq 0 \end{aligned}$$

  for $0 < t < 1$.

- The piecewise linear vector field over the two triangles $(\mathbf{p}, \mathbf{p}_i, \mathbf{q}_i)$ and $(\mathbf{p}, \mathbf{q}_i, \mathbf{p}_{(i+1) \bmod n})$ is curvature continuous (see (4.21))

- $\delta^2 + \varepsilon^2 \rightarrow \min$.

These three conditions form a minimization problem with boundary conditions. It has a unique solution for $\alpha, \beta, \gamma, \delta, \varepsilon$ and therefore for $\mathbf{q}_i$ and $\mathbf{w}_i$:

$$\mathbf{q}_i = \frac{1}{2}\left(\mathbf{p}_i + \mathbf{p}_{(i+1) \bmod n}\right)$$

$$\mathbf{w}_i = \begin{cases} \frac{1}{2}\frac{\lambda_i}{\lambda_{(i+1) \bmod n}}\left(\mathbf{v}_i + \mathbf{v}_{(i+1) \bmod n}\right) & \text{for} \quad -\frac{\lambda_i}{\lambda_{(i+1) \bmod n}} \geq 1 \\ \frac{1}{2}\frac{\lambda_{(i+1) \bmod n}}{\lambda_i}\left(\mathbf{v}_i + \mathbf{v}_{(i+1) \bmod n}\right) & \text{for} \quad -\frac{\lambda_i}{\lambda_{(i+1) \bmod n}} < 1 \end{cases}$$

where $\lambda_i, \lambda_{(i+1) \bmod n}$ are obtained from $\mathbf{v}_i = \lambda_i\,(\mathbf{p}_i - \mathbf{p})$ and $\mathbf{v}_{(i+1) \bmod n} = \lambda_{(i+1) \bmod n}\,(\mathbf{p}_{(i+1) \bmod n} - \mathbf{p})$. Figure 4.41 illustrates this.

If the critical point to be described is a first order critical point of index +1, we construct a piecewise linear vector field out of its topological skeleton shown in figure 4.37b in the following way: the vectors $\mathbf{v}_i$ assigned to the control points $\mathbf{p}_i$ $(i = 1, 2, 3)$ are computed by

$$\begin{aligned} \|\mathbf{v}_i\| &= \|\mathbf{v}_0\| \\ \text{angle}(\mathbf{v}_i, \mathbf{p}_i - \mathbf{p}) &= \text{angle}(\mathbf{v}_0, \mathbf{p}_0 - \mathbf{p}) \end{aligned}$$

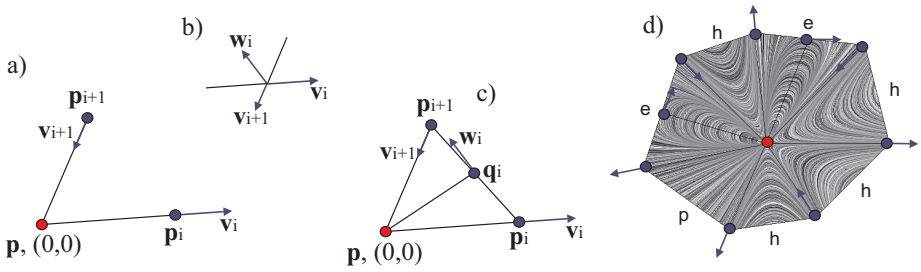


Figure 4.41: Constructing an elliptic area: a) initial sector; b) location of $\mathbf{w}_i$ relative to $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$; c) refined triangulation after inserting $\mathbf{q}_i$ with $\mathbf{w}_i$; d) application to the example in figure 4.37a.

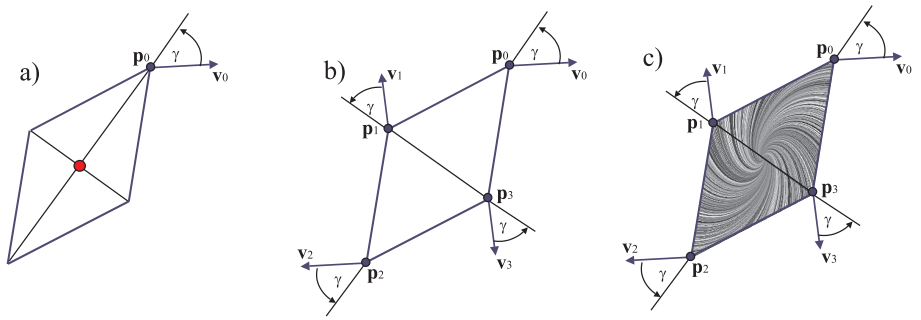


Figure 4.42: Modeling a first order critical point of index +1; a) topological skeleton; b) constructed triangulation; c) resulting piecewise linear vector field.

Then a piecewise linear interpolation over the two subtriangles $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_3)$ and $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ gives the desired first order critical point. Figure 4.42 illustrates this.

Problem 2: In [141] it has been shown that the tangent curve of a linear vector field is in general a certain exponential curve. In order to describe a separatrix (i.e. a special tangent curve) as a parabola segment, we search for a useful special configuration of the vector field where a tangent curve simplifies to a parabola. We formulate

Theorem 7 *Let a non-degenerate 2D triangle* $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ *assigned with the 2D vectors* $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$ *have the following properties:*

$$\mathbf{v}_0 = \lambda_0 (\mathbf{p}_1 - \mathbf{p}_0) \tag{4.72}$$

$$\mathbf{v}_2 = \lambda_2 (\mathbf{p}_2 - \mathbf{p}_1) \tag{4.73}$$

$$\mathbf{v}_1 = \frac{1}{2} \left(\lambda_2 (\mathbf{p}_1 - \mathbf{p}_0) + \lambda_0 (\mathbf{p}_2 - \mathbf{p}_1) \right) \tag{4.74}$$

for certain $\lambda_0, \lambda_2 > 0$. *Applying a linear interpolation of the vectors* $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$ *inside the triangle* $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$, *the following statements for the resulting linear vector field hold:*

1. *The tangent curve passing through* $\mathbf{p}_0$ *passes through* $\mathbf{p}_2$ *as well.*

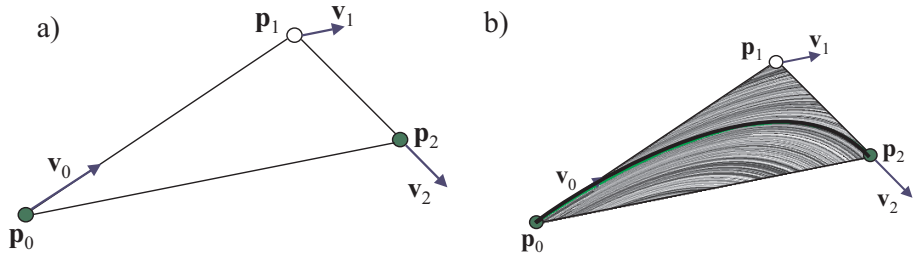


Figure 4.43: Illustration of theorem 7; a) illustration of condition (4.72) - (4.74); b) resulting vector field and parabola shaped tangent curve.

2. *The tangent curve through* $\mathbf{p}_0$ *and* $\mathbf{p}_2$ *has the same shape (but not necessarily the same parameterization) as the parabola defined by the Bézier points* $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$.

To prove theorem 7 we have to show that for every point on the parabola defined by the Bézier points $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ the tangent direction and the direction of $\mathbf{v}$ at the curve location coincide. Let

$$\mathbf{x}(t) = \sum_{i=0}^{2} B_i^2(t)\ \mathbf{p}_i$$

be the parabola where $B_i^2(t)$ are the Bernstein polynomials (see [55]). Since $\sum_{i=0}^{2} B_i^2(t) \equiv 1$, the Bernstein polynomials can be considered as the barycentric coordinates relative to the points $\mathbf{p}_i$. Thus we obtain

$$\mathbf{v}(\mathbf{x}(t)) \ = \ \sum_{i=0}^{2} B_i^2(t)\ \mathbf{v}_i. \tag{4.75}$$

Inserting (4.72) - (4.74) into (4.75) gives

$$\begin{aligned} \mathbf{v}(\mathbf{x}(t)) &= \Big((1-t)\,\lambda_0 + t\,\lambda_2\Big)\Big((1-t)\,(\mathbf{p}_1 - \mathbf{p}_0) + t\,(\mathbf{p}_2 - \mathbf{p}_1)\Big) \\ &= \frac{1}{2}\Big((1-t)\,\lambda_0 + t\,\lambda_2\Big)\,\dot{\mathbf{x}}(t) \end{aligned} \tag{4.76}$$

which proves the theorem. Figure 4.43 illustrates theorem 7.

Now theorem 7 can easily be used to construct a piecewise linear vector field which describes a G^1 piecewise quadratic separatrix of a given skeleton. Figure 4.44 illustrates this for the separatrix shown in figure 4.38.

Now we can formulate the *algorithm* to construct a piecewise linear vector field of a given topological skeleton:

1. Construct the piecewise linear vector field inside the control polygons of all general critical points.

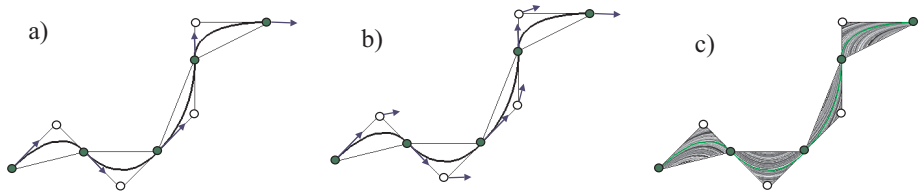


Figure 4.44: Constructing a piecewise linear vector field consisting of a piecewise G^1 quadratic separatrix; a) construct vectors at the junction points using (4.72) and (4.73); b) construct vectors at intermediate points using (4.74); c) resulting piecewise linear vector field and separatrix.

2. Construct the piecewise linear vector field inside the control polygons of all first order critical points of index +1.

3. Construct the piecewise linear vector field describing all separatrices.

4. Specify the flow direction vector at additional interesting points and in the corner vertices of the domain.

5. Triangulate the remaining parts of the vector field using only the already defined vertices, and apply a piecewise linear interpolation on this triangulation. To do this, we used a Delaunay triangulation. In this step no further critical points have to be obtained. For a correct and complete topological skeleton the appearance of new critical points can be prevented by interactively introducing new auxiliary vertices and their assigned vectors in the still uncovered areas of the vector field.

The result of this algorithm is a piecewise linear vector field of exactly the same topology as specified in the topological skeleton. Figure 4.45 illustrates this algorithm by constructing a piecewise linear vector field to the topological skeleton introduced in figure 4.39. This vector field with 3 general critical points and 3 first order critical points is constructed as piecewise linear vector field consisting of 79 vertices and 138 triangles.

4.4.3 Simplification and compression of vector fields

In recent years the simplification and compression of vector fields have become a popular research topic in scientific visualization. Flow data sets (and the vector fields derived from them) are continuously growing, so that their simplification and compression becomes significant in the visualization process.

Simplification of a vector field means finding a new vector field which keeps the most important properties but skips the less important details. In [43] the topology of a 2D vector field is simplified using area metrics. There the input vector field has to consist of a simple topology (i.e. only first order critical points and separatrices of type 1 (see section 4.1.2)). The output vector field is a simple vector field with a reduced number of critical points. In [194] a vector field of a rich simple topology (i.e. with a large number of critical points) is simplified by replacing clusters of first order critical points by a higher order

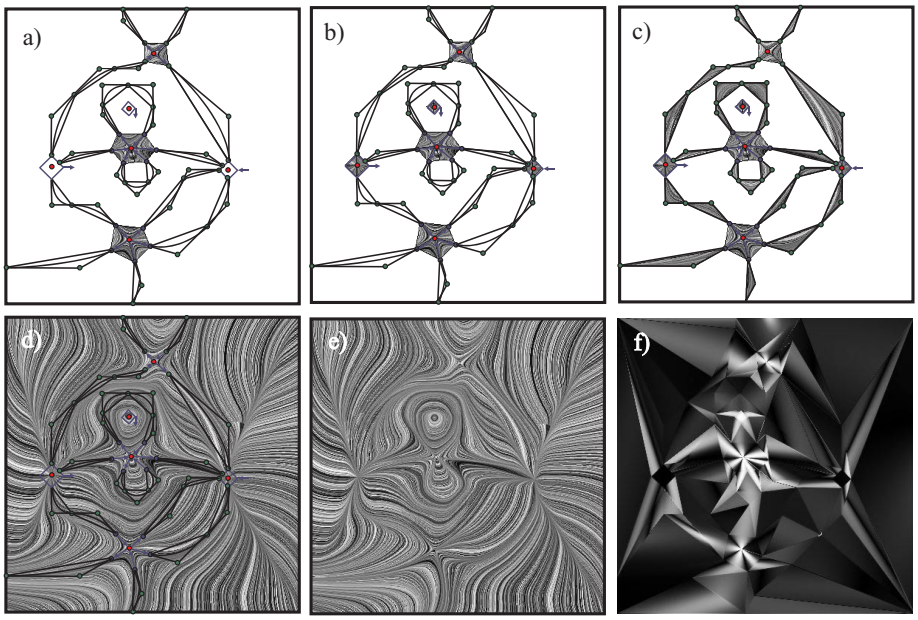


Figure 4.45: Constructing the piecewise linear vector field for the topological skeleton of figure 4.39; a) construct piecewise linear vector field for general critical points; b) construct piecewise linear vector field for first order critical points of index +1; c) construct piecewise linear vector field for separatrices; d) Delaunay triangulate remaining parts and apply piecewise linear interpolation; e) final vector field consists of 79 vertices and 138 triangles; f) curvature plot of e).

critical point. Both approaches in [43] and [194] can be used for multiresolution representations of vector fields. Note that a topological simplification of a vector field does not necessarily cause a compression. For instance the simplified vector fields in [194] may even be bigger than the originals.

To compress a vector field means finding a new vector field of smaller size which keeps important properties of the original one. Compression algorithms are well researched in the context of images and scalar fields. So the first approaches to compressing vector fields focused on adapting compression techniques of other data classes. In this way [86], [183] and [67] construct hierarchies of compressed vector fields.

Although the topology of a vector field is an important feature, compression techniques coming from these areas are usually not topology-preserving. In fact, the compressed vector fields may have significant different topologies than the originals.

[128] introduces the first approach to compress a vector field while preserving its topology by applying a bottom-up clustering similar to [183]. For vector fields with a poor topology (i.e. only a few critical points and separatrices), significant compression rates can be achieved. Figure 4.46a shows the gradient field from one of Franke's data sets

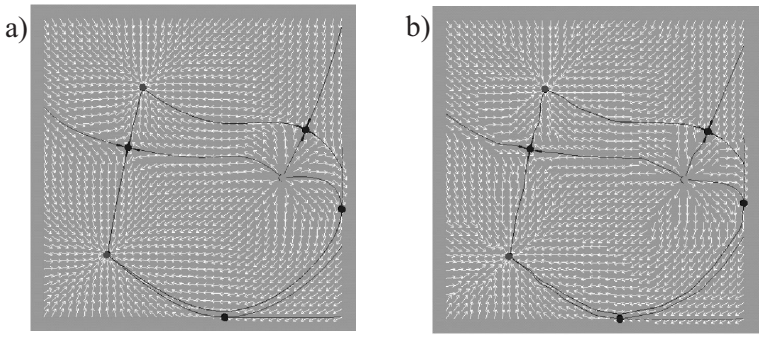


Figure 4.46: a) test data set (4.77) on a regular 38 x 38 grid; b) compressed version of the same topology, compression ratio 90%; (images from [128]).

$$s(x,y) = \frac{3}{4}\,e^{-\frac{(9\,x-2)^2+(9\,y-2)^2}{4}} + \frac{1}{2}\,e^{-\frac{(9\,x-7)^2+(9\,y-3)^2}{4}} - \frac{1}{5}\,e^{-(9\,x-4)^2+(9\,y-7)^2} + \frac{3}{4}\,e^{-\frac{(9\,x+1)^2}{49}-\frac{(9\,y+1)^2}{10}} \tag{4.77}$$

in the domain $[0,1]^2$, sampled by a 38 x 38 grid. Figure 4.46b shows the compressed vector field of the same topology; the achieved compression ratio there was 90%.

[49] obtains a simplified representation of a scalar field (and thus of its derived gradient vector field) by extracting the Morse complexes and reconstructing them by piecewise linear scalar fields.

In the remaining part of this section we want to introduce a new topology-preserving compressing technique for 2D vector fields of a simple topology. This technique is a direct application of the vector field design approach discussed in this section 4.4.

Given a 2D vector field of a simple topology, its topological skeleton can be extracted automatically (see [87]). The result of this extraction process is the exact classification of all first order critical points, and numerically integrated separatrices. To convert this into a set of control polygons described in section 4.4.1, the numerically integrated separatrices have to be replaced by piecewise G^1 quadratic curve segments. This may be done interactively or automatically by placing an appropriate number of sample points on the numerically integrated curve and declaring them as junction points of the parabola segments.

After the topological skeleton is constructed as shown in section 4.4.1, a vector field of exactly this topology can be constructed following the approach of section 4.4.2. The new vector field obtained this way has exactly the same topology as the original one. It turns out that for vector fields with rather poor topology, the new vector field is a compressed version of the original one.

To demonstrate this compression algorithm we consider the vector field (4.77) as shown in figure 4.46a. The Integrate&Draw version and the curvature plot of this vector field are shown in figure 4.47a and 4.47b. After automatically extracting the topology, the first order critical points are modeled as shown in figure 4.47c. Figure 4.47d shows how the separatrices are modeled in the piecewise linear vector field. (Since some of the separatrices in this vector field

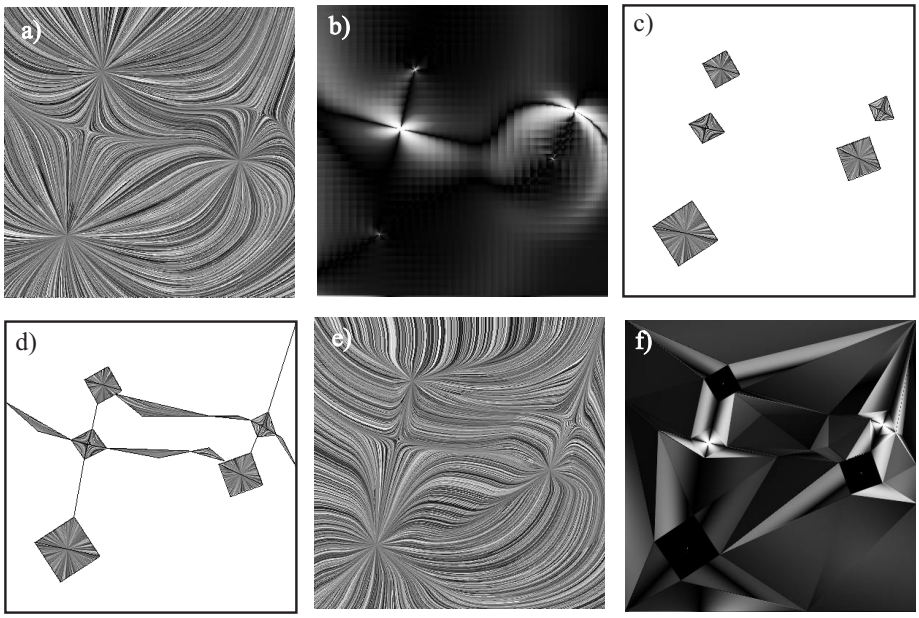


Figure 4.47: a) vector field (4.77) on a 38 x 38 regular grid: 1444 grid points; b) curvature plot of a); c) remodeling the critical points as piecewise linear vector field; d) remodeling the separatrices as piecewise linear vector field; e) complete remodeled piecewise linear vector field consists of 40 vertices and 68 triangles: compression ratio 95%; f) curvature plot of e).

are approximately straight lines, the triangles representing them collapse to line segments.) Figure 4.47e shows the final piecewise linear vector field which consists of 40 vertices and 68 triangles. Thus the compression ratio to the original vector field is 95%; the visual differences between figure 4.47e and the original in figure 4.47a are only marginal. Figure 4.47f shows the curvature plot of figure 4.47e.

Another example of a vector field with a richer topology is shown in figure 4.48. Figure 4.48a shows a fragment of the data set already shown in figure 4.27. It consists of a regular 34 x 34 grid and has therefore 1056 grid points. Figure 4.48b shows it curvature plot. Figure 4.48c shows the construction of the critical points as a piecewise linear vector field. Figure 4.48d shows the construction of the separatrices. Figure 4.48e shows the complete remodeled piecewise linear vector field consisting of 124 vertices and 226 triangles. This gives a compression ratio of 79% in comparison to figure 4.48a. The curvature plot in figure 4.48f both reveals the underlying triangulation and shows the effect of compression in comparison to figure 4.48b.

The new topology-preserving compression algorithm introduced here gives high compression rates if the original vector field has a rather poor topology. This means only a small amount of information is necessary to describe the topology. In fact, the compression ratios are higher than the only comparable approach in [128]. However, for a rich topology a higher amount of information is necessary to describe it. Hence the compression ratios are lower.

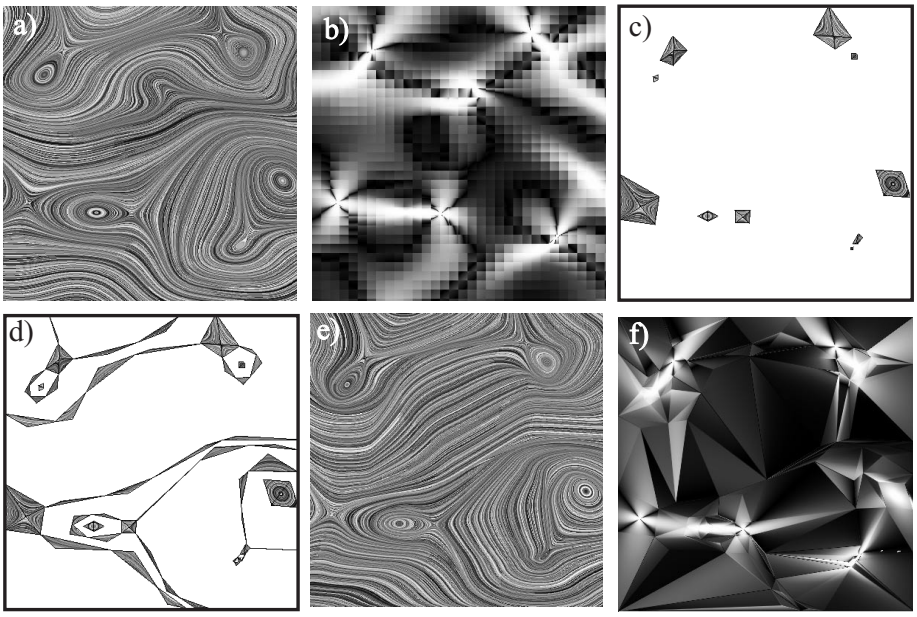


Figure 4.48: a) fragment of the vector field shown in figure 4.27 on a 34 x 34 regular grid: 1056 grid points; b) curvature plot of a); c) remodeling the critical points as piecewise linear vector field; d) remodeling the separatrices as piecewise linear vector field; e) complete remodeled piecewise linear vector field consists of 124 vertices and 226 triangles: compression ratio 79%; f) curvature plot of e).

A multiresolution version of the new compression algorithm is possible by successively inserting auxiliary vertices and their assigned vectors before the step of triangulating the remaining areas.

Chapter 5

CAGD for the Visualization of Multiparameter Data

Multiparameter data is another data class on which scientific visualization focuses. In recent years a large number of visualization techniques for multiparameter data have been developed which focus on different data characteristics and aims of the analysis.

Using the data classification of [23], the data we consider here can be written as E_n^{mS} with $m \geq 2$. Furthermore it is assumed that a certain grid in the n-dimensional domain is present. This means that at each point of an n-dimensional grid m scalar values are measured or computed[1]. We call this kind of data m-variate data on an n-dimensional grid. The n variables defining the underlying grid are also called *independent variables* while the m variables at the grid points are called *dependent variables*.

A commonly used model to describe multiparameter data is a table with $n + m$ columns, one column for each (dependent or independent) variable. In such a table each column has to be annotated to state if it describes either an independent or a dependent variable. A row in the table defines an *observation case* O which is simply an $(m + n)$-tuple of scalar values realizing the $(m + n)$ variables. Figure 5.1a gives an illustration.

We can see that a description of a data set as a table gives a priori an equal treatment of independent and dependent variables. In fact, independent and dependent variables are only distinguished by the annotations to the columns, not by the table itself.

Given a multiparameter data set, there are two general strategies to find a visualization. The first strategy is to abstract from dependent and independent variables and treat them equally for the visualization. The second strategy is to distinguish carefully between independent and dependent variables in the visual representation. The independent variables which usually describe the spatial or temporal context of the data are mapped to adequate attributes of the visualization such as position or time of appearance/disappearance.

Which strategy should be preferred depends on the particular application . For the purpose of this work (i.e. applying CAGD methods to multiparameter data) we focus on the first strategy and treat the visualization techniques which

[1] If the data set is incomplete, fewer than m scalar values may be present at a grid point.

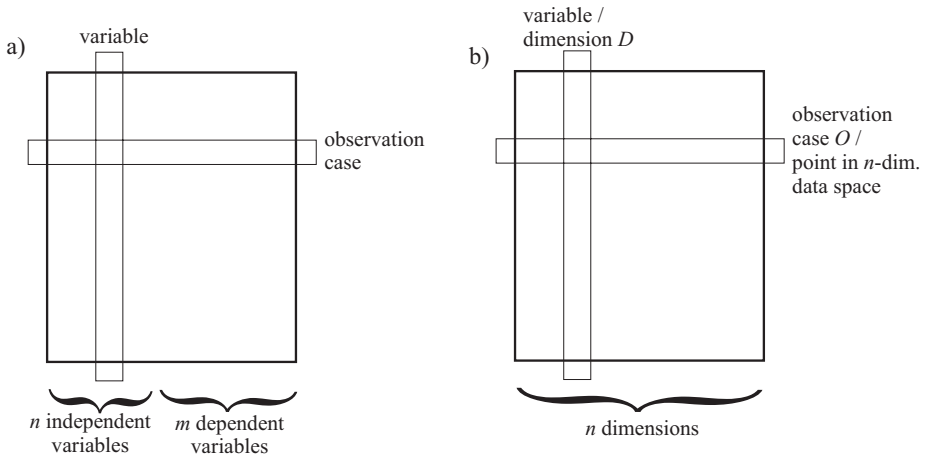


Figure 5.1: a) description of multiparameter data as a table; b) description of multidimensional data by abstracting from dependent and independent variables.

apply there. A detailed discussion of the second strategy (i.e. multiparameter data with spatial and temporal context) can be found in [167].

Considering the table description of figure 5.1a again, an abstraction from independent and dependent variables can simply be achieved by omitting the dependent/independent annotations of the columns. Virtually the same effect results if we set all annotations of the columns to "independent". This way the data set of the type E_n^{mS} is transformed to the type E_{m+n}^P. Data of this type E_{m+n}^P is also called *multidimensional data*.

The similar effect of abstracting from independent and dependent variables is caused by setting all annotations of the columns to "dependent". This way a data set of the type E_n^{mS} is transformed to the type E_0^{qS} where q is the number of observation cases in E_n^{mS}. Data of the type E_0^{qS} is called *multivariate data*.

For the rest of the chapter we deal with multidimensional data for which a simplified description as a table is illustrated in figure 5.1b. Each of the n columns[2] is assigned a dependent variable which now is also called *dimension* D. A row in the table is called *observation case*. It consists of an n-tuple of scalars which describe a point in the n-dimensional data space.

Figure 5.2 shows the pipeline for the visualization of multidimensional data. We recognize the three steps (filtering, mapping, rendering) of the general visualization pipeline for scientific data (see section 2.3.1). In addition, the step of data selection plays an important role for multiparameter data. In this step the data set to be visualized is reduced either by choosing certain dimensions or observation cases or by applying a data selection process.

It is the purpose of this chapter to explore the applicability of CAGD methods in the visualization of multidimensional data. Since interpolation issues do not play such an important role as for instance in flow visualization, the only part of the visualization pipeline where we see CAGD applications is the map-

[2]We use the simplified notation n instead of $m + n$ because the old m and n are not distinguished any more.

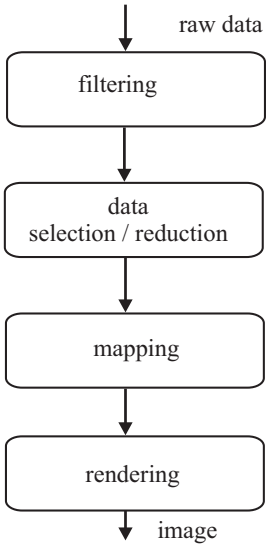


Figure 5.2: Visualization pipeline for multiparameter data.

ping part. In fact, there is a variety of mapping techniques which use curves and surfaces to represent the high amount of data which is usually present. Following [167] we can distinguish between 5 classes of visualization techniques: panel matrices, icon based techniques, line representations, pixel based techniques and hierarchical techniques[3].

Panel matrices ([210]) pick pairs of variables and visualize the data as a number of matrix plots of these pairs of variables. Representatives of this class of visualization techniques are scatter plot matrices ([32]), prosection views ([175]) and hyperslices ([199]). All these techniques have in common the fact that they use point clouds for the visualization of the 2D matrices. Hence an application of curves and surfaces is not done for panel matrices.

Pixel based techniques ([111]) map every observation case to a certain pixel and color code its value there. This way a high amount of data can be encoded onto the screen. Representatives of these techniques are grouping techniques ([110]) and recursive pattern techniques ([109]). Due to the nature of pixel based techniques, applications of curves/surfaces to them do not exist.

For the remaining classes of visualization techniques for multidimensional data (icon based techniques, line representations, hierarchical techniques), the application of curves/surfaces is possible. We treat each of these classes in one of the following sections 5.1 – 5.3 in detail.

5.1 Icon Based Techniques

The idea of icon based techniques is to create a geometric object for each observation case (i.e. each line in the table describing the multidimensional data). This object is called an *icon*; in it all values of an observation case are encoded

[3][167] introduced this distinction particularly for multivariate data. Concerning the data transformations described above, we can use them for multidimensional data as well.

in the following features:

- shape of the icon
- size of the icon
- drawing attributes (color, transparency)
- location of the icon.

Observing a number of these icons, the user may detect geometric similarities or trends in the patterns of the icons which denote inner correlations of the multiparameter data.

A variety of different icons have been developed like stick figures ([149]), color icons ([125]), Chernoff faces ([28]), data jacks ([85]), and shape coding ([15]).

Due to the fact that the human eye reacts rather sensitively to the shape of a surface, surfaces are promising candidates to build icons for multidimensional data. The ShapeVis approach in [193] was especially designed to realize this idea. Thus we treat it in detail in the next section.

5.1.1 The ShapeVis approach

The ShapeVis approach introduced in [193] builds a curve or surface shaped icon for each observation case of a multidimensional data set. These icons are placed in the 2D or 3D presentation space in such a way that observation cases with similar properties are visualized spatially close to each other. Here we consider an observation case O (i.e. a row in the table describing the multidimensional data) as an n-tuple $(c_1, ..., c_n) \in \mathbb{R}^n$ $(c_1, ..., c_n > 0)$. Interpreting O as a point in n-dimensional data space, we can formulate the task as finding a map from the n-dimensional data space to the 2D/3D presentation space.

To find an appropriate location of an observation case in the 2D/3D presentation space, the application of a common spring model is a popular approach ([145], [16], [91]). For such a common spring model, every dimension of the n-dimensional data set is related to a fixed *dimension point* $\mathbf{d}_i \in \mathbb{R}^2(\mathbb{R}^3), (i = 1, ..., n)$. An observation case is related to a point $\mathbf{p}$ in presentation space. To find the location of $\mathbf{p}$, we consider n springs - from each dimension point $\mathbf{d}_i$ to $\mathbf{p}$. The stiffness of the springs are set to the values $c_1, ..., c_n$. Then the location $\mathbf{p}$ is searched where the spring model is in balance. For fixed $\mathbf{d}_i$ we can compute this location explicitly:

$$\mathbf{p} = \frac{\sum_{i=1}^{n} c_i \cdot \mathbf{d}_i}{\sum_{i=1}^{n} c_i}. \tag{5.1}$$

Figure 5.3a gives an illustration for $n = 4$.

Using this common spring model for placing observation cases in the presentation space has the following advantages, so that the technique is useful and often applied:

- The location of $\mathbf{p}$ gives spatially intuitive information about the observation case O: the more O is related to the i-th dimension (i.e., the bigger c_i is), the closer moves $\mathbf{p}$ towards $\mathbf{d}_i$.
- Since every observation case is visualized only as a point, a large number of observation cases can be visualized. The result is a point cloud.

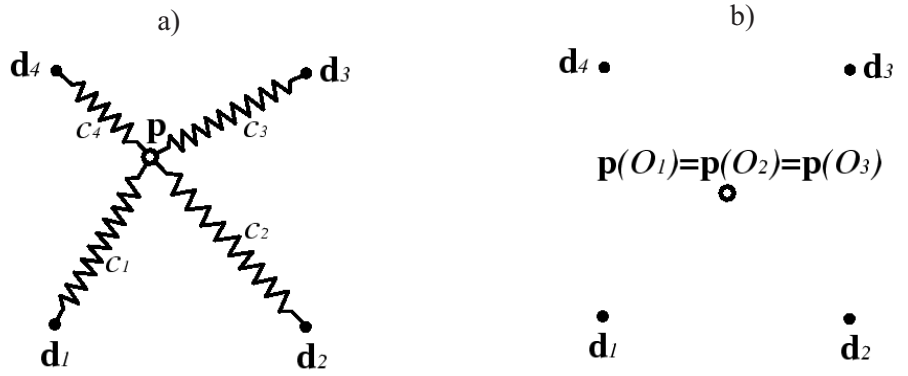


Figure 5.3: a) Visualizing the observation case $O = (c_1, ..., c_4)$ of a four dimensional data set. The dimension points $\mathbf{d}_1, ..., \mathbf{d}_4$ are fixed, $\mathbf{p}$ moves to the position of balance of the spring system. b) Limitations of the common spring model: the observation cases $O_1 = (1, 2, 1, 2)$ and $O_2 = (2, 1, 2, 1)$ collapse to one point (ambiguity). So do the observation cases O_1 and $O_3 = (2, 4, 2, 4)$ (insensitivity to coordinate scalings).

- Observation cases with similar properties are spatially close in the visualization. We can search for clusters in the visualized point cloud.

Unfortunately there are limitations to this common spring model as well:

1. Ambiguity: observation cases with different coordinates $c_1, ..., c_n$ may collapse to one point in the visualization. This means that similar observation cases are spatially close in the visualization, but we do not know whether or not observation cases which are spatially close in the visualization are similar.

2. Insensitivity to coordinate scalings: the observation cases $(c_1, ..., c_n)$ and $(k \cdot c_1, ..., k \cdot c_n)$ with $k > 0$ cannot be distinguished in the visualization because they are mapped to the same point.

Figure 5.3b illustrates these limitations.

To decrease the impact of the drawbacks 1 and 2, research has been done on finding appropriate locations for the dimension points $\mathbf{d}_i$. In [145] and [16] the points $\mathbf{d}_i$ can be moved interactively. [74] solves a mass-spring system numerically for finding suitable locations of the $\mathbf{d}_i$. All these approaches can limit the drawbacks 1 and 2 but do not solve them.

The ShapeVis approach which we describe here is able to solve drawbacks 1 and 2. The main idea is to assign an observation case not only with a point but with a small closed free-form curve/surface. The location of the curve/surface gives spatial information similar to the location of the point $\mathbf{p}$ in the common spring model using (5.1). The additional size and shape information of the curve/surface gives more intuitive knowledge about the observation case and solves drawbacks 1 and 2. To define such a curve/surface, the common spring model (5.1) has to be extended to an enhanced spring model.

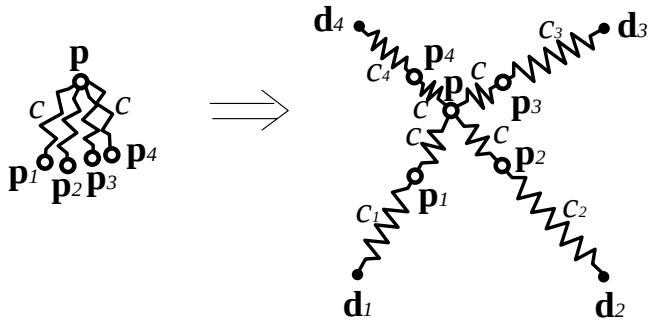


Figure 5.4: The enhanced spring model for an observation case $O = (c_1, ..., c_4)$ of a four dimensional data set. The observation case is described by the points $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_4$. The constant c is considered to be bigger than $c_1, ..., c_4$.

5.1.1.1 The enhanced spring model

As in the common spring model, we place a fixed point $\mathbf{d}_i \in \mathbb{R}^2(\mathbb{R}^3)$ for every dimension of the data set. For a given observation case $O = (c_1, ..., c_n)$, we consider a point $\mathbf{p}$ in $\mathbb{R}^2(\mathbb{R}^3)$. We attach n springs with the constant stiffness $c > 0$ to $\mathbf{p}$. The other ends of the springs are named $\mathbf{p}_1, ..., \mathbf{p}_n$. Now we consider n more springs - from $\mathbf{p}_i$ to $\mathbf{d}_i$ with the stiffness c_i for $i = 1, ..., n$. The points $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ are free movable while the points $\mathbf{d}_1, ..., \mathbf{d}_n$ are fixed. Then we search for the state of balance of this spring system. Figure 5.4 gives an illustration.

Applying this spring model, an observation case $O = (c_1, ..., c_n)$ is described by the $n+1$ points $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$. The constant c is considered to be at least one scale bigger than $c_1, ..., c_n$. Its influence will be discussed later on.

For given dimension points $\mathbf{d}_1, ..., \mathbf{d}_n$ and the constant c, the location of the points $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ describing the observation case $O = (c_1, ..., c_n)$ can be computed explicitly:

$$\mathbf{p} = \frac{\sum_{i=1}^{n} w_i \cdot \mathbf{d}_i}{\sum_{i=1}^{n} w_i} \tag{5.2}$$

with

$$w_i = \frac{c_i}{c + c_i} \quad \text{for } i = 1, ..., n. \tag{5.3}$$

Then we obtain for $\mathbf{p}_1, ..., \mathbf{p}_n$:

$$\mathbf{p}_i = \frac{c \cdot \mathbf{p} + c_i \cdot \mathbf{d}_i}{c + c_i} \quad \text{for } i = 1, ..., n. \tag{5.4}$$

Obviously, the locations of $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ depend on the value of the constant c. To study the impact of c, we consider the special cases of converging c to infinity and zero:

$$\lim_{c \to \infty} \mathbf{p} = \lim_{c \to \infty} \mathbf{p}_1 = ... = \lim_{c \to \infty} \mathbf{p}_n = \frac{\sum_{i=1}^{n} c_i \cdot \mathbf{d}_i}{\sum_{i=1}^{n} c_i}. \tag{5.5}$$

This means that for $c \to \infty$ the points $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ collapse to the point which describes the observation cases in the common spring model described by (5.1).

Therefore, the common spring model is contained as a special case in the enhanced spring model. For $c \to 0$ we obtain

$$\begin{aligned} \lim_{c\to 0} \mathbf{p} &= \frac{\sum_{i=1}^{n} \mathbf{d}_i}{n} \\ \lim_{c\to 0} \mathbf{p}_i &= \mathbf{d}_i \quad \text{for } i = 1, ..., n. \end{aligned} \tag{5.6}$$

For $c \to 0$, the locations of $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ do not depend on the observation case anymore. Therefore, c should be chosen rather large.

The points $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ describe an observation case $O = (c_1, ..., c_n)$ uniquely. This means that from given $\mathbf{d}_1, ..., \mathbf{d}_n, \mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ and c we can compute all c_i $(i = 1, ..., n)$. To do so we only have to solve the linear system of the equations (5.2)-(5.4) with the unknowns $c_1, ..., c_n$. This gives a unique solution if no two of the dimension points coincide and the points $\mathbf{p}, \mathbf{p}_i, \mathbf{d}_i$ are colinear for $i = 1, ..., n$.

5.1.1.2 Obtaining closed curves/surfaces

Even if the points $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ describe an observation case uniquely - for the visualization of an observation case $n + 1$ separate points are not suitable. We therefore define a closed curve/surface which gives an intuitive imagination of the location of $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$. In the following we only describe the surface case for a 3D visualization. The case of a closed curve in a 2D visualization can be obtained as a special case of the surface.

Defining a surface out of the control points $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ using usual approaches like Bézier- or B-Spline surfaces fails because these approaches depend on the order of the control point sequence. For our problem, the order of the points $\mathbf{p}_1, ..., \mathbf{p}_n$ should not influence the surface.

We define a closed surface as a deformation of a small sphere around $\mathbf{p}$:

$$\begin{aligned} \mathbf{x}(\lambda, \phi) = \mathbf{p} + f(\lambda, \phi) \cdot \begin{pmatrix} \cos\phi \cdot \cos\lambda \\ \cos\phi \cdot \sin\lambda \\ \sin\phi \end{pmatrix} \\ \left(0 \le \lambda < 2\pi \;\; ; \;\; -\frac{\pi}{2} \le \phi \le \frac{\pi}{2}\right). \end{aligned} \tag{5.7}$$

In (5.7), $\mathbf{p}$ is a 3D point and $f(\lambda, \phi)$ is a certain bivariate scalar function. For $f(\lambda, \phi) = \text{const}$, (5.7) describes a sphere around $\mathbf{p}$ in spherical coordinates. This sphere will be deformed by the function $f(\lambda, \phi)$. The special case of a 2D visualization is contained in (5.7) by setting $\phi = 0$.

To define $f(\lambda, \phi)$ we introduce the auxiliary functions

$$f_i(\lambda, \phi) = \begin{cases} \dfrac{(\mathbf{p}_i - \mathbf{p}) \cdot \begin{pmatrix} \cos\phi \cdot \cos\lambda \\ \cos\phi \cdot \sin\lambda \\ \sin\phi \end{pmatrix}}{\|\mathbf{p}_i - \mathbf{p}\|} \\ \qquad \text{if } (\mathbf{p}_i - \mathbf{p}) \cdot \begin{pmatrix} \cos\phi \cdot \cos\lambda \\ \cos\phi \cdot \sin\lambda \\ \sin\phi \end{pmatrix} > 0 \\ 0 \qquad \text{else} \end{cases} \tag{5.8}$$

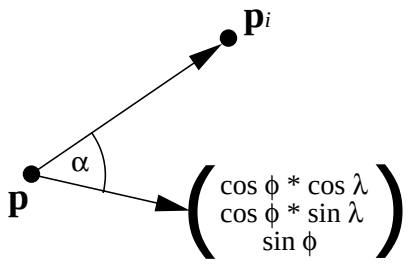


Figure 5.5: The auxiliary function $f_i(\lambda, \phi)$: $f_i(\lambda, \phi) = \cos\alpha$ if $\cos\alpha > 0$, else 0.

for $i = 1, ..., n$. Figure 5.5 illustrates (5.8).

Now we can define the deforming function $f(\lambda, \phi)$ by

$$f(\lambda, \phi) = f_0 + \sum_{i=1}^{n} \|\mathbf{p}_i - \mathbf{p}\| \cdot (f_i(\lambda, \phi))^{sh}. \tag{5.9}$$

Generally, the surface $\mathbf{x}(\lambda, \phi)$ shows deformations in all directions $\mathbf{p}_i - \mathbf{p}$. The bigger $\|\mathbf{p}_i - \mathbf{p}\|$ is, the stronger the deformations are. The constant sh determines how strongly the deformation is performed in the directions close to $\mathbf{p}_i - \mathbf{p}$. It has similarities to the shiny exponent in Phong's illumination model (see [59]): the bigger sh is, the sharper is the deformation (highlight in Phong's model). The impact of sh is illustrated in figure 5.8. The constant f_0 is for preserving continuity and parameterization regularity of the surface. In fact, for $f_0 > 0$ and sh a natural number, the surface defined by (5.7)-(5.9) is C^{sh-1}-continuous. Thus we should choose $sh > 2$ in order to obtain a C^1-continuous closed surface.

Another property is the fact that the points $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ are inside the closed surface defined by (5.7)-(5.9).

We consider a first example of a 2D visualization.

Figure 5.6a shows the visualization of the observation case $O = (1, 2, 1, 2)$. The closed curve was visualized as the boundary of a filled area. The curve gives us three kinds of information: location, size and shape. The location of the whole curve gives initial information about the object. The size gives information about how big the components of an object are generally. An object with generally high components c_i $(i = 1, ..., n)$ produces a bigger closed curve. The deformations of the curve show which dimensions the object is more related to. In our example the object is more related to $\mathbf{d}_2$ and $\mathbf{d}_4$ than to $\mathbf{d}_1$ and $\mathbf{d}_3$.

A surface defined by (5.7)-(5.9) describes an observation case uniquely. This means that from the location, size and shape of such a surface we can uniquely infer all components $c_1, ..., c_n$ of the original observation case. For showing this, suppose sh is high. Then the surface converges to the line segments $(\mathbf{p}, \mathbf{p}_1), ..., (\mathbf{p}, \mathbf{p}_n)$ (figure 5.6b illustrates). This means that from the surface we obtain the location of $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ directly. As shown above, these points give the object uniquely. Figure 5.7 illustrates this by showing the visualization of the objects from figure 5.3b.

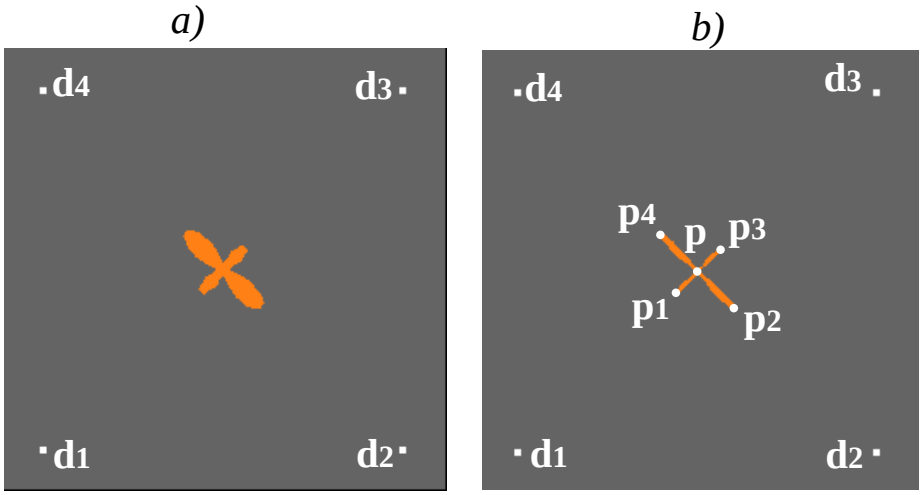


Figure 5.6: a): Visualizing the observation case $O = (1, 2, 1, 2)$ with $sh = 10$. b): The same observation case visualized with a high sh ($sh = 100$). We obtain $\mathbf{p}, \mathbf{p}_1, ..., \mathbf{p}_n$ directly from the curve.

5.1.1.3 Visualizing the closed curves/surfaces

To visualize a curve/surface related to an observation case O, we have to determine the following parameters: c, sh, f_0 and the locations of $\mathbf{d}_1, ..., \mathbf{d}_n$. All these parameters are chosen globally, i.e. they are the same for every visualized object.

The locations of the dimension points $\mathbf{d}_i$ influence the intuitiveness of the visualization but not the uniqueness of the object description. We placed the $\mathbf{d}_i$ equally distributed on a unit sphere.

The parameter c has influence on the size of the surface. A higher c leads to a smaller surface. For $c \to \infty$, the surfaces collapse to points.

The parameter sh influences how sharp the deformations are around the directions $\mathbf{p}_i - \mathbf{p}$. The extreme case $sh \to \infty$ is shown in figure 5.6b.

The parameter $f_0 > 0$ is necessary for continuity preserving of the surface. It should be chosen rather small.

The influence of the parameters c and sh is illustrated in figure 5.8 for the 2D case.

For visualizing a higher number of objects we use the following visualization scenario:

1. Get a global impression by visualizing all observation cases with a high parameter c. Since the observation cases are almost points, the visualization is the same as from the common spring model. We try to detect clusters in the visualization. Observation cases which are not spatially close (for instance observation cases in different clusters) are not similar to each other. For observation cases in a cluster we have to continue analyzing:

2. Zoom into a cluster and increase c. The points will appear as closed surfaces. If two surfaces have different shape, the observation cases are

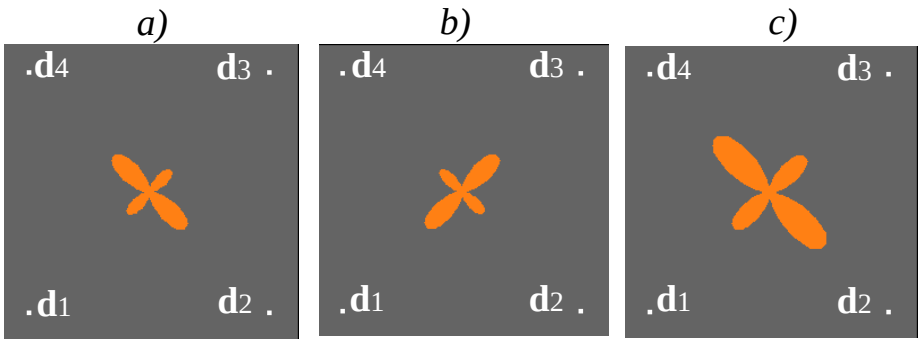


Figure 5.7: Visualizing the observation cases a): (1,2,1,2); b): (2,1,2,1); c): (2,4,2,4). In contrary to the common spring model shown in figure 5.3b, these observation cases can be well distinguished. For all observation cases in this figure we have chosen $sh = 10$.

not similar to each other. If two surfaces have a similar shape but different sizes, one observation case is the scaled image of the other one. If the surfaces are similar in shape and size, the observation cases are similar to each other.

The only problem here is how to distinguish between observation cases that are in the same location. In this case the surfaces may intersect and hide each other in part or completely. A solution to this problem is to offer the option of visualizing the surfaces transparently or in wire frame representation. In addition, the surfaces are visualized in different colors. In this way we can distinguish between surfaces in the same location as long as they are not exactly identical.

5.1.1.4 Applications of ShapeVis

We applied the ShapeVis technique to two test data sets which are publicly available on the WWW. Both data sets have been initially explored in [211].

The first data set[4] contains information about 38 automobiles including miles per gallon, weight, drive ratio, horsepower, displacement and number of cylinders. This means that we have 38 observation cases in a 6-dimensional data set. The dimension points $\mathbf{d}_1, ..., \mathbf{d}_6$ were placed in an equidistant way on the unit sphere. Figure 5.9a gives an overview over the data set in a 3D visualization. We used the parameters $c = 15, sh = 10, f_0 = 0.2$. Each of the closed surfaces was approximated by 1944 triangles. The creation of the 3D scene consisting of a number of triangles on a Silicon Graphics Indigo 2 Workstation under IRIS Explorer was computed in approximately 13 seconds. Navigation through the scene is possible at interactive time rates.

In figure 5.9a we recognize a cluster of surfaces in the upper middle part of the picture. For the cars lying in this cluster we can make the assumption that they have similar properties; but we have to check it. To do this, we zoom into this area and visualize with $c = 30$. Figure 5.9b shows the result. As we can see here, the surfaces for Buick Century Special, Dodge Aspen and AMC Concord

[4] available at http://lib.stat.cmu.edu/DASL/Stories/ClusteringCars.html

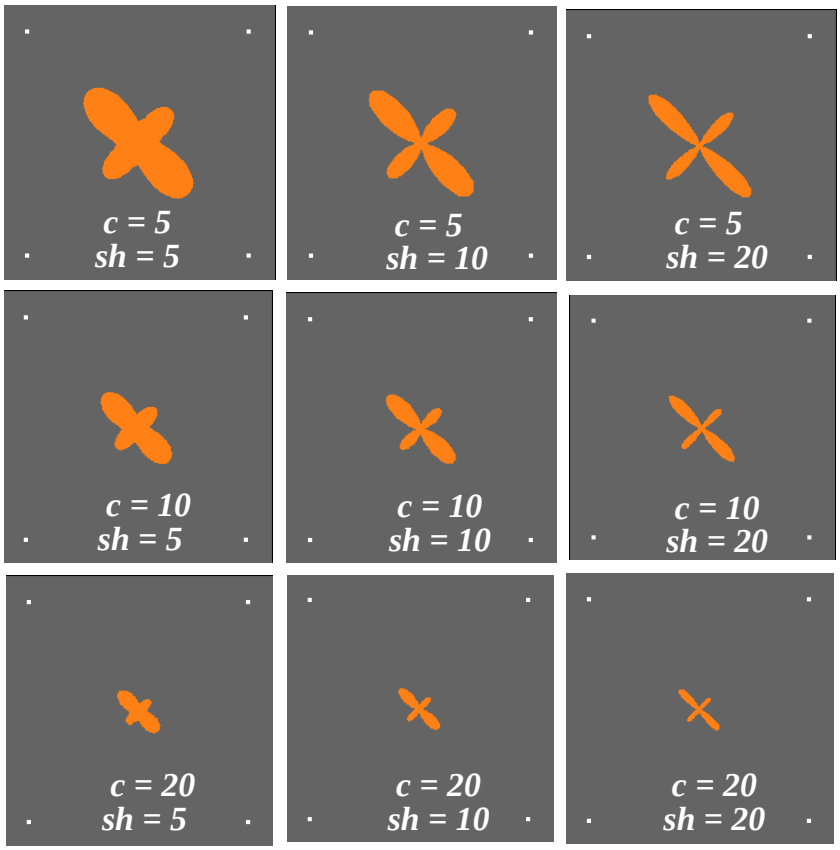


Figure 5.8: Visualizing the observation case (1,2,1,2) with different parameters *sh* and *c*.

D/L are similar in location, size and shape. These cars have, therefore, similar properties. The other 8 surfaces in figure 5.9b are also similar to each other in location, size and shape. Thus the properties of these 8 cars are similar to each other as well.

The second data set[5] measures various quality of living parameters of US cities. It is a 9 dimensional data set with 329 observation cases. The values for all dimensions are normalized to $[0, 1]$ in such a way that the higher the number the better the city.

The 9 dimensional points were placed equally distributed on a sphere. Figure 5.10a gives an overview over the data set. Most of the objects are in one big cluster, there are only a few outliers. Figure 5.10b shows the cluster in more detail. Figure 5.11 shows the magnification of the five objects lower right of figure 5.10b. As we can see here, the surfaces for Springfield, St. Louis, Baltimore and Hartford have similar shape and size. These cities therefore have similar living conditions. The shape of the surface related to Miami-Hiale differs from the 4 other cities. The living conditions in Miami-Hiale are therefore different in

[5]available at http://lib.stat.cmu.edu/datasets/places.dat

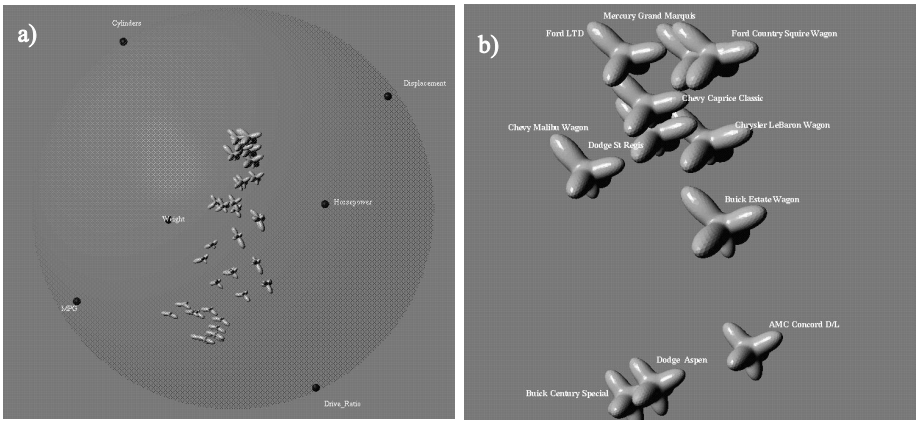


Figure 5.9: Visualization of the car data set (38 observation cases, 6 dimensions); a) overview; b) zoom into the cluster.

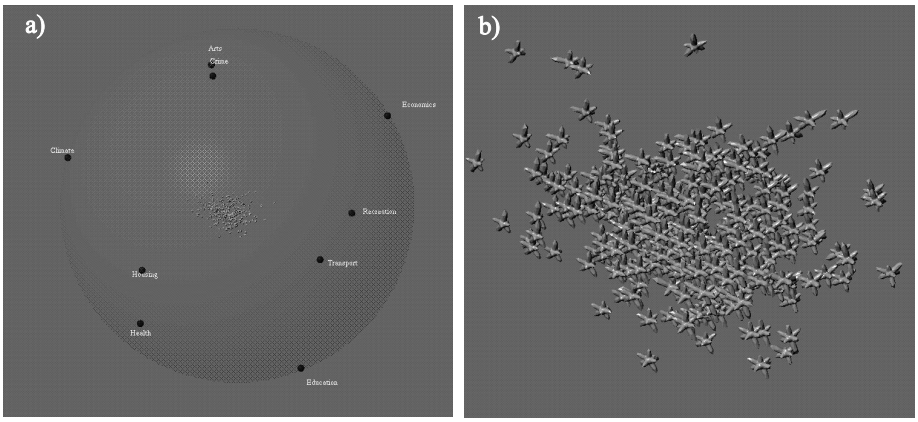


Figure 5.10: Visualization of the city data set; a) overview; b) zoom into the cluster.

comparison to the four other cities, even if the five closed surfaces are spatially close to each other.

5.1.1.5 Results and improvements of ShapeVis

We have shown that ShapeVis is able to describe an observation case uniquely as a small curve/surface. These curves/surfaces allow an intuitive visual comparison of different observation cases and thus allow the detection of inner correlations in a multidimensional data set.

However, there are limitations to the ShapeVis approach as well. For a large number of observation cases, many surfaces have to be visualized simultaneously. Since for visualization purposes each surface has to be converted to a triangular mesh, the huge number of triangles present may not allow an interactive exploration of the data. To solve this problem, [121] modifies ShapeVis by introducing simplified surfaces which consist only of n cylinders. Figure 5.11b illustrates the simplified ShapeVis approach for a 6-dimensional demographic

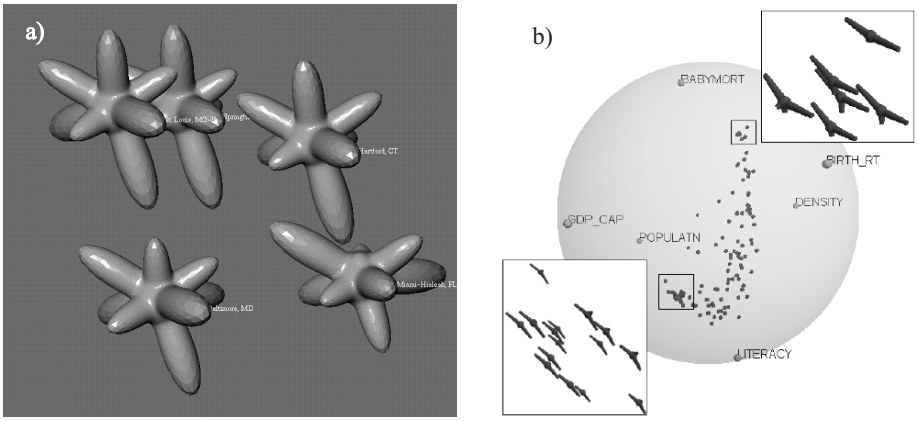


Figure 5.11: a) city data set - zoom of the 5 objects lower right in figure 5.10b; b) 6-dimensional demographic data set using simplified ShapeVis (image from [121]).

data set.

5.1.2 Designing appropriate icons

Obviously the success of a visualization using icons depends on the choice of an appropriate class of icons which represent the given multidimensional data in an optimal way. Unfortunately it is rather complicated to find an icon which represents the data adequately. Such an icon has to satisfy the following demands:

- All relevant data of an observation case should be encoded.
- There should not be redundancies in encoding the data of an observation case[6].
- Appropriate maps between the variables of the data set and the attributes of the icon have to be found. For example, a variable with a continuous domain should not be mapped to a binary attribute.
- The icons should be intuitive.

To find appropriate icons, an extensive research has been done which was mainly influenced by psychology and perception theory. Numbers of experiments have been carried out to find out to which features (shape, color,...) the human eye responds best under certain conditions. From the results of these experiments, rules and information can be obtained on how to design appropriate icons. It is beyond the scope of this work to survey these areas. Instead we refer to [201] which gives an overview on perception issues of icons. Particularly the perception of surfaces is treated in [116]

To design appropriate icons, icon editors ([58]) have been created. The ShapeVis example of section 5.1.1 has shown that for certain applications curves

[6]Sometimes redundancies are used to emphasize certain properties of the data. Especially when the icon has more attributes than the data set has dimensions, the remaining attributes should be encoded redundantly. However, in most cases an icon is desired which does not have more attributes than dimensions are present.

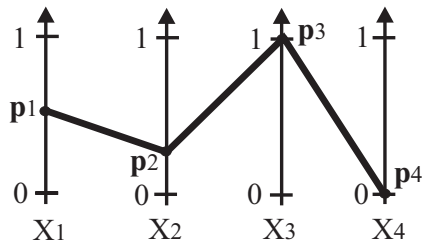


Figure 5.12: Parallel coordinates: the 4-dimensional point $(0.5, 0.25, 1.0, 0.0)$ is described by the line sequence $(\mathbf{p}_1, \mathbf{p}_2), (\mathbf{p}_2, \mathbf{p}_3), (\mathbf{p}_3, \mathbf{p}_4)$.

and surfaces are good candidates to be included into the design of icons. However, they still seem to be rarely used for icons. Instead, more simple geometric objects like lines, arcs, or boxes are preferred. Nevertheless, the ShapeVis approach has shown that a more frequent and systematic usage of curves/surfaces in the design of appropriate icons can lead to better solutions for a number of visualization problems.

5.2 Line Representations

The main idea of line representations for multidimensional data is to map a point in n-dimensional data space (i.e. an observation case) to a sequence of straight lines in 2D presentation space.

The most common representative of this class of visualization techniques are *parallel coordinates* ([99]). To visualize a point with the coordinates $(c_1, ..., c_n)$ of the n-dimensional data space, n parallel coordinate axes $X_1, ..., X_n$ are used. The coordinates $(c_1, ..., c_n)$ are mapped onto the points $\mathbf{p}_1, ..., \mathbf{p}_n$ on the corresponding coordinate axes $X_1, ..., X_n$. Then the point $(c_1, ..., c_n)$ is represented by the sequence of line segments $(\mathbf{p}_1, \mathbf{p}_2)$, $(\mathbf{p}_2, \mathbf{p}_3)$,..., $(\mathbf{p}_{n-1}, \mathbf{p}_n)$. This way we have a one-to-one correspondence between points in n-dimensional space and line segments in 2D. Figure 5.12 illustrates this for $n = 4$.

In recent years parallel coordinates became a standard tool in the visualization for multidimensional data. The reasons for this are:

- Parallel coordinates is a simple technique which can easily be applied and interpreted by a user.
- Parallel coordinates allow the visualization of a high number of dimensions.
- The number of n-dimensional points which can be visualized is rather high as well.
- correlations between adjacent dimensions (i.e. dimensions where its coordinate axes in parallel coordinates are adjacent to each other) can be detected.
- There is a strong foundation of the mathematical background of parallel coordinates.

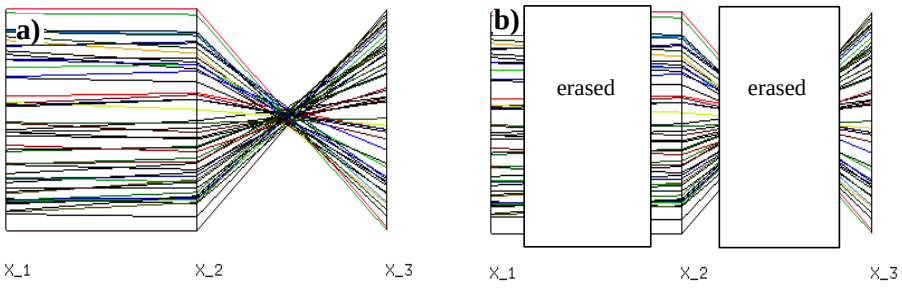


Figure 5.13: a): Parallel coordinates for two dimensions. In b), areas between the coordinate axes are erased. We can still see the correlations between the coordinate axes. Thus we may place more information into the erased space.

Nowadays there is a variety of systems and applications of parallel coordinates. An example of a visual analysis of data sets using parallel coordinates can be found in [97]. [6] describes a system specialized in parallel coordinates. In [30] parallel coordinates are used for cluster identification in higher dimensional data sets. [202] uses parallel coordinates to explore higher dynamical systems. A survey on both foundations and applications of parallel coordinates can be found in [98].

There are limitations of the parallel coordinates technique as well:

- If the number of points in n-dimensional data space is too high, the corresponding line segments tend to overly and hide each other ending up in a visual clutter; particular line segments cannot be detected.
- Correlations between non-adjacent coordinate axes can hardly be seen.

To deal with these problems, a number of approaches exist which focus on an automatic or interactive alignment and exchange of the coordinate axes ([6], [97]). [146] uses parahistograms to additionally encode the frequency of a certain line segment. Another approach to dealing with these problems is introduced in section 5.2.1.

Another representative of the class of line representations are star shaped coordinates. The main idea is similar to parallel coordinates but the coordinate axes here are not placed parallel to each other but are star shaped, originating in a common point. This way a point in n-dimensional data space is represented by a closed line sequence consisting of n straight line segments on the 2D screen.

5.2.1 Higher order parallel coordinates

In this section we introduce an extension of parallel coordinates which makes use of curves. We give this extension which is similar to [190] in order to enable parallel coordinates to detect also correlations between more than two dimensions which is one of the serious drawbacks in the common approach. To do so we use the fact that the space between adjacent coordinate axes is sometimes redundant. Figure 5.13 gives an example.

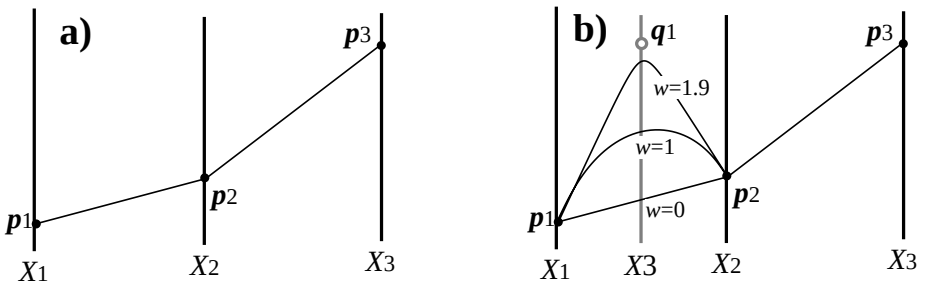


Figure 5.14: a) usual parallel coordinates: one three-dimensional object is visualized by the line segments $(\mathbf{p}_1, \mathbf{p}_2), (\mathbf{p}_2, \mathbf{p}_3)$; b) higher order parallel coordinates: the axis X_3 is additionally inserted between X_1 and X_2; $\mathbf{q}_1$ has the same height as $\mathbf{p}_3$. Then the points $\mathbf{p}_1, \mathbf{q}_1, \mathbf{p}_2$ define the curve which replaces the line segment $\mathbf{p}_1, \mathbf{p}_2$ of the usual approach. A weight w controls the influence of $\mathbf{q}_1$.

We use the redundant space between two adjacent axes to insert more information. To do so, we replace the line segments between adjacent axes with free-form curves. The reason for that is the same as the reason for applying curves and surfaces in other areas of visualization: curves have the advantage that the human eye reacts rather sensitively to small changes in their shapes, thus they are promising candidates to encode more information in a small area. The line segments used in the normal parallel coordinate approach can be interpreted as polynomial curves of order one. This gives reason to name the approach of replacing the line segments with curves "higher order parallel coordinates".

To define the curves, we place a number of additional coordinate axes from $\{X_1, ..., X_N\}$ between the adjacent axes X_k, X_{k+1}. Then we use the points $\mathbf{p}_k, \mathbf{p}_{k+1}$ and the corresponding points on the newly inserted additional axes as control points of the curve. The shape of the curve is controlled by a weight w which determines the influence of the inner control points. Figure 5.14 shows an example.

5.2.1.1 The curve scheme

Considering figure 5.14 again, we want to specify the impact of the weight parameter w which controls the influence of the additional control point $\mathbf{q}_1$. As a first condition, for $w = 0$ we want the curve to be degenerate into the line segment $(\mathbf{p}_1, \mathbf{p}_2)$. This makes sure that the usual parallel coordinates are a special case of higher order parallel coordinates. As a second condition we demand that if w is increasing to a maximal amount, the curve is going to converge to the line segments $(\mathbf{p}_1, \mathbf{q}_1), (\mathbf{q}_1, \mathbf{p}_2)$. In this case the influence of $\mathbf{q}_1$ to the curve shape is maximal.

At first glance the usage of rational Bézier- or B-spline curves seems to be a promising candidate for higher order parallel coordinates. Unfortunately, rational curve concepts fail the second condition (w becomes maximal) if we have more than two additional axes between two adjacent axes in the usual parallel coordinate approach. So the curve scheme we use here has to be more involved than a simple rational curve approach.

In the following we call the axes of the usual parallel coordinates approach

main axes; the axes placed between the main axes are called *additional axes*. Given are two adjacent main axes X_k, X_{k+1} with the points $\mathbf{p}_k, \mathbf{p}_{k+1}$ for a particular observation case. Between X_k and X_{k+1} we place n_k additional axes $Y_1, ..., Y_{n_k}$ with $\{Y_1, ..., Y_{n_k}\} \subseteq \{X_1, ..., X_n\}$. They give the data points $\mathbf{q}_1, ..., \mathbf{q}_{n_k}$ for a particular observation case. Then we define a curve out of $\mathbf{p}_k, \mathbf{q}_1, ..., \mathbf{q}_{n_k}, \mathbf{p}_{k+1}$ in the following way: we use a piecewise cubic B-spline approach with the de Boor points $\mathbf{d}_0, ..., \mathbf{d}_{2n_k+3}$ over the knot sequence $t_0, ..., t_{2n_k+7}$ (see [55]):

$$
\begin{aligned}
t_0 &= ... = t_3 = 0 \\
t_{2i+2} &= \begin{cases} i - \frac{1}{4} & \text{if} \quad 0 \le w < 1 \\ i + \frac{w-2}{4} & \text{if} \quad 1 \le w \le 2 \end{cases} \qquad \text{for} \quad i = 1, ..., n_k \\
t_{2i+3} &= \begin{cases} i + \frac{1}{4} & \text{if} \quad 0 \le w < 1 \\ i + \frac{w-1}{4} & \text{if} \quad 1 \le w \le 2 \end{cases} \qquad \text{for} \quad i = 1, ..., n_k \\
t_{2\,n_k+4} &= ... = t_{2\,n_k+7} = n_k + 1.
\end{aligned} \tag{5.10}
$$

The de Boor point $\mathbf{d}_i$ is computed as a linear combination of certain auxiliary points $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ for $i = 0, ..., 2n_k + 3$ which are defined as:

$$
\mathbf{a}_i = \frac{2\,n_k + 3 - i}{2\,n_k + 3}\,\mathbf{p}_k + \frac{i}{2\,n_k + 3}\,\mathbf{p}_{k+1} \qquad \text{for} \qquad i = 0, ..., 2\,n_k + 3
$$

$$
\begin{aligned}
\mathbf{b}_0 &= \mathbf{p}_k \quad , \quad \mathbf{b}_1 = \frac{3}{4}\,\mathbf{p}_k + \frac{1}{4}\,\mathbf{q}_1 \quad , \quad \mathbf{b}_2 = \frac{1}{4}\,\mathbf{p}_k + \frac{3}{4}\,\mathbf{q}_1 \\
\mathbf{b}_{2i+1} &= \frac{3}{4}\,\mathbf{q}_i + \frac{1}{4}\,\mathbf{q}_{i+1} \qquad \text{for} \qquad i = 1, ..., n_k - 1 \\
\mathbf{b}_{2i+2} &= \frac{1}{4}\,\mathbf{q}_i + \frac{3}{4}\,\mathbf{q}_{i+1} \qquad \text{for} \qquad i = 1, ..., n_k - 1 \\
\mathbf{b}_{2\,n_k+1} &= \frac{3}{4}\,\mathbf{q}_{n_k} + \frac{1}{4}\,\mathbf{p}_{k+1} \;,\; \mathbf{b}_{2\,n_k+2} = \frac{1}{4}\,\mathbf{q}_{n_k} + \frac{3}{4}\,\mathbf{p}_{k+1} \;,\; \mathbf{b}_{2\,n_k+3} = \mathbf{p}_{k+1}
\end{aligned} \tag{5.11}
$$

$$
\begin{aligned}
\mathbf{c}_0 &= \mathbf{p}_k \quad , \quad \mathbf{c}_1 = \mathbf{p}_k \\
\mathbf{c}_{2i} &= \mathbf{q}_i \quad , \quad \mathbf{c}_{2i+1} = \mathbf{q}_i \qquad \text{for} \qquad i = 1, ..., n_k \\
\mathbf{c}_{2\,n_k+2} &= \mathbf{p}_{k+1} \quad , \quad \mathbf{c}_{2\,n_k+3} = \mathbf{p}_{k+1}.
\end{aligned}
$$

Then the de Boor points are defined as

$$
\mathbf{d}_i = \begin{cases} (1-w)\,\mathbf{a}_i + w\,\mathbf{b}_i & \text{for} \quad 0 \le w < 1 \\ (2-w)\,\mathbf{b}_i + (w-1)\,\mathbf{c}_i & \text{for} \quad 1 \le w \le 2 \end{cases} \tag{5.12}
$$

Figure 5.15 illustrates this for $n_k = 2$.

The weight parameter w which ranges between 0 and 2 was introduced to control the influence of the additional axes $Y_1, ..., Y_{n_k}$ (and the corresponding

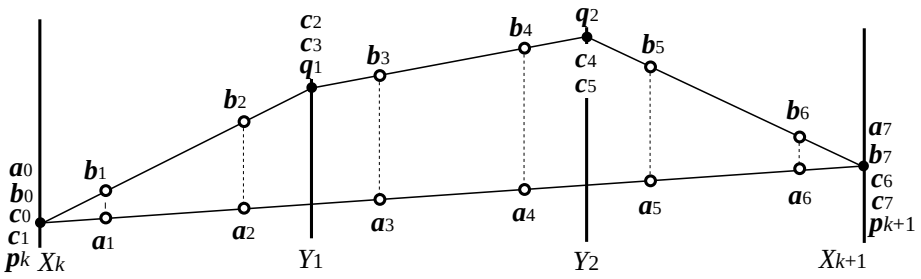


Figure 5.15: Two additional axes Y_1, Y_2 are placed between the main axes X_k and X_{k+1}. It is shown how the auxiliary points $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i (i = 0, ..., 7)$ are derived from $\mathbf{p}_k, \mathbf{q}_1, \mathbf{q}_2, \mathbf{p}_{k+1}$. The resulting de Boor points $\mathbf{d}_i$ are linear combinations of $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ using the weight parameter w.

additional points $\mathbf{q}_1, ..., \mathbf{q}_{n_k}$) on the curve. The parameter w can be moved interactively by the user. For $w = 0$ the curve degenerates into the line segment $(\mathbf{p}_k, \mathbf{p}_{k+1})$. This means that the concept of normal parallel coordinates is still a special case of higher order parallel coordinates. In this case only correlations of two main axes can be detected, the auxiliary axes and their correlations do not play a role there. For $w = 2$, the curve turns out to be the sequence of line segments $(\mathbf{p}_k, \mathbf{q}_1), (\mathbf{q}_1, \mathbf{q}_2), ..., (\mathbf{q}_{n_k-1}, \mathbf{q}_{n_k}), (\mathbf{q}_{n_k}, \mathbf{p}_{k+1})$. In this case, only correlations between the pairs of dimensions (X_k, Y_1), (Y_1, Y_2),...,(Y_{n_k-1}, Y_{n_k}), (Y_{n_k}, X_{k+1}) can be detected. To see correlations between all the dimensions, w has to be moved interactively between the extreme values $w = 0$ and $w = 2$. The choice of a particular w which best represents the correlations depends on the data, the screen size and the experience of the user. Thus the user should be able to interactively move the parameter to emphasize different kinds of pairwise correlations and so explore global correlations.

Figure 5.16 illustrates the influence of w for a configuration where two additional axes are inserted between two adjacent main axes. The test data in this figure was constructed in such a way that an observation case has approximately inverse values in the two main axes X_k, X_{k+1}, Furthermore, the values in the auxiliary axes Y_1 coincides with X_{k+1} while the values of Y_2 coincide with X_k. Hence this is a data set with strong correlations between the 4 dimensions X_k, Y_1, Y_2, X_{k+1}.

5.2.1.2 Detecting correlations between more than two dimensions

Given a set of points in n-dimensional data space, the usual parallel coordinate approach is useful for detecting correlations between two dimensions if the corresponding coordinate axes are located adjacent to each other. In real data, correlations between more than two dimensions are possible. Suppose there are correlations between the pairs of dimensions $(d_1, d_2), (d_2, d_3), (d_3, d_1)$. Then a relevant correlation between all three dimensions (d_1, d_2, d_3) may exist or not. In [184] the correlations between three and more dimensions are considered by applying Shannon's information theory ([168]). Interpreting each dimension of the table which describes the data as a random variate, and each observation case as the simultaneous realization of the n random variates, the joint information between a tuple of random variates can be computed. If this value

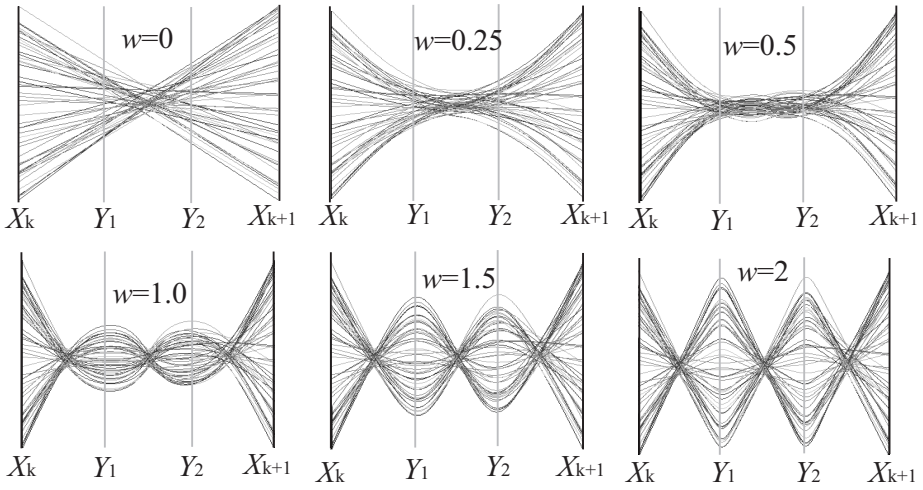


Figure 5.16: Two additional coordinate axes between two adjacent main axes. Higher order parallel coordinates for different choices of w.

exceeds a certain threshold, a relevant correlation between the dimensions has been detected. See [184] for details of this.

To test the ability of higher order parallel coordinates to deal with correlations between more than two dimensions, we explore a number of 4-dimensional test data sets. They are visualized by placing two additional axes Y_1, Y_2 between the main axes X_k, X_{k+1}.

The test data set shown in figure 5.17a consists of equally distributed random values on the main axis X_k. The values for the dimensions Y_1, Y_2 are approximately the same as the values for X_1 (except for slight perturbations). The value of X_{k+1} is approximately the inverse value of X_k (except for a slight perturbations). Thus here is a relevant correlation between the four dimensions. The visualization shows similar and regular patterns.

Figure 5.17b shows another quadruple of dimensions with relevant correlations between each other. Here X_k and Y_1 have essentially the same values while Y_2 and X_{k+1} have essentially inverse values. Again, similar and regular patterns can be recognized.

Figure 5.17c shows the correlations between the four dimensions in such a way that X_k and Y_2 have essentially the same values while Y_1 and X_{k+1} have essentially inverse values. Again the visualization looks regular.

In figure 5.17d Y_1 was chosen independently of the other dimensions. Although there are still correlations between the three dimensions X_k, Y_2, X_{k+1}, the visualization looks "wild". No similar behavior of the curves can be detected.

Figure 5.17 gives reason for the following statement: higher order parallel coordinates seem to provide a way of detecting and visualizing correlations between the dimensions $X_1, ..., X_m$. We choose two of these axes as main axes and place the remaining axes as additional axes between them. The set of all curves between the main axes gives an impression of the correlations between $X_1, ..., X_m$. If the curves show a similar behavior, correlations between $X_1, ..., X_m$ can be inferred. If the curves behave independently of each other, no symmetric pattern can be recognized in the curve plot. Then no relevant corre-

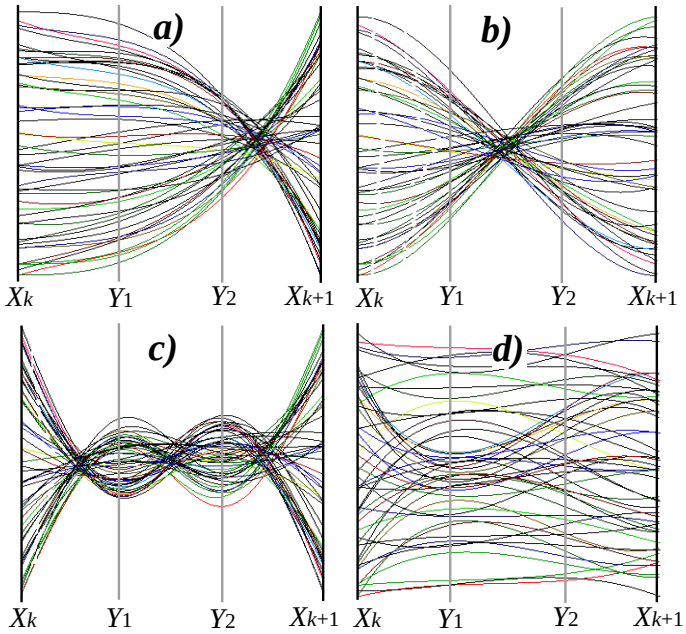


Figure 5.17: Two additional axes Y_1, Y_2 are placed between the main axes X_k, X_{k+1}. a)-c): correlation between all 4 dimensions. d): no correlation between all 4 dimensions. Y_2 was chosen independently of the other dimensions.

lation is found. In [117] further tests are done which confirm these statements.

Another example of constructed test data sets is shown in figures 5.18 and 5.19. Both test data set 1 (shown in figure 5.18) and test data set 2 (shown in figure 5.19) consist of 70 points in an 8-dimensional data space. No differences are visible in the visualization using usual parallel coordinates (figures 5.18a and 5.19a). We can see correlations between adjacent dimensions: objects with a high value in the axis X_i have to a high probability a high value in X_{i+1} as well. We cannot see whether or not there are correlations between more than two dimensions.

Figures 5.18b and 5.19b use higher order parallel coordinates for the test data sets 1 and 2. Between each two main axes two additional axes are inserted. In figure 5.18b the curves between the main axes do not show a similar behavior. Thus no correlations between quadruples of dimensions are found in test data set 1. In figure 5.19b the curves between the main axes show a similar behavior; we found quadruples of dimensions with correlations to each other. Here it makes sense to look for correlations between all 8 dimensions.

Figures 5.18c and 5.19c show another application of higher order parallel coordinates for the test data sets 1 and 2. Here we have two main axes X_1, X_2, and 6 additional axes between them. Figure 5.18c shows no correlations between them in data set 1. This is no new information after analyzing figure 5.18b; it is only for comparison with figure 5.19c. In figure 5.19c the curves show a similar behavior. Thus there are correlations between all 8 dimensions. All visualizations for higher order parallel coordinates in figures 5.18 and 5.19 were computed using the weight $w = 1$.

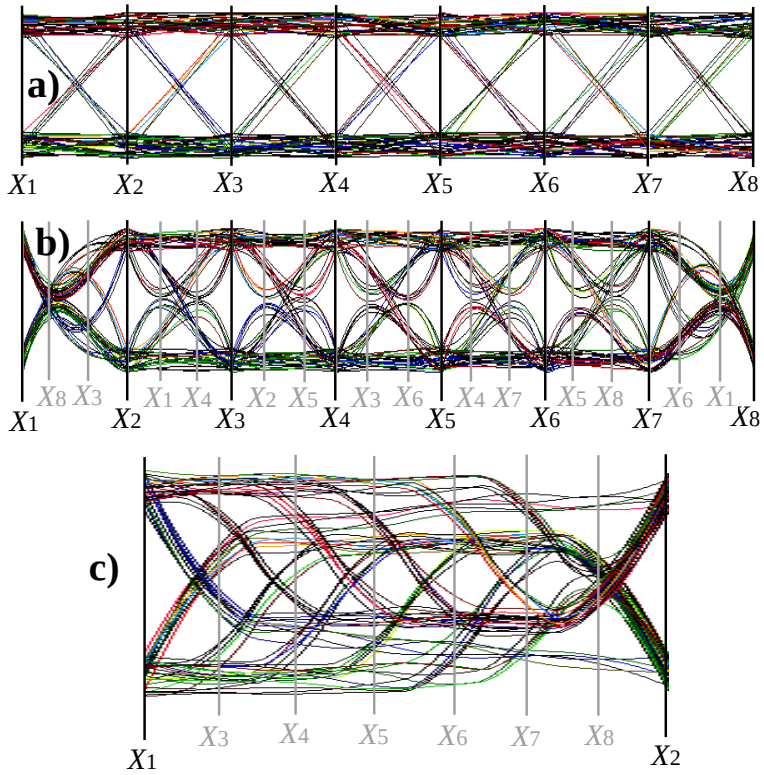


Figure 5.18: Test data set 1 ; a) normal parallel coordinates; b) higher order parallel coordinates, between each two main axes, two additional axes are inserted ; c) higher order parallel coordinates; two main axes X_1, X_2, 6 additional axes between them.

5.2.1.3 Application scenario for higher order parallel coordinates

Given an n-dimensional data set we start out with the usual parallel coordinate approach. Here the order of the axes is subject to interactive change. If correlations between adjacent axes X_k, X_{k+1} are detected, additional axes can be inserted between X_k and X_{k+1} in order to explore whether or not there are correlations between more than two axes (including X_k and X_{k+1}). To do so, the weight parameter is interactively moved between 0 and 2. We demonstrate this on the car data set[7] which was already explored in section 5.1.1 using the ShapeVis approach.

Figure 5.20a shows the visualization using usual parallel coordinates. We can detect correlations between the following pairs of dimensions: (MPG, Weight), (Horsepower, Displacement), (Displacement, Cylinders). No relevant correlations are found between (Weight, Driveratio), (Driveratio, Horsepower). Figure 5.20b shows the visualization using higher order parallel coordinates with one additional axis between each pair of adjacent main axes. Correlations between triples of dimensions can be detected for (Horsepower, Weight, Displacement), (Displacement, Horsepower, Cylinders). No relevant correlations are found in the triplets (MPG, Driveratio, Weight), (Weight, Horsepower, Driveratio),

[7] available at http://lib.stat.cmu.edu/DASL/Stories/ClusteringCars.html

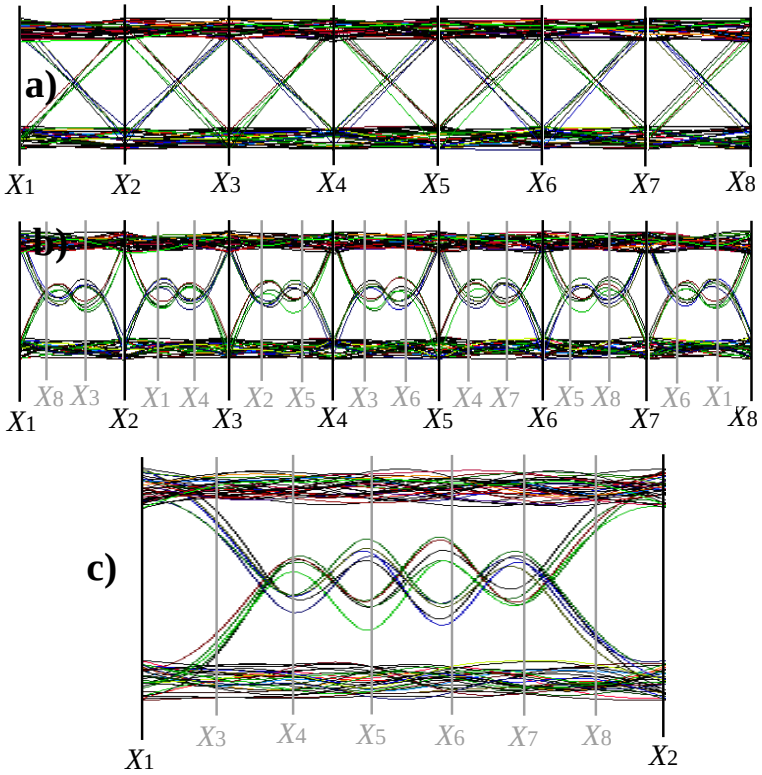


Figure 5.19: Test data set 2 ; a) normal parallel coordinates; b) higher order parallel coordinates, between each two main axes, two additional axes are inserted ; c) higher order parallel coordinates; two main axes X_1, X_2, 6 additional axes between them.

(Driveratio, Cylinders, Horsepower). The curves for these triples do not show similar patterns.

As with the usual parallel coordinate approach, using higher order parallel coordinates is an interactive process. The user has to find appropriate sequences of main axes as well as the additional axes between them. To explore the behavior of the curve the user can move the weight parameter interactively.

Obviously, the success of higher order parallel coordinates depends on the ability of the initial usual parallel coordinates approach to detect all pairs of dimensions with relevant correlations. This is not a trivial task and for usual parallel coordinates is only possible by (interactively or automatically) finding suitable arrangements and exchanges of the coordinate axes. For a higher number of dimensions this is going to be laborious. There a combination with scatter plot matrices is possible to find all pairs of dimensions with relevant correlations to each other.

5.2.2 Theoretical considerations for higher order parallel coordinates

Up to here, higher order parallel coordinates have been introduced from an perception driven standpoint. Straight line segments have been replaced by

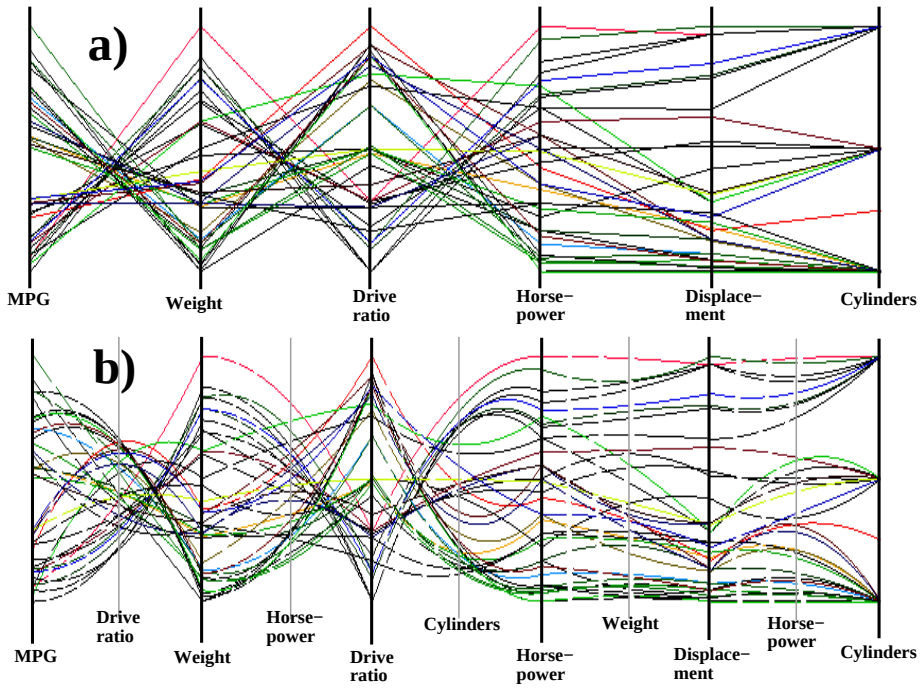


Figure 5.20: Car data set; a) usual parallel coordinates; b) higher order parallel coordinates with one additional axis between each pair of adjacent main axes.

curves, and it has been observed that in this way correlations between more than two dimensions may become visible. However, normal parallel coordinates have a strong theoretical background. So it makes sense to ask for the theoretical consequences of replacing lines by curves to get a deeper understanding of higher order parallel coordinates.

We start with collecting some well-known properties of normal parallel coordinates for only two dimensions. Given is a point $\mathbf{x} = (x_1, x_2)$ in 2D cartesian coordinates. Placing the two coordinate axes X_1, X_2 in the cartesian coordinate system parallel to the y-axis at the locations a_1, a_2, the point $\mathbf{x}$ in cartesian coordinates can be represented in parallel coordinates by the straight line

$$\mathbf{s}_\mathbf{x}(t) = (1-t) \begin{pmatrix} a_1 \\ x_1 \end{pmatrix} + t \begin{pmatrix} a_2 \\ x_2 \end{pmatrix}.$$

Figure 5.21 gives an illustration.

Now we consider not only one point in parallel coordinates but a set of points which are located on a straight line. The pointwise transformation of these points into parallel coordinates gives a bundle of lines in parallel coordinates which intersect in a common point. Thus a line in cartesian coordinates is represented by a point in parallel coordinates; we have a *point* $\leftrightarrow$ *line duality* between cartesian coordinates and parallel coordinates (see [98]). Figure 5.22 gives an illustration.

Now we consider the image of a set of points in cartesian coordinates which are located not on a straight line but on a general curve. Figure 5.23a shows a number of points in cartesian coordinates which are located on a pair of

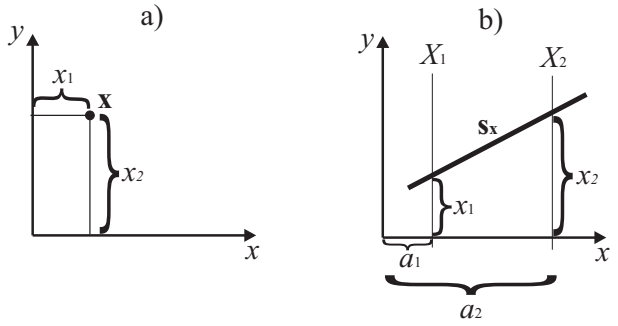


Figure 5.21: Parallel coordinates: the point $\mathbf{x}$ in a) is represented in parallel coordinates by the line $\mathbf{s_x}$ in b).

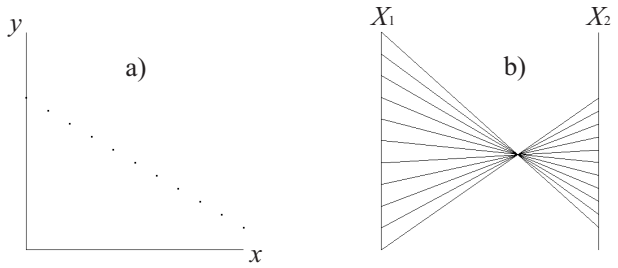


Figure 5.22: Point $\leftrightarrow$ line duality; a set of points located on a line in cartesian coordinates shown in a) is mapped to a set of lines intersecting in a common point in parallel coordinates shown in b).

hyperbolas. Their images in parallel coordinates are shown in figure 5.23b. As we can see there, this set of lines envelopes an ellipse. This ellipse (shown in figure 5.23c) can be considered as the parallel coordinate image of the hyperbola of figure 5.23a. In fact, considering the family of lines in parallel coordinates as the envelope of a curve, it has been shown ([100]) that there is a *conic* $\leftrightarrow$ *conic duality* between cartesian coordinates and parallel coordinates.

Now we want to extend this duality to a general *curve* $\leftrightarrow$ *curve duality* between cartesian coordinates and parallel coordinates. Suppose the curve $\mathbf{x}(t) = (x_1(t), x_2(t))^T$ in cartesian coordinates is the dual to the curve $\mathbf{p}(t) = (p_1(t), p_2(t))^T$ in parallel coordinates. (The curve $\mathbf{p}(t)$ is considered to be a point curve; its envelope (i.e. its set of tangent lines) is the set of lines in parallel coordinates which are the dual counterparts of the points of $\mathbf{x}(t)$). To establish the correlation between $\mathbf{x}(t)$ and $\mathbf{p}(t)$, we have to show how to compute $\mathbf{x}(t)$ from $\mathbf{p}(t)$ and vice versa. Given the curve $\mathbf{p}(t)$, the curve $\mathbf{x}(t)$ can be found by intersecting the tangent lines of $\mathbf{p}(t)$ with the coordinate axes X_1, X_2, and considering the y-coordinates of these intersection points as the coordinates of $\mathbf{x}(t)$:

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ \\ x_2(t) \end{pmatrix} = \begin{pmatrix} p_2(t) - (\, p_1(t) - a_1 \,) \frac{\dot{p}_2(t)}{\dot{p}_1(t)} \\ \\ p_2(t) - (\, p_1(t) - a_2 \,) \frac{\dot{p}_2(t)}{\dot{p}_1(t)} \end{pmatrix}. \tag{5.13}$$

Figure 5.24 illustrates this.

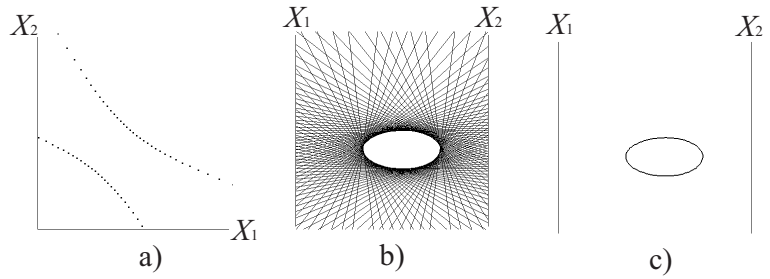


Figure 5.23: Conic $\leftrightarrow$ conic duality; a) a set of points located on a hyperbola; b) their images in parallel coordinates envelopes an ellipse (c).

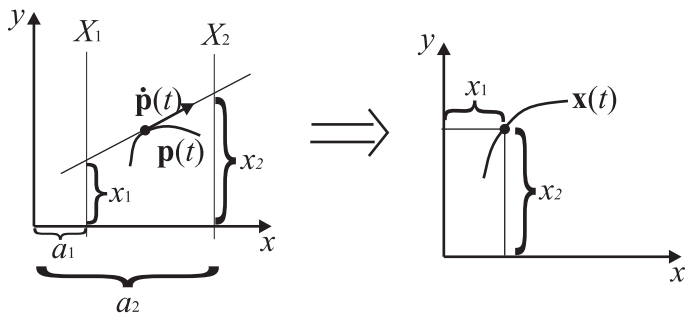


Figure 5.24: Obtaining $\mathbf{x}(t)$ in cartesian coordinates from $\mathbf{p}(t)$ in parallel coordinates.

To obtain $\mathbf{p}(t)$ from $\mathbf{x}(t)$ we use the fact that the point $\mathbf{p}(t)$ must be on the line $\mathbf{s}_{\mathbf{x}(t)}$ for any point $\mathbf{x}(t)$ on the curve. Hence we use the approach

$$\mathbf{p}(t) = (1-\alpha(t)\,)\begin{pmatrix} a_1 \\ x_1(t) \end{pmatrix} + \alpha(t)\begin{pmatrix} a_2 \\ x_2(t) \end{pmatrix}. \tag{5.14}$$

Figure 5.25 illustrates this. The only remaining problem is to find the function $\alpha(t)$. To do so, we compute the tangent vector of $\mathbf{p}(t)$ by computing the derivative of (5.14):

$$\dot{\mathbf{p}}(t) = \begin{pmatrix} \dot{p}_1(t) \\ \dot{p}_2(t) \end{pmatrix} = \begin{pmatrix} \dot{\alpha}(t)\,(a_2-a_1) \\ (\,(1-\alpha)\,\dot{x}_1 \;+\; \alpha\,\dot{x}_2 \;+\; \dot{\alpha}\,(x_2-x_1)\,)(t) \end{pmatrix}. \tag{5.15}$$

In order to obtain the unknown function $\alpha(t)$, we consider the curve $\mathbf{y}(t)$ which is obtained by reverse-transforming $\mathbf{p}(t)$ given by (5.14) and (5.15) into cartesian coordinates. This means that we insert (5.14) and (5.15) into

$$\mathbf{y}(t) = \begin{pmatrix} y_1(t) \\ \\ y_2(t) \end{pmatrix} = \begin{pmatrix} p_2(t) - (p_1(t)-a_1)\,\frac{\dot{p}_2(t)}{\dot{p}_1(t)} \\ \\ p_2(t) - (p_1(t)-a_2)\,\frac{\dot{p}_2(t)}{\dot{p}_1(t)} \end{pmatrix}.$$

and obtain

$$\mathbf{y}(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \frac{(1-\alpha(t))\,\dot{x}_1(t) \;+\; \alpha(t)\,\dot{x}_2(t)}{\dot{\alpha}} \begin{pmatrix} -\alpha(t) \\ 1-\alpha(t) \end{pmatrix}.$$

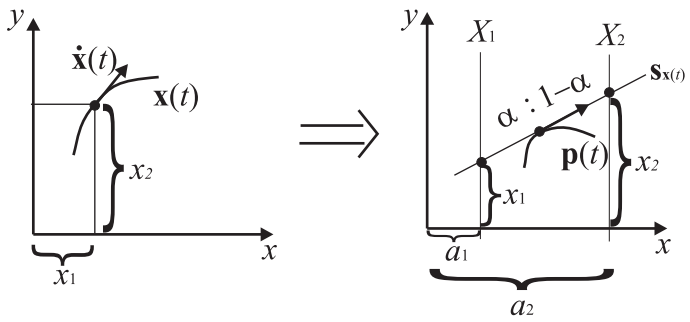


Figure 5.25: Obtaining $\mathbf{p}(t)$ in parallel coordinates from $\mathbf{x}(t)$ in cartesian coordinates: the function $\alpha(t)$ has to be found.

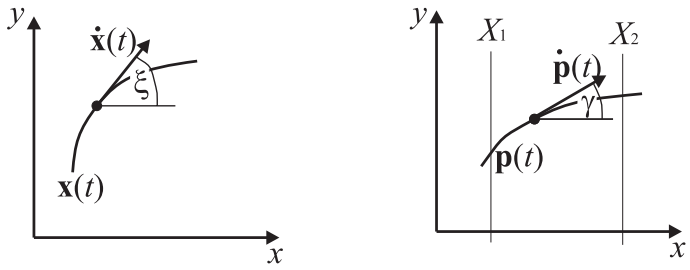


Figure 5.26: Slope angles γ and ξ.

From this and $\mathbf{y}(t) = \mathbf{x}(t)$ we obtain the condition

$$(1 - \alpha(t))\,\dot{x}_1(t) \;+\; \alpha(t)\,\dot{x}_2(t) = 0$$

which yields

$$\alpha(t) = \frac{\dot{x}_1(t)}{\dot{x}_1(t) - \dot{x}_2(t)}. \tag{5.16}$$

This way (5.14) and (5.16) give the transformation from $\mathbf{x}(t)$ to $\mathbf{p}(t)$ which shows the curve $\leftrightarrow$ curve duality between cartesian coordinates and parallel coordinates.

Having introduced how to transform a curve $\mathbf{x}(t)$ into a dual curve $\mathbf{p}(t)$, we ask for common geometric characteristics of both curves. Let

$$\xi(t) = \arctan \frac{\dot{x}_2(t)}{\dot{x}_1(t)}$$

be the slope angle of the curve $\mathbf{x}(t)$, and let

$$\gamma(t) = \arctan \frac{\dot{p}_2(t)}{\dot{p}_1(t)}$$

be the slope angle of the dual curve $\mathbf{p}(t)$. Figure 5.26 illustrate ξ and γ. Then we search for dual properties in the curvature of $\mathbf{x}(t)$ and $\mathbf{p}(t)$. To do so, we have

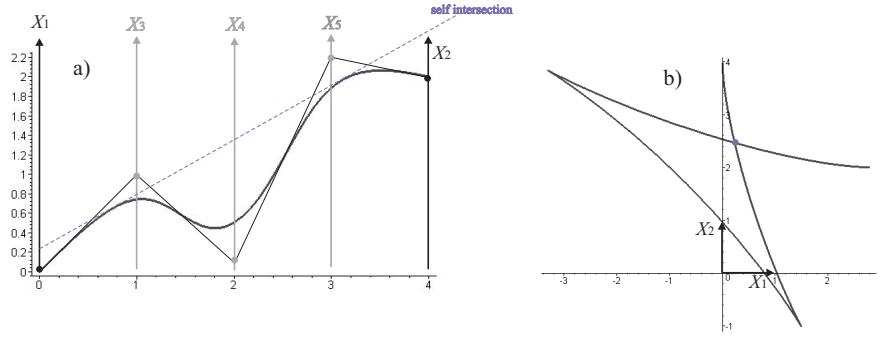


Figure 5.27: a) higher order parallel coordinates: 3 additional axes X_3, X_4, X_5 are placed between the two main axes X_1, X_2; shown is the 5-dimensional point (0 , 2 , 1, 0.1 , 2.2); b) transformation of the curve in a) into cartesian coordinates; the self-intersection in this curve corresponds to the common tangent of two parts of the curve in a) (dashed line).

to consider $\dot{\mathbf{x}}(t)$ and $\ddot{\mathbf{x}}(t)$ as well as $\dot{\mathbf{p}}(t)$ and $\ddot{\mathbf{p}}(t)$ by computing the derivatives of (5.14) and (5.16). Let $\kappa_{\mathbf{x}}(t)$ be the curvature of $\mathbf{x}(t)$, and let $\kappa_{\mathbf{p}}(t)$ be the curvature of $\mathbf{p}(t)$. Then we obtain the following correlation[8]:

$$\kappa_{\mathbf{x}}(t)\ \kappa_{\mathbf{p}}(t) = \frac{\sqrt{2}}{4\,(a_2 - a_1)^2}\ (\ (1 - \sin 2\xi\,)\,(1 + \cos 2\gamma\,)\ )^{\frac{3}{2}} \tag{5.17}$$

(5.17) shows that the curvatures of $\mathbf{x}$ and $\mathbf{p}$ behave reversed. If one curve has a large curvature, the other one has a rather small curvature. In fact, also the cusp $\leftrightarrow$ inflection point duality between cartesian coordinates and parallel coordinates (see [98]) is a byproduct of (5.17).

5.2.2.1 Application of the curve $\leftrightarrow$ curve duality to higher order parallel coordinates

Now we can apply the curve $\leftrightarrow$ curve duality introduced above to higher order parallel coordinates. In common parallel coordinates, a point in n-dimensional cartesian space is transformed to a sequence of straight lines. To understand higher order parallel coordinates, we have to find the objects in n-dimensional cartesian space which transformation to parallel coordinates gives a sequence of curves, i.e higher order parallel coordinates. Following the curve $\leftrightarrow$ curve duality, these objects are n-dimensional curves in cartesian coordinates.

Figure 5.27a shows a higher order parallel coordinate approach for a 5- dimensional point. Here we have chosen two main axes X_1, X_2 and 3 additional axes X_3, X_4, X_5 between them, the weight w (see (5.12)) was set to 1. This curve in parallel coordinates can be transformed to cartesian coordinates by (5.13). The resulting dual curve in cartesian coordinates is shown in figure 5.27b. The two cusps of the curve in figure 5.27b correspond to the two inflection points of the curve in figure 5.27a. The self-intersection of the curve in figure 5.27b

[8] by computing $\dot{\mathbf{p}}(t)$ and $\ddot{\mathbf{p}}(t)$ from (5.14) and (5.16), computing $\kappa_{\mathbf{x}}(t)$ from $\dot{\mathbf{x}}(t)$ and $\ddot{\mathbf{x}}(t)$, and computing $\kappa_{\mathbf{p}}(t)$ from $\dot{\mathbf{p}}(t)$ and $\ddot{\mathbf{p}}(t)$. These computations were done by a formula manipulating program.

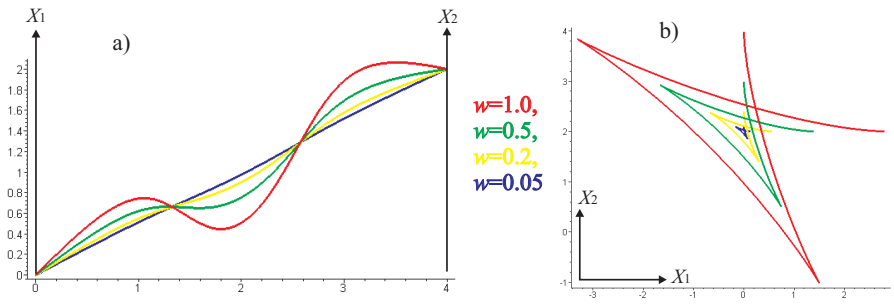


Figure 5.28: Influence of the weight w; a) higher order parallel coordinates; b) dual curve of a) in cartesian coordinates.

corresponds to the dashed line in figure 5.27a which is tangent of two different curve points.

Since the curve in higher order parallel coordinates is a piecewise C^2 continuous cubic B-spline curve described by (5.10) – (5.12), (5.13) yields that its dual curve in cartesian coordinates is a piecewise C^1 continuous cubic B-spline curve.

Now we explore the impact of the weight w (see (5.12)) in higher order parallel coordinates. Figure 5.28a shows the higher order parallel coordinate curves of the same example as in figure 5.27a for 4 different choices of w, including the special case $w = 0$ which describes common parallel coordinates. Figure 5.28b shows the dual curves of figure 5.28a in cartesian coordinates. As we can see here, w causes an approximate downscaling of the curve. In fact, for the special case $w = 0$ the curve degenerates to a point.

The example in figure 5.28 gives reason for the following statement: the higher order parallel coordinate approach (introduced in section 5.2.1) and the ShapeVis approach (introduced in section 5.1.1) are based on a similar idea! In fact, ShapeVis maps a point in n-dimensional cartesian space to a curve/surface in 2D/3D cartesian space where location, size and shape of the curve/surface determine the whole amount of present information. Higher order parallel coordinates does essentially the same but maps these curves from cartesian coordinates to parallel coordinates. As discussed in section 5.2.1, this has the advantage that correlations between a certain number of dimensions can be detected while ShapeVis is limited to global statements about the similarity of observation cases.

From figure 5.28 we can also see that the weight w (see (5.12)) in higher order parallel coordinates has the inverse impact to the value c (see (5.2)–(5.4)) in the ShapeVis approach. A high c leads to a collapsing of the Curves/surfaces to a point in ShapeVis while a small w collapses a curve in higher order parallel coordinates to a sequence of straight lines and thus to a point in the higher dimensional data space.

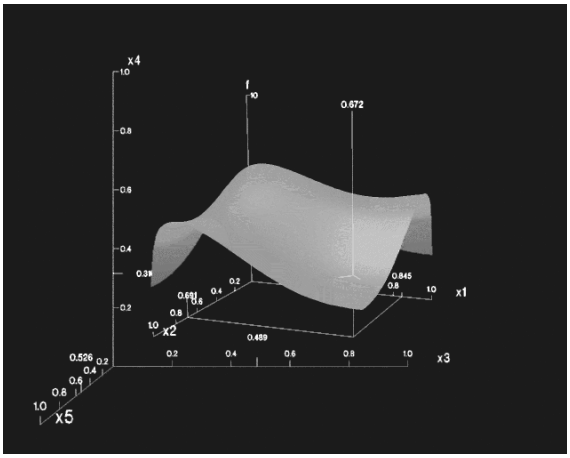


Figure 5.29: 5-dimensional function $f(x_1, x_2, x_3, x_4, x_5)$ in the Worlds-Within-Worlds technique; the last 3 dimensions build the outer coordinate system, at a particular location of this, the function $f(x_1, x_2)$ is visualized in the inner coordinate system as a height field (image from [18]).

5.3 Hierarchical Techniques

Hierarchical techniques try to find a mapping from a higher dimensional grid to a hierarchical grid arrangement in 2D or 3D. This way both local and global properties of a higher dimensional data set should become visible. Representatives of hierarchical techniques are Dimensional Stacking ([124]) and Cone Trees ([157]).

Due to the nature of these techniques, hierarchical techniques tend to demand a rather large screen space to determine the location of the higher dimensional grid. Hence there is little space left to represent the data values in the grid points; simple color techniques like encoding the data in the pixel color are preferred to the application of curves and surfaces.

An exception to this statement is the Worlds-Within-Worlds approach described in [18]. The idea there is to pick three coordinates of the higher dimensional coordinate system to built an "outer world". At a certain location of this outer world (i.e. for particular values of the outer three coordinates), a new "inner" coordinate system consisting of three of the remaining dimensions is constructed. This procedure may be repeated recursively until in the "most inner" coordinate system the data may be represented by a curve or a surface. Figure 5.29 gives an illustration of the visualization of a 5-dimensional function $f(x_1, x_2, x_3, x_4, x_5)$. The first three dimensions define the outer coordinate system. Hence the coordinates x_3, x_4, x_5 define the location in 3D in which a new coordinate system is created. In this coordinate system the 2-dimensional function $f(x_1, x_2)$ for a constant (x_3, x_4, x_5) is simply visualized as a height field. In addition, various interactive functionalities are offered by Worlds-Within-Worlds systems to define the coordinate systems and to navigate in them.

Contrary to other hierarchical techniques, Worlds-Within-Worlds focuses only on certain subspaces of the higher dimensional data space for visualiza-

tion. Hence the amount of data present may become small enough to enable the application of surfaces. However, the applied surfaces are usually simply height surfaces. Their treatment does not need further CAGD methods and ideas.

Chapter 6

CAGD for Further Data Classes

Considering the classification of scientific data of [23] and section 2.1, there is a number of data classes we have not treated yet. In fact, up to now we have only considered volume data, flow data and multiparameter data which are currently the most popular data classes in Visualization.

It is the purpose of this chapter to give an overview on CAGD methods for some of the remaining data classes. None of these classes is treated in detail here, either because no systematic application of CAGD methods is done or because a complete and exhaustive literature on this issue already exists. In the following sections 6.1 – 6.3 we treat scattered data, tensor data, and information visualization.

6.1 Scattered Data

The defining property of scattered data is the fact that the data values are not located on a regular grid. On the contrary, usually no grid is given at all. So we might have to deal with the problem that in some areas there are too many data values (oversampling) while in other areas the distribution of the data is rather coarse (undersampling).

Scattered data may be given in two ways:

- Given are n sample points $\mathbf{x}_i = (x_i, y_i),\ (i = 1, ..., n)$ in the 2D euclidian plane, and a scalar value $s_i,\ (i = 1, ..., n)$ for each of these points. Then we search for a scalar function $s(x, y)$ over the 2D euclidian plane which interpolates the scalars at the sample points, i.e. $s(x_i, y_i) = s_i$ for $(i = 1, ..., n)$. We call this kind of data *2D scattered data*.
- Given are n sample points $\mathbf{x}_i = (x_i, y_i, z_i),\ (i = 1, ..., n)$ in the 3D euclidian space. We search for a surface which interpolates these n sample points. We call this kind of data *3D scattered data*.

Each of these scattered data classes is treated in one of the following sections 6.1.1 and 6.1.2. Research has also been done on multidimensional scattered data (see [4]) which is not treated here.

6.1.1 2D scattered data

Concerning the data classification of [23], 2D scattered data can be expressed as $E^S_{[2]}$ where no connectivity of the sample points in the 2D domain is assumed. Interpreting the scalars s_i as height values over the (x, y)-domain, the 2D scattered data problem can be considered as a surface interpolation problem by searching for an interpolating height surface $s(x, y)$. In fact, the most complicated part of dealing with 2D scattered data is to find an appropriate interpolation. The remaining parts of visualizing scattered data (mapping, rendering – as in the usual visualization pipeline for scientific data) can be done by standard methods for surfaces. Hence the field of dealing with 2D scattered data is called scattered data interpolation instead scattered data visualization.

A variety of interpolation schemes for scattered data have been developed. Since there is number of excellent and comprehensive surveys on this issue (see [95], [62], [63]), we can restrict ourselves here to a rough classification of 2D scattered data methods.

The methods of 2D scattered data interpolation can be classified in three groups (see [95]):

- Shepard methods
- Radial basis functions
- Surfaces over a triangulation.

Especially for the last-named group, a variety of CAGD methods can be applied. (In fact, many of these methods have been developed for scattered data interpolation.) After building a triangulation over the sample points in the 2D domain, a triangular surface is constructed over each of the resulting triangles. By estimating derivative information in the vertices of the triangulation, a certain smoothness of the adjacent triangular patches can be achieved. Standard approaches are the Clough-Tocher interpolant ([11], [55]) the Powell-Sabin interpolant ([153],[55]), and Nielson's C^1 interpolant ([140]).

6.1.2 3D scattered data

Using the data classification of [23] again, 3D scattered data can be described as $E^P_{[3]}$ where no connectivity of the sample points is assumed. This means in particular that only the location of the 3D sample points matters while the scalar values at these sample points are not considered. This class of data often appears by 3D scanning of bodies and landscapes.

Also for this kind of scattered data a number of comprehensive surveys exist ([132], [4]). Most of the approaches here focus on constructing a piecewise triangular interpolating surface where the vertices of the triangles are the sample points $\mathbf{x}_i$. If a higher degree of smoothness is desired, triangular surface patches may be constructed using the same basic approaches as for surfaces over triangulations for 2D scattered data. Smooth surface approaches which were especially developed for 3D scattered data can be found in [48], [77].

6.2 Tensor Data

In recent years the visualization of tensor data has become a growing subject in scientific visualization. Tensors of an order n can be considered as an n-dimensional matrix which has a certain transformation behavior. The tensors which are most often treated in scientific visualization are symmetric second order tensors. This type of tensor appears by describing the diffusion behavior of objects or the affecting stress inside a solid object. Also the Jacobian determinant of a 3D vector field (see section 4.1.8) can be interpreted as a second order tensor.

A *symmetric second order tensor field* can be described as a map $\mathbf{T}$ from the 3D domain into the vector space of symmetric second order tensors:

$$\mathbf{T}(x,y,z) = \begin{pmatrix} u_{11}(x,y,z) & u_{12}(x,y,z) & u_{13}(x,y,z) \\ u_{12}(x,y,z) & u_{22}(x,y,z) & u_{23}(x,y,z) \\ u_{13}(x,y,z) & u_{23}(x,y,z) & u_{33}(x,y,z) \end{pmatrix}.$$

It can be shown that any second order tensor (i.e. any 3×3 matrix) can be broken up into a symmetric second order tensor and and a vector ([46]). For visualization purposes, the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ of $\mathbf{T}$ and their corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are of special interest.

A number of techniques have been developed to visualize tensor fields. Since tensor fields can be interpreted as a generalization of vector fields, most of the visualization techniques for tensor fields are generalizations of vector field techniques. Among the techniques for tensor field visualization there is a number of techniques which uses curves and surfaces. In the following we collect these techniques.

One popular approach to visualizing tensor fields is to place appropriate icons at certain locations in the flow which contain the tensor information for a particular location in the field. For this purpose, an ellipsoid ([113]) is especially useful as an icon because it covers exactly all degrees of freedom of a symmetric second order tensor. To do so, the three main directions of the ellipsoid are the three eigenvectors of $\mathbf{T}$ while the three main radii of the ellipsoid are the three eigenvalues of $\mathbf{T}$. Figure 6.1 illustrates an example of visualizing a symmetric second order tensor as an ellipsoid.

If a tensor field $\mathbf{T}$ describes the diffusion of an object, a ellipsoid as visualization technique has a nice geometric interpretation: it describes the shape of a drop of fluid set out at the current location after a short time of diffusion. In [114] the tensor characteristics are mapped onto the color on spheres which are directly volume rendered.

For vector fields, stream lines have been proven to be a useful visualization tool. The counterparts for tensor field are *hyperstreamlines* ([46]) which are simply the stream lines of the three eigenvector fields $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$. Figure 6.2a shows some minor hyperstreamlines (i.e. the streamlines of the eigenvector-field corresponding to the smallest eigenvalue) for a stress tensor field induced by two compressive forces. In [206] tensor lines are used instead of hyperstreamlines. Since a diffusion process is a probabilistic phenomenon, tensor lines incorporate a probabilistic propagation of the path. This has been shown to be especially useful in isotropic areas, i.e. in areas where at least two of the eigenvalues of

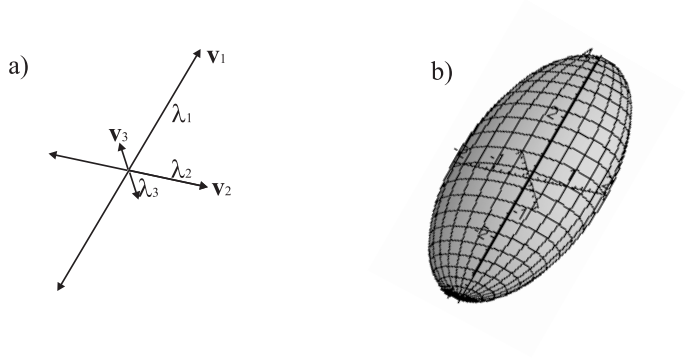


Figure 6.1: Visualizing a symmetric second order tensor $\mathbf{T}$ as an ellipsoid; a) $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are the eigenvectors of $\mathbf{T}$ while $\lambda_1, \lambda_2, \lambda_3$ are the eigenvalues of $\mathbf{T}$; b) resulting ellipsoid.

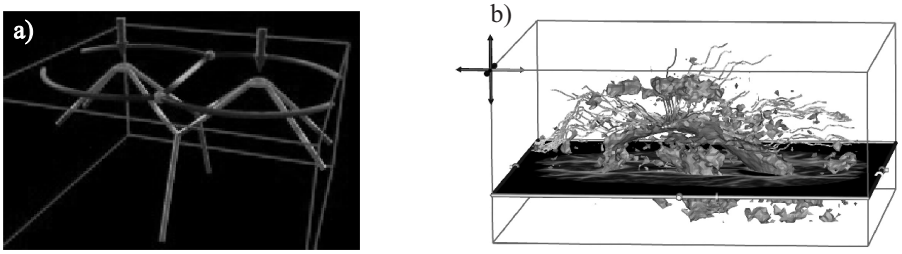


Figure 6.2: a) minor hyperstreamlines for a stress tensor field (image from [123]); b) tensor lines (yellow) and hyperstreamlines (cyan) for a diffusion data set (image from [206]).

$\mathbf{T}$ are close to each other. Figure 6.2b shows tensor lines (yellow) and hyperstreamlines (cyan) for a diffusion data set.

Another method which is directly extended from vector fields is the consideration of the topology of symmetric second order tensor fields. To do this we have to determine the tensor counterparts of critical points in vector fields. [90] and [123] show that these points are degenerate points which are defined by the property that at least two eigenvalues of $\mathbf{T}$ collapse. [90] classifies these degenerate points and introduces in this way a visualization technique for tensor fields similar to [87] for vector fields.

In [21], tensor fields are visualized by deformation surfaces: an initial interrogation surface (for example a plane segment) is deformed in response to the stress tensor acting upon it. This way the resulting deformed surfaces give insight into the behavior of the tensor field.

6.3 Information Visualization

In the last years information visualization has become one of the "hot topics" of Scientific Visualization. Currently there are even aspirations to make it a research area of its own and to develop it independently of classical Scientific Visualization.

This is a remarkable tendency especially because there is no generally accepted definition of the concept "information visualization" yet. In particular the distinction between the "classical" data visualization and information visualization is not done consistently.

Particularly vague is the borderline between information visualization and the visualization of multiparameter data. In fact, information visualization is sometimes characterized as visualization of multiparameter data where the distinction between dependent and independent variables is abrogated. Instead, only "abstract" independent variables are considered. Following this characterization of information visualization, all approaches which we treated in chapter 5 belong to the field of information visualization. In fact, a number of techniques for information visualization was developed for multiparameter data, and vice versa. (For example, the ShapeVis approach of section 5.1.1 was originally developed in the context of information visualization.)

Another characterization which is sometimes used to define the concept of information visualization is the property that the data consists of additional structural information which cannot be described by data models such as [23]. This structural information may be a hierarchical or another network of relations between the data elements. A survey on information visualization can be found in [27].

Curves and surfaces in information visualization can be used both to represent the data values and to represent the internal structures inside the data. The representation of the data values was already discussed in chapter 5 of this work[1]. Also for the structural information, curves and surfaces can be applied. To show this, we give a few examples which do not claim to be complete.

A very popular approach in information visualization are Focus&Context techniques which visualize a small part of the data in full detail while giving a rough overview of the remaining parts. A representative of this class of techniques is the Hyperbolic Viewer in [137]. Figure 6.3 shows a technique called Magic Eye View ([25]) which maps hierarchical data onto the surface of a sphere. This way the center of focus of the technique can be changed by changing the view direction onto the sphere. Figure 6.3a shows a version of Magic Eye View where the edges of the hierarchy on the sphere are represented as straight lines. The meaningfulness of this visualization can be increased by representing the edges as rational quadratic Bézier curves (which correspond to great circles on the sphere), as shown in figure 6.3b.

Another application of surfaces in information visualization is the treatment of Blobs ([74], [177]). Blobs are implicit surfaces which are obtained by placing higher dimensional data points into the 3D space in an appropriate way and applying a cluster analysis there[2]. This way the Blobs are the isosurfaces of a certain field function which makes sure that the clusters are completely surrounded by the Blobs. To visualize Blobs, standard methods for extracting implicit surfaces can be used. Figure 6.4 shows an example.

Another application of implicit surfaces in information visualization can be

[1] Chapter 5 treats multiparameter data but is true in the context of information visualization as well.

[2] The usage of Blobs can also be considered as an approach to visualizing multidimensional data as described in chapter 5. Since this technique was explicitly developed under the concept of information visualization, we leave it in this section.

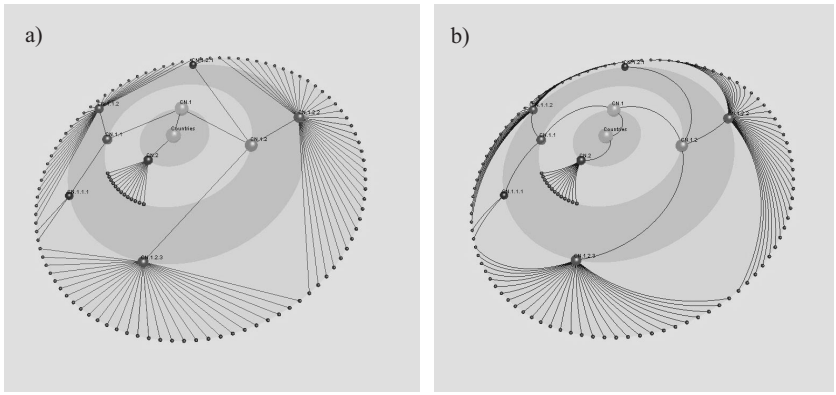


Figure 6.3: Magic Eye View for hierarchical data; the rings on the sphere represent the levels of hierarchy; a) edges of the graph represented as straight lines; b) edges of the graph represented as rational quadratic Bézier curves.

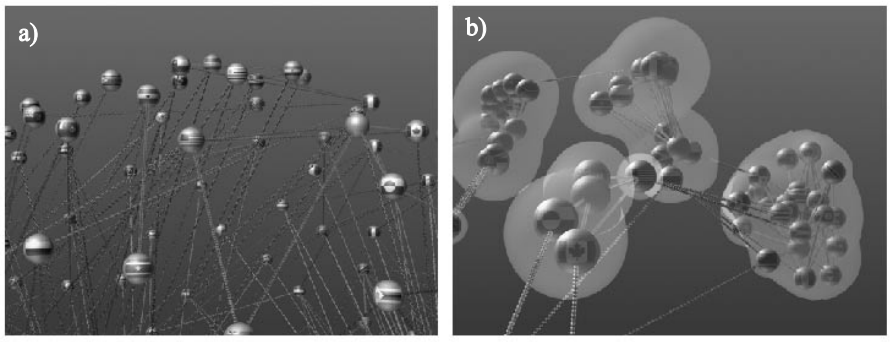


Figure 6.4: Blobs; a) initial object layout; b) Blob surfaces denoting the clusters (images from [177]).

found in [159] where the results of text analyses are visualized.

In [169] the applicability of superquadrics to information visualization is studied. There it was shown that the number of different superquadrics which can be distinguished by the human eye is rather limited. Hence superquadrics are only partially useful for information visualization.

Chapter 7

Scientific Visualization for CAGD

The CAGD design process is a complex and heterogeneous process which consists of a number of different steps and involves different kinds of data. Parts of this data have a similar structure to the data which is used in Scientific Visualization. These are the parts where Scientific Visualization can be applied for the CAGD process. Visualizing this data may give the designer information on how to continue the design process. Based on this visualization, he or she can decide if a redesign is necessary or if the next steps of the design process can be started.

The different processes and kinds of data which appear in the CAGD process were treated in the sections 2.1 and 2.3.2. We base the investigations in this chapter on the CAGD pipeline which we introduced in section 2.3.2. In particular we consider the present data at each step of the CAGD pipeline to find possible applications of Scientific Visualization. The left hand side of figure 7.1 shows the CAGD pipeline of figure 2.2 again. In addition, the right hand side of figure 7.1 classifies the present data in each part of the pipeline.

The input data at the beginning of the CAGD pipeline is either non-existing, an informal description (hand drawing), or a set of sample points. In the last named case, a plot of these points may give the designer an overview of the present data set in preparation for the next design steps. However, the graphical output of the sample points is just an application of basic 2D or 3D computer graphics; a particular knowledge about Scientific Visualization is not necessary.

In the next step of the CAGD pipeline, *specify task*, information is input which does not have the structure of visualization data. In fact, most of the decisions to be made here are binary decisions, such as the decisions about a desired interpolation and approximation of the data. Hence an application of Scientific Visualization in not appropriate there.

A similar statement holds for the next steps of the CAGD pipeline, *selection of the curve/surface type*, *specify further requests of curves/surfaces*, and *determine global degrees of freedom*. All these steps collect information about the properties of the curve/surface to be designed. Data of the kind used in

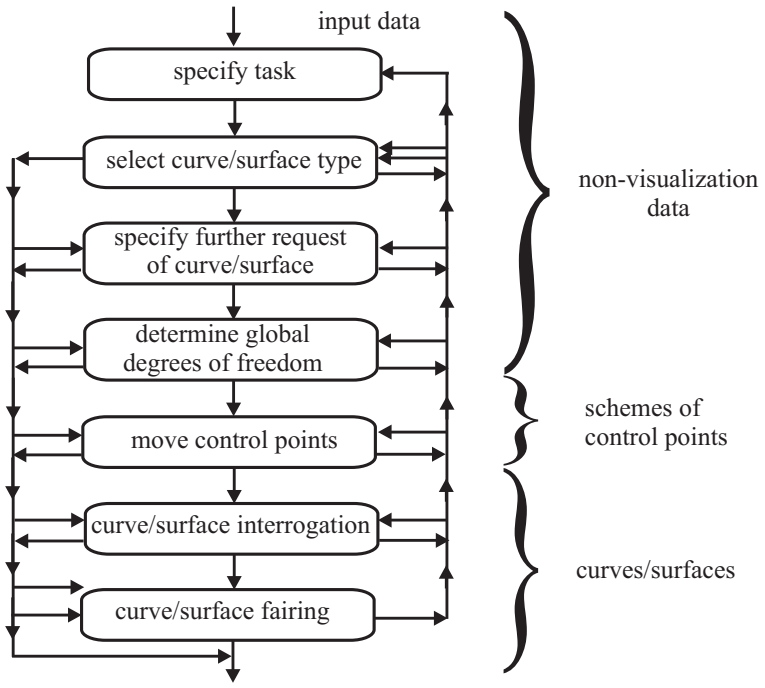


Figure 7.1: CAGD pipeline (left) and classification of the present data (right).

Scientific Visualization is not present.

After the step *determine global degrees of freedom* in the CAGD pipeline, an initial scheme of control points is present which can now be interactively moved by the designer. Such a scheme of control points can be considered as data of the type E_3^P (see section 2.1). Thus, in this part of the CAGD pipeline the application of Scientific Visualization becomes possible. We treat the application of Scientific Visualization for the step of moving the control points in section 7.1.

After finding appropriate locations of the control points, the scheme of control points is automatically converted to a curve/surface, i.e. data of the type $E_{[1]}^{V_2}$, $E_{[1]}^{V_3}$, or $E_{[2]}^{V_3}$ (see section 2.1). A number of techniques exist for providing visual representations of curves/surfaces. We give a summary of them in section 7.2.

In the next step of the CAGD pipeline, *curve/surface interrogation*, the quality of the designed curve/surface is analyzed. Here, a visual analysis is a natural approach. We dedicate section 7.3 to the application of Scientific Visualization for curve/surface interrogation.

The final step of the CAGD pipeline (*curve/surface fairing*), does not need a particular treatment in terms of Scientific Visualization because it can be seen in combination with surface interrogation methods. In fact, possible applications of visualization for surface fairing focus on curve/surface interrogation

are treated in section 7.3.

7.1 Visualization for Schemes of Control Points

To apply Scientific Visualization for schemes of control points, we see two approaches:

1. The structure of control points has to be specified in such a way that all degrees of freedom can be changed interactively and in an intuitive way.
2. The scheme of control points has to be displayed on the screen.

The second point is a rather trivial application of classical computer graphics. In fact, the scheme of control points is normally represented by a scheme of points and straight line segments which can be directly sent to the rendering pipeline of a graphical workstation. Further applications of knowledge which comes from Scientific Visualization is not necessary here.

For approach 1, the structure of the control points is usually directly given by the chosen type of curves/surfaces. For example, for Bézier- or B-spline curves/surfaces, the present control point schemes have the property that every degree of freedom is uniquely represented by the system of control points. In other words: each Bézier- or de Boor point can be moved freely (and independently of the other control points) by the designer.

If the chosen type of curve is a rational Bézier- or B-spline curve, the weights of the control points can be described by *Farin points*. Farin points are a design tool for handling the weights of rational curves in an intuitive way. In particular there is a one-to-one correlation between the weights and the Farin points for rational curves. This means that all weights can be controlled by the system of Farin points, and all Farin points can be moved independently of each other. Unfortunately, this property gets lost when considering Farin points on rational surfaces. In order to keep Farin points a useful design tool for rational surfaces, we propose the application of methods of Scientific Visualization in the section. In fact, we introduce appropriate icons to give the designer the necessary tools to design the weights of the surfaces. Before doing so, we briefly introduce Farin points for rational Bézier curves.

Given two Bézier points $\mathbf{b}_i, \mathbf{b}_{i+1}$ and their assigned weights w_i, w_{i+1} in affine space, we consider the corresponding points

$$\underline{\mathbf{b}}_i = \begin{pmatrix} w_i\,\mathbf{b}_i \\ w_i \end{pmatrix} \quad , \quad \underline{\mathbf{b}}_{i+1} = \begin{pmatrix} w_{i+1}\,\mathbf{b}_{i+1} \\ w_{i+1} \end{pmatrix}$$

in homogeneous coordinates in projective space. Defining the Farin point in projective space as $\underline{\mathbf{f}}_i = \frac{1}{2}(\underline{\mathbf{b}}_i + \underline{\mathbf{b}}_{i+1})$, its counterpart in affine space is

$$\mathbf{f}_i = \frac{w_i \cdot \mathbf{b}_i \; + \; w_{i+1} \cdot \mathbf{b}_{i+1}}{w_i \; + \; w_{i+1}}. \tag{7.1}$$

The location of $\mathbf{f}_i$ on the line through $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ determines the ratio of w_i and w_{i+1} uniquely and in an intuitive way. For positive weights, $\mathbf{f}_i$ is located

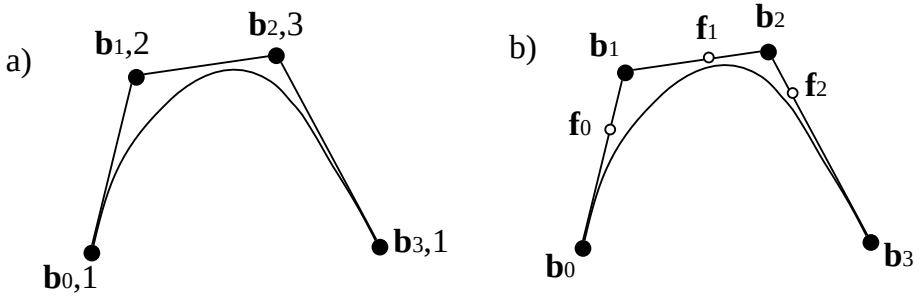


Figure 7.2: Rational Bézier curve of degree 3 with the weights $w_0 = 1$, $w_1 = 2$, $w_2 = 3$, $w_3 = 1$; a) described by Bézier points and assigned weights, b) described by Bézier points and Farin points.

between $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$, and ratio$(\mathbf{b}_i, \mathbf{f}_i, \mathbf{b}_{i+1}) = \frac{w_{i+1}}{w_i}$. Intuitively: the larger w_i is relative to w_{i+1}, the closer $\mathbf{f}_i$ moves toward $\mathbf{b}_i$.

Given the Bézier polygon $\mathbf{b}_0, ..., \mathbf{b}_n$, the assigned weights $w_0, ..., w_n$ can be described by the Farin points $\mathbf{f}_0, ..., \mathbf{f}_{n-1}$. Figure 7.2 shows an example for $n = 3$. Farin points for rational Bézier curves have the following properties:

- Uniqueness: the Farin points $\mathbf{f}_0, ..., \mathbf{f}_{n-1}$ describe the weights $w_0, ..., w_n$ uniquely except for a common factor. This common factor has no influence on the shape of the curve.
- Independence: each of the Farin points $\mathbf{f}_0, ..., \mathbf{f}_{n-1}$ can be moved freely on the lines of the Bézier polygon. The adjacent Farin points are not affected by moving $\mathbf{f}_i$.
- Intuitivity: instead of increasing or decreasing the weights a designer moves Farin points on the Bézier polygon. He or she may find this more intuitive.
- Extended convex hull: for positive weights, the rational Bézier curve lies not only in the convex hull of $\mathbf{b}_0, ..., \mathbf{b}_n$, but also in the convex hull of $\mathbf{b}_0, \mathbf{f}_0, ..., \mathbf{f}_{n-1}, \mathbf{b}_n$.

A comprehensive introduction of rational Bézier curves and Farin points (weight points) can be found in [54].

Considering the above-mentioned properties of Farin points for Bézier curves, it is a natural next step to consider Farin points for rational Bézier surfaces. Unfortunately, a straightforward extension to surfaces creates problems because the independence property gets lost. As we will see in the next sections, Farin points for Bézier surfaces (both triangular and tensorproduct) overdefine the weights and therefore have dependencies to each other. In other words: moving a particular Farin point of a Bézier surface may cause contradictions in the system of all Farin points. A Farin point is no longer freely movable. An important design feature gets lost.

The solutions to this problem which we want to present in the following use ideas and concepts from the visualization of multiparameter data. As mentioned in section 5.1.2, for visualizing multivariate data using iconic techniques, it is crucial to choose an appropriate icon. The problem of finding appropriate

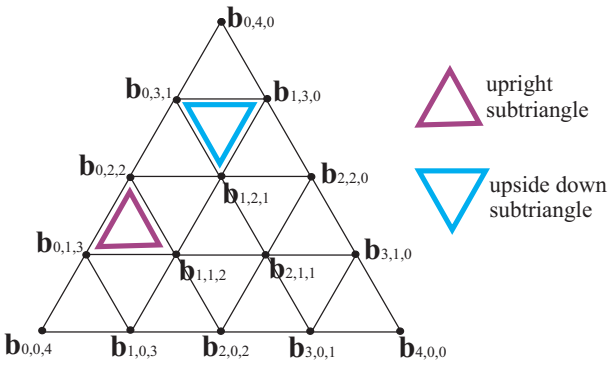


Figure 7.3: Bézier triangle of degree 4 consisting of 15 Bézier points; this Bézier triangle consist of 10 "upright" subtriangles and 6 "upside down" subtriangles.

schemes of Farin points is an analogous problem to the problem of finding appropriate icons. In particular the requirement to have a one-to-one correspondence between all degrees of freedom of the weights and the system of Farin points has its counterpart in the requirement that a designed icon for the visualization of multiparameter data should describe all dimensions completely and free of contradictions. This makes it possible to apply ideas and methods of the design of icons for multiparameter data.

Following [189], section 7.1.1 introduces subsets of Farin points for triangular Bézier surfaces where the Farin points in these subsets are independently of each other and cover all degrees of freedom. Section 7.1.2 does the same for tensorproduct surfaces.

Although we formulate the approaches in the following sections 7.1.1 and 7.1.2 only for rational Bézier surfaces, they are applicable for rational B-spline surfaces as well.

Notation: For describing basic geometric constructions in Euclidian space we use the following pseudo-code:

$\mathbf{l} := \mathrm{lin}(\mathbf{a}, \mathbf{b})$: let $\mathbf{l}$ be the line through the points $\mathbf{a}$ and $\mathbf{b}$.

$\mathbf{l} := \mathrm{par}(\mathbf{l}_0, \mathbf{a})$: let $\mathbf{l}$ be the line parallel to the line $\mathbf{l}_0$ through $\mathbf{a}$.

$\mathbf{a} := \mathrm{int}(\mathbf{l}_1, \mathbf{l}_2)$: let $\mathbf{a}$ be the intersection point of the lines $\mathbf{l}_1$ and $\mathbf{l}_2$.

Furthermore we need the concept of cross ratio for 4 collinear points. We use the definition

$$\mathrm{cr}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \frac{\mathrm{ratio}(\mathbf{a}, \mathbf{b}, \mathbf{d})}{\mathrm{ratio}(\mathbf{a}, \mathbf{c}, \mathbf{d})}. \tag{7.2}$$

7.1.1 Farin points for Bézier triangles

A triangular Bézier surface of degree n is given by the Bézier points $\mathbf{b}_{i,j,k}$ with $0 \leq i, j, k \leq n$ and $i + j + k = n$. These Bézier points are arranged in a triangular scheme as illustrated in figure 7.3 for $n = 4$. This scheme is called Bézier triangle. Inside a Bézier triangle there are a number of upright subtriangles consisting of 3 adjacent Bézier points $\mathbf{b}_{i,j,k}$, $\mathbf{b}_{i+1,j,k-1}$, $\mathbf{b}_{i,j-1,k-1}$. Figure 7.3 illustrates an upright subtriangle.

For handling Bézier triangles, usually only the upright subtriangles are

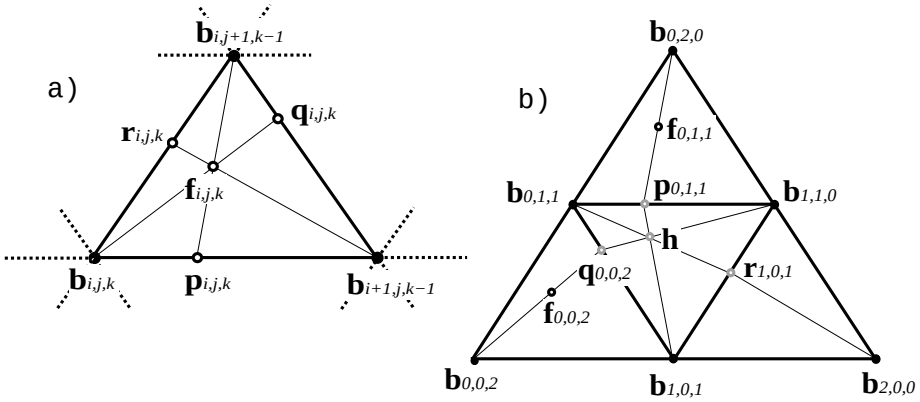


Figure 7.4: a) Farin points $\mathbf{f}_{i,j,k}$, $\mathbf{p}_{i,j,k}$, $\mathbf{q}_{i,j,k}$, $\mathbf{r}_{i,j,k}$ for the Bézier subtriangle $\mathbf{b}_{i,j,k}$, $\mathbf{b}_{i+1,j,k-1}$, $\mathbf{b}_{i,j+1,k-1}$; b) Bézier triangle of degree 2. Given the Farin points $\mathbf{f}_{0,0,2}$, $\mathbf{f}_{0,1,1}$, the Farin point $\mathbf{f}_{1,0,1}$ is not freely movable. In fact $\mathbf{f}_{1,0,1}$ must lie on $\mathrm{lin}(\mathbf{r}_{1,0,1}, \mathbf{b}_{2,0,0})$.

treated (see for instance the triangular de Casteljau scheme ([54])). However, it turns out that for the approaches we want introduce here it is also necessary to consider "upside down" subtriangles. Such a subtriangle is given by the three adjacent Bézier points $\mathbf{b}_{i,j,k}$, $\mathbf{b}_{i+1,j,k-1}$, $\mathbf{b}_{i+1,j-1,k}$. An "upside down" subtriangle is also illustrated in figure 7.3.

Given a Bézier triangle, the ratios of the (positive) weights of a subtriangle can be described by one Farin point inside this subtriangle. For a subtriangle consisting of the Bézier points $\mathbf{b}_{i,j,k}$, $\mathbf{b}_{i+1,j,k-1}$, $\mathbf{b}_{i,j+1,k-1}$ with the assigned weights $w_{i,j,k}$, $w_{i+1,j,k-1}$, $w_{i,j+1,k-1}$, we define the Farin point as barycentric combination of the 3 Bézier points:

$$\mathbf{f}_{i,j,k} = \frac{w_{i,j,k} \cdot \mathbf{b}_{i,j,k} \; + \; w_{i+1,j,k-1} \cdot \mathbf{b}_{i+1,j,k-1} \; + \; w_{i,j+1,k-1} \cdot \mathbf{b}_{i,j+1,k-1}}{w_{i,j,k} \; + \; w_{i+1,j,k-1} \; + \; w_{i,j+1,k-1}}. \quad (7.3)$$

For positive weights, $\mathbf{f}_{i,j,k}$ is inside the triangle $\mathbf{b}_{i,j,k}$, $\mathbf{b}_{i+1,j,k-1}$, $\mathbf{b}_{i,k+1,j-1}$. Obviously, the location of $\mathbf{f}_{i,j,k}$ in the triangle determines the ratio of the 3 weights uniquely. Furthermore we define the Farin points $\mathbf{p}_{i,j,k}$, $\mathbf{q}_{i,j,k}$, $\mathbf{r}_{i,j,k}$ on the edges of the triangle as determining the ratios of each two of the Bézier points:

$$\begin{aligned}
\mathbf{p}_{i,j,k} &= \frac{w_{i,j,k} \cdot \mathbf{b}_{i,j,k} \; + \; w_{i+1,j,k-1} \cdot \mathbf{b}_{i+1,j,k-1}}{w_{i,j,k} \; + \; w_{i+1,j,k-1}}, \\
\mathbf{q}_{i,j,k} &= \frac{w_{i+1,j,k-1} \cdot \mathbf{b}_{i+1,j,k-1} \; + \; w_{i,j+1,k-1} \cdot \mathbf{b}_{i,j+1,k-1}}{w_{i+1,j,k-1} \; + \; w_{i,j+1,k-1}}, \\
\mathbf{r}_{i,j,k} &= \frac{w_{i,j,k} \cdot \mathbf{b}_{i,j,k} \; + \; w_{i,j+1,k-1} \cdot \mathbf{b}_{i,j+1,k-1}}{w_{i,j,k} \; + \; w_{i,j+1,k-1}}.
\end{aligned}$$

The geometric correlation between $\mathbf{f}_{i,j,k}$, $\mathbf{p}_{i,j,k}$, $\mathbf{q}_{i,j,k}$, $\mathbf{r}_{i,j,k}$ is shown in figure 7.4a.
After Farin points for one subtriangle, we consider all Farin points of a Bézier triangle, i.e. the Farin points of all upright subtriangles of the Bézier triangle. Unfortunately, these Farin points are not independent of each other any more.

To show this, we consider the example of a Bézier triangle of degree 2 as shown in figure 7.4b. Suppose we know the Farin points $\mathbf{f}_{0,0,2}$ and $\mathbf{f}_{0,1,1}$. Then the following constructions give $\mathbf{r}_{1,0,1}$:

$$\begin{aligned}
\mathbf{q}_{0,0,2} &:= \text{int}(\text{lin}(\mathbf{b}_{1,0,1}, \mathbf{b}_{0,1,1}), \text{lin}(\mathbf{b}_{0,0,2}, \mathbf{f}_{0,0,2}))\ , \\
\mathbf{p}_{0,1,1} &:= \text{int}(\text{lin}(\mathbf{b}_{0,1,1}, \mathbf{b}_{1,1,0}), \text{lin}(\mathbf{b}_{0,2,0}, \mathbf{f}_{0,1,1}))\ , \\
\mathbf{h} &:= \text{int}(\text{lin}(\mathbf{p}_{0,1,1}, \mathbf{b}_{1,0,1}), \text{lin}(\mathbf{q}_{0,0,2}, \mathbf{b}_{1,1,0}))\ , \\
\mathbf{r}_{1,0,1} &:= \text{int}(\text{lin}(\mathbf{b}_{1,0,1}, \mathbf{b}_{1,1,0}), \text{lin}(\mathbf{b}_{0,1,1}, \mathbf{h})).
\end{aligned}$$

The Farin point $\mathbf{f}_{1,0,1}$ must lie on $\text{lin}(\mathbf{r}_{1,0,1}, \mathbf{b}_{2,0,0})$. It is not freely movable any more. If $\mathbf{f}_{1,0,1}$ is not on $\text{lin}(\mathbf{r}_{1,0,1}, \mathbf{b}_{2,0,0})$, the system of the Farin points $\mathbf{f}_{0,0,0}$, $\mathbf{f}_{0,1,1}$, $\mathbf{f}_{1,0,1}$ is not contradiction-free. $\mathbf{f}_{0,0,2}$, $\mathbf{f}_{0,1,1}$ and $\mathbf{f}_{1,0,1}$ are not independent of each other.

To overcome this problem (and therefore make Farin points on Bézier triangles usable as a design tool) there are two approaches:

1. Allow the user to move every Farin point and adjust the adjacent Farin points simultaneously such that the system of all Farin points stays contradiction-free.

2. Do not offer the user all Farin points to move, but only a certain subset. The Farin points in this subset should be independent of each other and describe all weights of the Bézier points uniquely (except for a common factor).

In the following we present solutions for both approaches. Approach 1 is treated in section 7.1.1.1, section 7.1.1.2 shows solutions for approach 2.

7.1.1.1 Adjusting adjacent Farin points

Given a triangular Bézier point scheme, we consider all Farin points $\mathbf{f}_{i,j,k}$ of upright Bézier subtriangles. As we know from the example in figure 7.4b, these Farin points are not independent of each other.

Figure 7.5 shows a number of subtriangles from a Bézier triangles. In particular, figure 7.5 shows the subtriangle $\mathbf{b}_{0,0,0}, \mathbf{b}_{1,0,-1}, \mathbf{b}_{0,1,-1}$ together with its 6 adjacent upright subtriangles[1]. Suppose the system of all Farin points $\mathbf{f}_{i,j,k}$ is contradiction-free. After moving $\mathbf{f}_{0,0,0}$ to $\widetilde{\mathbf{f}_{0,0,0}}$, the system of Farin points is generally not contradiction-free any more. In order to preserve the freedom of contradictions, we adjust the adjacent Farin points

$$\mathbf{f}_{0,-1,1}\ ,\ \mathbf{f}_{1,-1,0}\ ,\ \mathbf{f}_{1,0,-1}\ ,\ \mathbf{f}_{0,1,-1}\ ,\ \mathbf{f}_{-1,0,1}\ ,\ \mathbf{f}_{-1,1,0}$$

to the new points

$$\widetilde{\mathbf{f}_{0,-1,1}}\ ,\ \widetilde{\mathbf{f}_{1,-1,0}}\ ,\ \widetilde{\mathbf{f}_{1,0,-1}}\ ,\ \widetilde{\mathbf{f}_{0,1,-1}}\ ,\ \widetilde{\mathbf{f}_{-1,0,1}}\ ,\ \widetilde{\mathbf{f}_{-1,1,0}}.$$

[1]Some Bézier points in figure 7.5 have negative indices which were not allowed in the definition of Bézier triangles. This is due to the fact that we applied an index transformation to keep them simple. This way the indices in figure 7.5 denote the relative indices to the index (i, j, k). For instance, $\mathbf{b}_{1,0,-1}$ means the Bezier point $\mathbf{b}_{i+1,j,k-1}$.

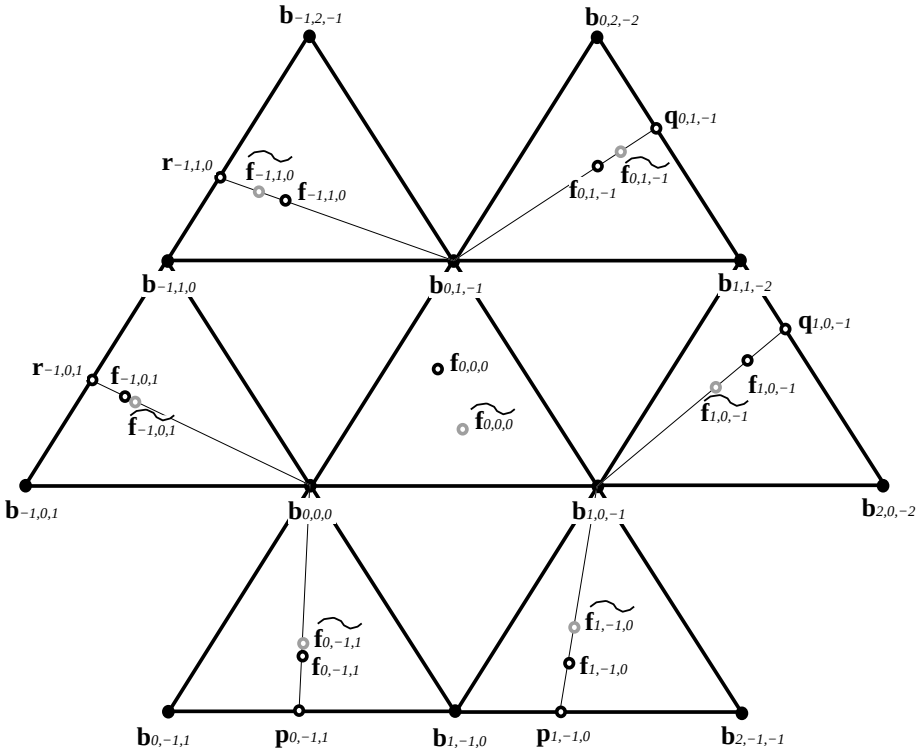


Figure 7.5: Moving the Farin point $\mathbf{f}_{0,0,0}$ to $\widetilde{\mathbf{f}_{0,0,0}}$. In order to preserve the freedom of contradictions, the adjacent Farin points $\mathbf{f}_{0,1,-1}$, $\mathbf{f}_{-1,1,0}$, $\mathbf{f}_{-1,0,1}$, $\mathbf{f}_{0,-1,1}$, $\mathbf{f}_{1,-1,0}$ have to be adjusted to $\widetilde{\mathbf{f}_{0,1,-1}}$, $\widetilde{\mathbf{f}_{-1,1,0}}$, $\widetilde{\mathbf{f}_{-1,0,1}}$, $\widetilde{\mathbf{f}_{0,-1,1}}$, $\widetilde{\mathbf{f}_{1,-1,0}}$.

We know that $\widetilde{\mathbf{f}_{0,1,-1}}$ lies on $\mathrm{lin}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{0,1,-1})$. The similar statements for the other Farin points can be seen in figure 7.5. Furthermore, it can be shown[2] that

$$\begin{aligned}
\mathrm{cr}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{0,1,-1}, \widetilde{\mathbf{f}_{0,1,-1}}, \mathbf{q}_{0,1,-1}) &= \mathrm{cr}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{-1,1,0}, \widetilde{\mathbf{f}_{-1,1,0}}, \mathbf{r}_{-1,1,0}) \,, \\
\mathrm{cr}(\mathbf{b}_{0,0,0}, \mathbf{f}_{-1,0,1}, \widetilde{\mathbf{f}_{-1,0,1}}, \mathbf{r}_{-1,0,1}) &= \mathrm{cr}(\mathbf{b}_{0,0,0}, \mathbf{f}_{0,-1,1}, \widetilde{\mathbf{f}_{0,-1,1}}, \mathbf{p}_{0,-1,1}) \,, \\
\mathrm{cr}(\mathbf{b}_{1,0,-1}, \mathbf{f}_{1,-1,0}, \widetilde{\mathbf{f}_{1,-1,0}}, \mathbf{p}_{1,-1,0}) &= \mathrm{cr}(\mathbf{b}_{1,0,-1}, \mathbf{f}_{1,0,-1}, \widetilde{\mathbf{f}_{1,0,-1}}, \mathbf{q}_{1,0,-1}) \,.
\end{aligned}$$

All we have to do now is to determine these cross ratios. For doing this, consider figure 7.6 which is a fragment of figure 7.5. We constructed the auxiliary point $\mathbf{c}$ in the following way:

$$\begin{aligned}
\mathbf{h} &:= \mathrm{int}(\mathrm{par}(\mathrm{lin}(\mathbf{b}_{0,0,0}, \mathbf{b}_{1,0,-1}), \widetilde{\mathbf{f}_{0,0,0}}), \mathrm{par}(\mathrm{lin}(\mathbf{b}_{0,0,0}, \mathbf{b}_{0,1,-1}), \mathbf{f}_{0,0,0})) \,, \\
\mathbf{c} &:= \mathrm{int}(\mathrm{lin}(\mathbf{b}_{0,0,0}, \mathbf{h}), \mathrm{lin}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{0,0,0})) \,.
\end{aligned}$$

Then it can be shown[3] that

$$\begin{aligned}
\mathrm{cr}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{0,0,0}, \mathbf{c}, \mathbf{p}_{0,0,0}) &= \mathrm{cr}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{0,1,-1}, \widetilde{\mathbf{f}_{0,1,-1}}, \mathbf{q}_{0,1,-1}) \\
&= \mathrm{cr}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{-1,1,0}, \widetilde{\mathbf{f}_{-1,1,0}}, \mathbf{r}_{-1,1,0}).
\end{aligned}$$

[2] by a direct algebraic computation using a formula manipulation program.
[3] by direct algebraic computation.

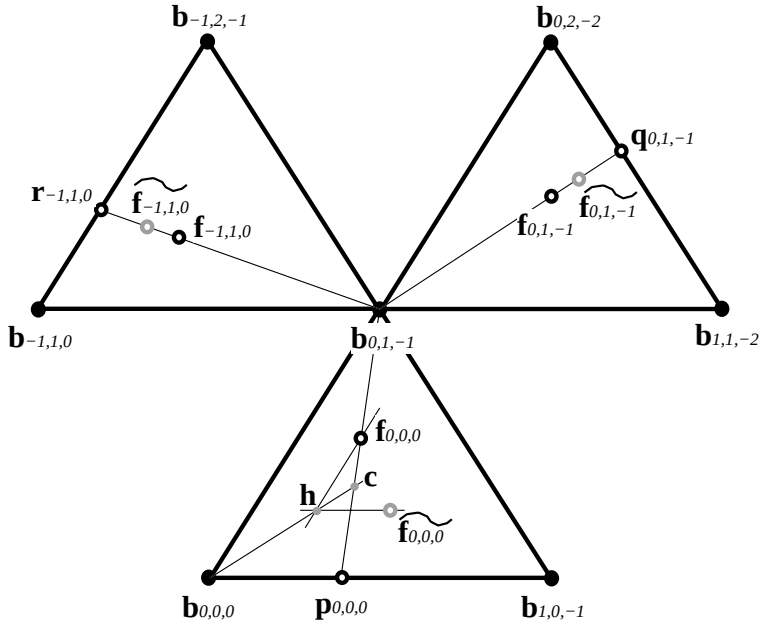


Figure 7.6: Moving the Farin point $\mathbf{f}_{0,0,0}$ to $\widetilde{\mathbf{f}_{0,0,0}}$ and adjusting the adjacent Farin points. Constructing the auxiliary point $\mathbf{c}$, we obtain: $\mathrm{cr}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{-1,1,0}, \widetilde{\mathbf{f}_{-1,1,0}}, \mathbf{r}_{-1,1,0}) = \mathrm{cr}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{0,1,-1}, \widetilde{\mathbf{f}_{0,1,-1}}, \mathbf{q}_{0,1,-1}) = \mathrm{cr}(\mathbf{b}_{0,1,-1}, \mathbf{f}_{0,0,0}, \mathbf{c}, \mathbf{p}_{0,0,0})$.

Therefore, the new adjusted Farin points $\widetilde{\mathbf{f}_{0,1,-1}}$ and $\widetilde{\mathbf{f}_{-1,1,0}}$ can be geometrically constructed. Similar constructions apply for the other adjacent Farin points.

7.1.1.2 Independent Farin points for Bézier triangles

In this section we want to establish a subset of independent Farin points for Bézier triangles which describes the weights of the Bézier points uniquely (except for a common factor). The first solution of this was published in [3]. There the problem was reduced to the curve problem by using not the Farin points $\mathbf{f}_{i,j,k}$ in the subtriangles but the points $\mathbf{p}_{i,j,k}$, $\mathbf{q}_{i,j,k}$, $\mathbf{r}_{i,j,k}$ on the edges of the subtriangles (see figure 7.4a). A system of these points - each of them movable on a line segment - gave the solution. This solution was obtained by searching a spanning tree for the control net which is interpreted as a graph.

In this section we use basic ideas of [3] but present a subset of independent Farin points $\mathbf{f}_{i,j,k}$, i.e. these Farin points are freely movable inside a whole subtriangle. Therefore we need fewer Farin points than in [3]. It turns out that in order to provide such a subset of Farin points, we have to consider not only upright subtriangles but "upside down" subtriangles as well.

First we have to make sure that such a system of independent Farin points exists at all. In doing this we keep in mind that a Farin point $\mathbf{f}_{i,j,k}$ is movable in 2D, and therefore its location covers 2 degrees of freedom.

The number d_f of degrees of freedom we have to cover in a triangular Bézier

n	1	2	3	4	5	6	7	...
$d_f(n)$	2	5	9	14	20	27	35	...
	even	odd	odd	even	even	odd	odd	...

Table 7.1: Number $d_f(n)$ of degrees of freedom to be covered by Farin points in triangular Bézier point schemes of the order n.

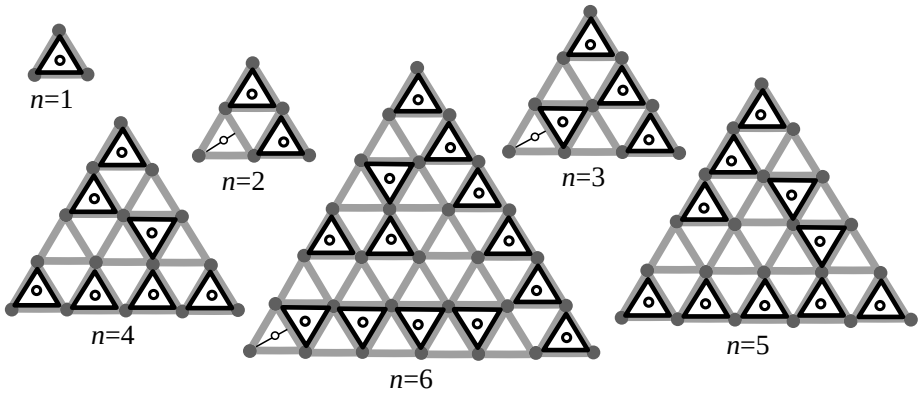


Figure 7.7: Schemes of independent Farin points for Bézier triangles of degree 1-6, strategy (α). Marked are all subtriangles in which the Farin point is considered. For $n = 2, 3, 6$, the weights of one corner Bézier point was fixed by introducing a Farin point on a line segment.

point scheme of the order n is:

$$d_f(n) = \frac{(n+1)\cdot(n+2)}{2} - 1. \tag{7.4}$$

This means that $d_f(n)$ is the number of the Bézier points minus 1. We have to fix the weights of all Bézier points except for a common factor. One weight can be chosen randomly, then the other weights are fixed. Table 7.1 shows the number $d_f(n)$ of degrees of freedom for small n. If $d_f(n)$ is even, we can find a system of $d_f(n)/2$ independent Farin points. For $d_f(n)$ is odd, there are 2 strategies:

(α) Find $\frac{d_f(n)-1}{2}$ independent Farin points and determine the weight of one corner point of the Bézier triangle explicitly.

(β) Find $\frac{d_f(n)-3}{2}$ independent Farin points and determine the weights of all 3 corner points of the Bézier triangle explicitly. This strategy has the advantage of preserving symmetry in the triangular scheme.

Figure 7.7 shows the solution for strategy (α) for small n. In this figure, all subtriangles for which a Farin point is used are marked. For $n = 2, 3, 6$ we have $d_f(n)$ odd. For the special treatment of the lower left corners we introduced Farin points on line segments which relate the weight of the corner to the average of the weights of the other two Bézier points in the subtriangle. Figure 7.8 shows how to get the solution for degree $n+6$ for a given solution for degree n. Figure 7.7 and 7.8 give the solution for strategy (α) for any degree n by induction.

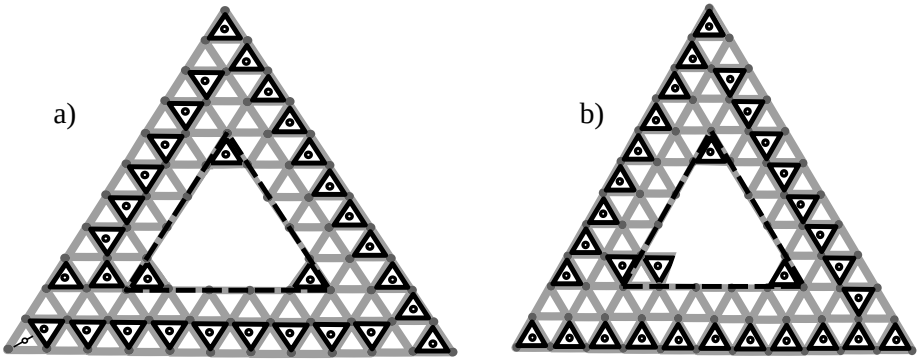


Figure 7.8: Obtaining the system of independent Farin points for the degree $n+6$ from a given solution for degree n, using strategy (α): a) for $d_f(n)$ even, b) for $d_f(n)$ odd.

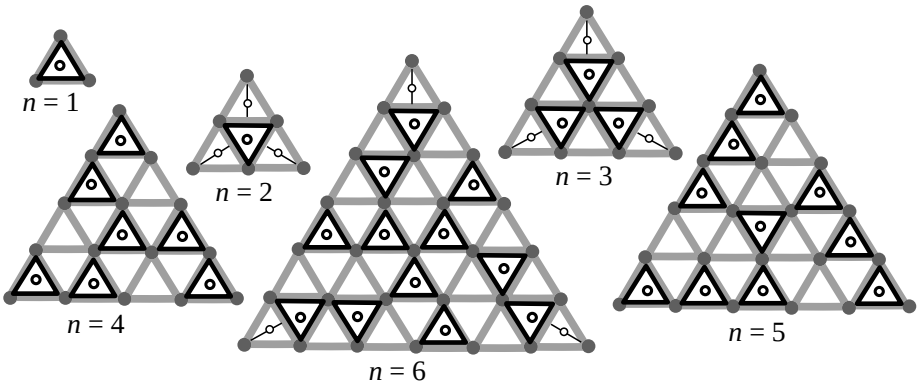


Figure 7.9: Schemes of independent Farin points for Bézier triangles of degree 1-6, strategy (β).

Figure 7.9 and 7.10 show the solution for strategy (β) by induction.

Figure 7.11 shows an example: the system of independent Farin points for a triangular Bézier point scheme of order 13.

Remarks:

- The systems of Farin points introduced in this section have the independence property but *not* the local control property. This means that moving one of the marked Farin points might change any other unmarked Farin point. If the local control property of Farin points is important we have to apply the automatic adjusting introduced in section 7.1.1.1. A system with both the independence property and the local control property seems not to exist.

- All schemes and constructions introduced in section 7.1.1 work both in the domain of the Bézier triangles and in the Bézier point scheme in 3D. This is because the Bézier point subtriangles in 3D can be considered as affine maps of the subtriangles in the domain, and all constructions are

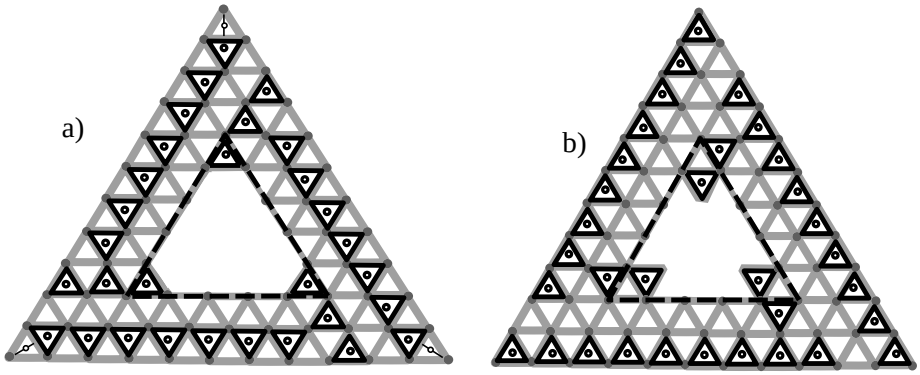


Figure 7.10: Obtaining the system of independent Farin points for the degree $n+6$ from a given solution for degree n, using strategy (β): a) for $d_f(n)$ even, b) for $d_f(n)$ odd.

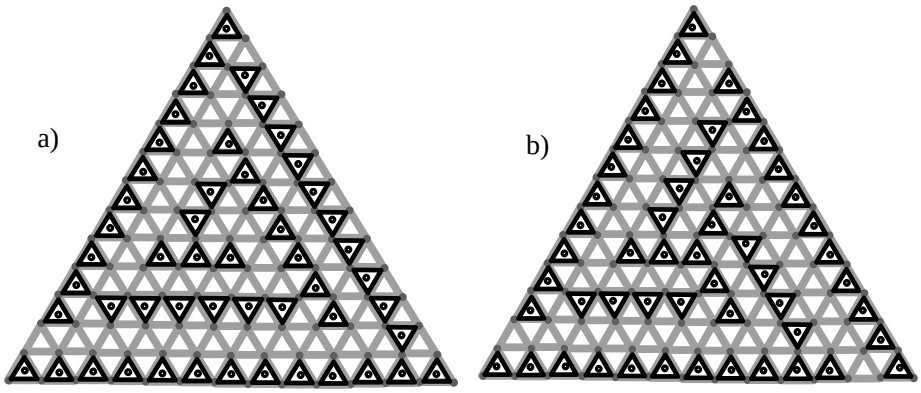


Figure 7.11: System of independent Farin points for $n = 13$; a) using the strategy (α); b) using the strategy (β).

affine invariant.

7.1.2 Farin points for tensorproduct Bézier surfaces

In this section we apply the concept of Farin points to tensorproduct surfaces which are described by a rectangular Bézier point scheme. To handle the weights of such a scheme by Farin points, we first have to solve the problem for a subquadrilateral. We look for a Farin point solution for a subquadrilateral both in the domain of the surface and in 3D.

For the domain case, the Bézier points $\mathbf{b}_{i,j}$, $\mathbf{b}_{i+1,j}$, $\mathbf{b}_{i+1,j+1}$, $\mathbf{b}_{i,j+1}$ form a rectangle, see figure 7.12a. Then we can define the points on the edges of the rectangle which define the ratios of each two of the weights:

$$\begin{aligned} \mathbf{p}_{i,j} &= \frac{w_{i,j} \cdot \mathbf{b}_{i,j} \ + \ w_{i+1,j} \cdot \mathbf{b}_{i+1,j}}{w_{i,j} \ + \ w_{i+1,j}} \ , \\ \mathbf{p}_{i,j+1} &= \frac{w_{i,j+1} \cdot \mathbf{b}_{i,j+1} \ + \ w_{i+1,j+1} \cdot \mathbf{b}_{i+1,j+1}}{w_{i,j+1} \ + \ w_{i+1,j+1}} \ , \end{aligned}$$

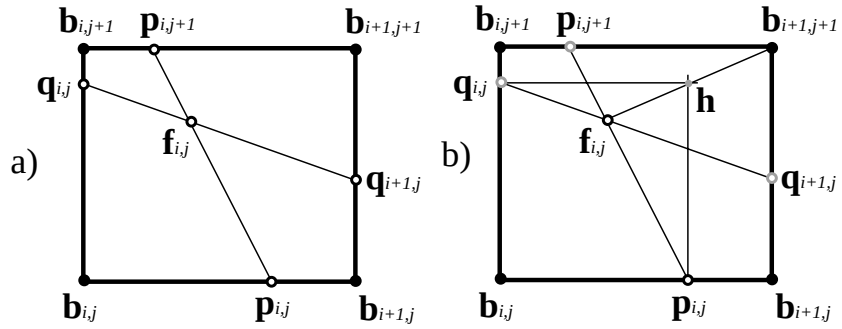


Figure 7.12: a) Defining the points $\mathbf{p}_{i,j}$, $\mathbf{p}_{i,j+1}$, $\mathbf{q}_{i,j}$, $\mathbf{q}_{i+1,j}$ and the Farin point $\mathbf{f}_{i,j}$ for the Bézier points $\mathbf{b}_{i,j}$, $\mathbf{b}_{i+1,j}$, $\mathbf{b}_{i+1,j+1}$, $\mathbf{b}_{i,j+1}$ assigned with the weights $w_{i,j}$, $w_{i+1,j}$, $w_{i+1,j+1}$, $w_{i,j+1}$. b) constructing $\mathbf{q}_{i,j}$, $\mathbf{q}_{i+1,j}$ and $\mathbf{p}_{i,j+1}$ from the given points $\mathbf{f}_{i,j}$ and $\mathbf{p}_{i,j}$ geometrically.

$$\begin{aligned}\mathbf{q}_{i,j} &= \frac{w_{i,j}\cdot\mathbf{b}_{i,j} \;+\; w_{i,j+1}\cdot\mathbf{b}_{i,j+1}}{w_{i,j} \;+\; w_{i,j+1}}\,,\\ \mathbf{q}_{i+1,j} &= \frac{w_{i+1,j}\cdot\mathbf{b}_{i+1,j} \;+\; w_{i+1,j+1}\cdot\mathbf{b}_{i+1,j+1}}{w_{i+1,j} \;+\; w_{i+1,j+1}}\,.\end{aligned}$$

Furthermore we define the Farin point

$$\begin{aligned}\mathbf{f}_{i,j} &= \mathrm{int}(\mathrm{lin}(\mathbf{p}_{i,j},\mathbf{p}_{i,j+1}),\mathrm{lin}(\mathbf{q}_{i,j},\mathbf{q}_{i+1,j}))\\ &= \frac{w_{i,j}\cdot\mathbf{b}_{i,j} \;+\; w_{i+1,j}\cdot\mathbf{b}_{i+1,j} \;+\; w_{i+1,j+1}\cdot\mathbf{b}_{i+1,j+1} \;+\; w_{i,j+1}\cdot\mathbf{b}_{i,j+1}}{w_{i,j} \;+\; w_{i+1,j} \;+\; w_{i+1,j+1} \;+\; w_{i,j+1}}\,.\end{aligned}$$

See figure 7.12a for an illustration. The Farin point $\mathbf{f}_{i,j}$ has the intuitivity property: increasing the weight of one Bézier point leads to moving $\mathbf{f}_{i,j}$ towards this Bézier point. Unfortunately, $\mathbf{f}_{i,j}$ is not sufficient to define uniquely the weight ratios in a subquadrilateral. In fact $\mathbf{f}_{i,j}$ is freely movable in 2D and therefore covers two degrees of freedom. What we have to determine are 3 degrees of freedom in a quadrilateral (the weight of one Bézier point can be chosen randomly, then the weights of the other 3 Bézier points have to be fixed).

Suppose the points $\mathbf{f}_{i,j}$ and $\mathbf{p}_{i,j}$ are given. Then the remaining points $\mathbf{p}_{i,j+1}$, $\mathbf{q}_{i,j}$ and $\mathbf{q}_{i+1,j}$ can be geometrically constructed in the following way:

$$\begin{aligned}\mathbf{p}_{i,j+1} &:= \mathrm{int}(\mathrm{lin}(\mathbf{p}_{i,j},\mathbf{f}_{i,j}),\mathrm{lin}(\mathbf{b}_{i,j+1},\mathbf{b}_{i+1,j+1}))\,,\\ \mathbf{h} &:= \mathrm{int}(\mathrm{lin}(\mathbf{f}_{i,j},\mathbf{b}_{i+1,j+1}),\mathrm{par}(\mathrm{lin}(\mathbf{b}_{i+1,j},\mathbf{b}_{i+1,j+1}),\mathbf{p}_{i,j}))\,,\\ \mathbf{q}_{i,j} &:= \mathrm{int}(\mathrm{lin}(\mathbf{b}_{i,j},\mathbf{b}_{i,j+1}),\mathrm{par}(\mathrm{lin}(\mathbf{b}_{i,j+1},\mathbf{b}_{i+1,j+1}),\mathbf{h}))\,,\\ \mathbf{q}_{i+1,j} &:= \mathrm{int}(\mathrm{lin}(\mathbf{q}_{i,j},\mathbf{f}_{i,j}),\mathrm{lin}(\mathbf{b}_{i+1,j},\mathbf{b}_{i+1,j+1}))\,.\end{aligned}$$

See figure 7.12b for an illustration.

For reasons of a simplified notation, we consider from now on the Bézier point subquadrilateral $\mathbf{b}_{0,0}$, $\mathbf{b}_{1,0}$, $\mathbf{b}_{1,1}$, $\mathbf{b}_{0,1}$. The points $\mathbf{p}_{0,0}$, $\mathbf{q}_{1,0}$, $\mathbf{p}_{0,1}$, $\mathbf{q}_{0,0}$ are not independent of each other. In fact, the location of three of these points determines the remaining one. To find a construction for this fact we consider a property illustrated in figure 7.13b: the three lines

$$\mathrm{lin}(\mathbf{b}_{0,0},\mathbf{b}_{1,1})\,,\ \mathrm{lin}(\mathbf{p}_{0,0},\mathbf{q}_{1,0})\,,\ \mathrm{lin}(\mathbf{q}_{0,0},\mathbf{p}_{0,1})$$

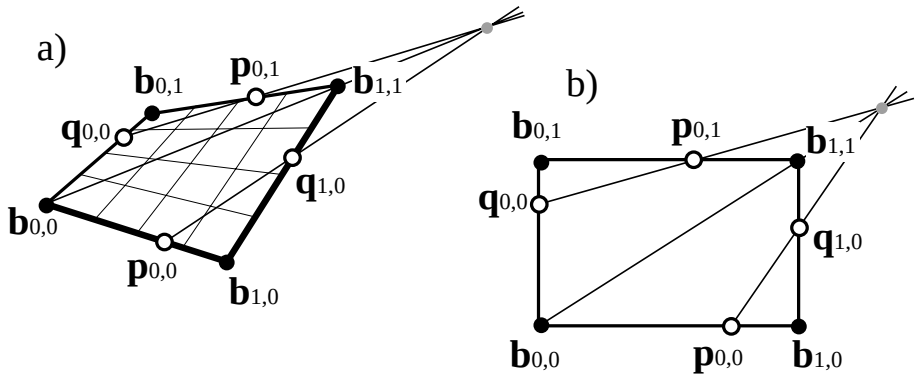


Figure 7.13: Weight points on the edges of a subquadrilateral: the lines $\text{lin}(\mathbf{b}_{0,0}, \mathbf{b}_{1,1})$, $\text{lin}(\mathbf{p}_{0,0}, \mathbf{q}_{1,0})$ and $\text{lin}(\mathbf{q}_{0,0}, \mathbf{p}_{0,1})$ either intersect at one point or are parallel. This is true both in the 3D case a) and in the domain case b).

either intersect at one point or are all parallel. As illustrated in figure 7.13a, the same property is true in 3D[4].

Given the Farin point $\mathbf{f}_{0,0}$ in the rectangular domain quadrilateral, we seek a way to determine all weight ratios. One way of doing this is to fix one of the weight points on the edges, as shown in figure 7.12b. This approach is not symmetric because we have to make the choice which point on the edges to fix. In order to introduce a symmetric method, we have to design an appropriate *icon* which covers all degrees of freedom. This icon will be based on the Farin point $\mathbf{f}_{0,0}$.

Consider figure 7.14a. We assume that the weights $w_{0,0}$, $w_{1,0}$, $w_{1,1}$, $w_{0,1}$ which are assigned to the Bézier points $\mathbf{b}_{0,0}$, $\mathbf{b}_{0,1}$, $\mathbf{b}_{1,1}$, $\mathbf{b}_{1,0}$ are all positive. This means that the point $\mathbf{p}_{0,0}$ lies between $\mathbf{b}_{0,0}$ and $\mathbf{b}_{1,0}$, similar for the other three weight points on the edges. Now we seek all possible locations for $\mathbf{p}_{0,1}$ between $\mathbf{b}_{0,1}$ and $\mathbf{b}_{1,1}$, so that $\mathbf{p}_{0,0}$ is between $\mathbf{b}_{0,0}$ and $\mathbf{b}_{1,0}$. Obviously, this is the grey marked area on the line $\mathbf{b}_{0,1}$, $\mathbf{b}_{1,1}$. We call this area the *permitted area*.

The permitted area of an edge of the rectangle is the area where the weight point can lie so that the weight point of the opposite edge is between the corner points. In figure 7.14a, the permitted area of the edge $\mathbf{b}_{0,0}$, $\mathbf{b}_{1,0}$ is the whole line segment $\mathbf{b}_{0,0}$, $\mathbf{b}_{1,0}$: no matter where $\mathbf{p}_{0,0}$ is located, the opposite point $\mathbf{p}_{0,1}$ will be between $\mathbf{b}_{0,1}, \mathbf{b}_{1,1}$.

The location of the permitted areas on the edges depends on the location of $\mathbf{f}_{0,0}$. All permitted areas can be found by intersecting the four lines $\text{lin}(\mathbf{b}_{0,0}, \mathbf{f}_{0,0})$, $\text{lin}(\mathbf{b}_{1,0}, \mathbf{f}_{0,0})$, $\text{lin}(\mathbf{b}_{1,1}, \mathbf{f}_{0,0})$, $\text{lin}(\mathbf{b}_{0,1}, \mathbf{f}_{0,0})$ with the rectangle. See figure 7.14a for an illustration.

Now we divide the four permitted areas in the ratio $t/(1-t)$, as shown in figure 7.14b (with $0 < t < 1$). The resulting four points on the edges of the rectangle can be considered as the Farin points on the edges. It turns out that these four points have the property described in figure 7.13b. This means that we have described the weight ratios in the quadrilateral completely by $\mathbf{f}_{0,0}$

[4]From this property it follows directly that the points $\mathbf{p}_{0,0}$, $\mathbf{q}_{1,0}$, $\mathbf{p}_{0,1}$, $\mathbf{q}_{0,0}$ are coplanar (see [54]).

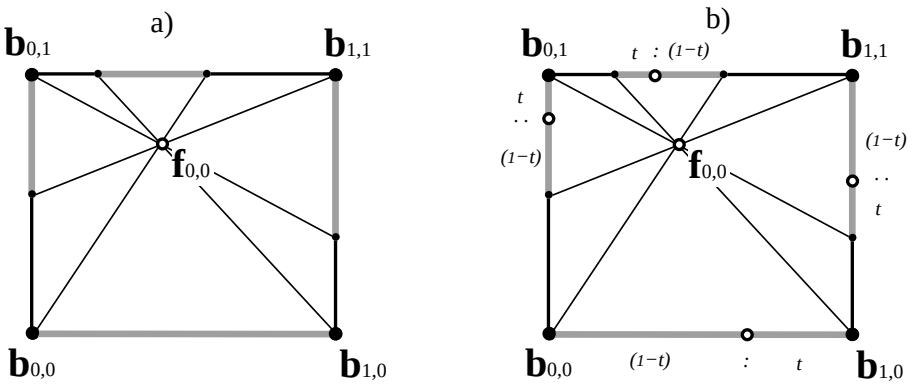


Figure 7.14: a) Constructing permitted areas (grey marked) on the edges of the rectangle by intersecting the lines $\mathrm{lin}(\mathbf{b}_{0,0}, \mathbf{f}_{0,0})$, $\mathrm{lin}(\mathbf{b}_{1,0}, \mathbf{f}_{0,0})$, $\mathrm{lin}(\mathbf{b}_{1,1}, \mathbf{f}_{0,0})$ and $\mathrm{lin}(\mathbf{b}_{0,1}, \mathbf{f}_{0,0})$ with the rectangle. b) Dividing the permitted areas in the ratio $t/(1-t)$. We obtain the weight points on the edges of the quadrilateral.

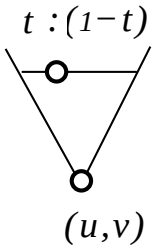


Figure 7.15: Extended Farin point: freely movable icon in the rectangular domain. The location of the lower hollow point gives $\mathbf{f}_{0,0}$. The upper hollow point determines the parameter t.

and t. To find a geometric meaning of t, we consider figure 7.14b again. The more t tends to 0, the closer the weight points tend to $\mathbf{b}_{0,1}$ or $\mathbf{b}_{1,0}$. The more t tends to 1, the closer the weight points tend to $\mathbf{b}_{0,0}$ or $\mathbf{b}_{1,1}$. This means that the parameter t is a measure which diagonal $\mathbf{b}_{0,0}, \mathbf{b}_{1,1}$ or $\mathbf{b}_{0,1}, \mathbf{b}_{1,0}$ is more emphasized. This gives a reason to introduce *extended Farin points*.

An extended Farin point is a freely movable icon as shown in figure 7.15. The location of the lower hollow point is the location of the Farin point $\mathbf{f}_{0,0}$. The upper hollow point is freely movable on the horizontal line segment and fixes which of the diagonals $\mathbf{b}_{1,0}, \mathbf{b}_{0,1}$ or $\mathbf{b}_{0,0}, \mathbf{b}_{1,1}$ is more emphasized. The diagonals are symbolized by the non-horizontal lines.

An extended Farin point describes the weight ratios of the Bézier points in a subquadrilateral uniquely and symmetrically: no particular edge or corner of the rectangle has to be chosen. The conditions of independence and intuitivity are also fulfilled by extended Farin points. An example of the usage of an extended Farin point is given in figure 7.16.

Farin points in 3D: Up to here, this chapter has treated Farin points in the rectangular domain of a quadrilateral. Now we consider 4 Bézier points $\mathbf{b}_{0,0}$, $\mathbf{b}_{1,0}$, $\mathbf{b}_{1,1}$, $\mathbf{b}_{0,1}$ as living in the 3D space of the surfaces. The straightforward

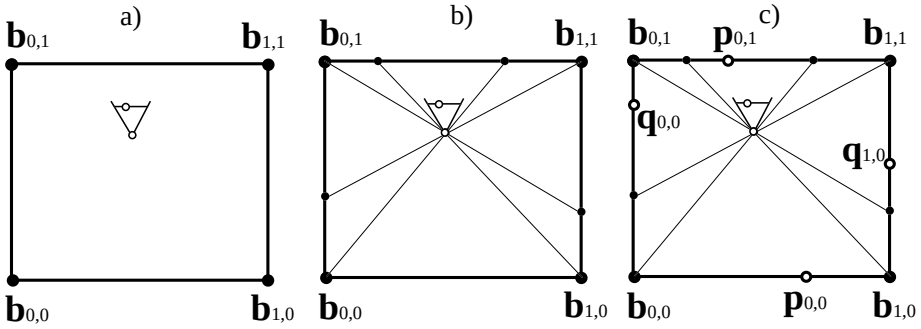


Figure 7.16: Usage of extended Farin points. a) Given is Bézier point rectangle $\mathbf{b}_{0,0}, \mathbf{b}_{1,0}, \mathbf{b}_{1,1}, \mathbf{b}_{0,1}$ and the extended Farin point inside the rectangle. b) Construct the permitted areas on the edges of the rectangle. c) Divide the permitted areas in the same ratio as the upper hollow point divides the horizontal line segment of the extended Farin point. The resulting four weight points $\mathbf{p}_{0,0}, \mathbf{q}_{1,0}, \mathbf{p}_{0,1}, \mathbf{q}_{0,0}$ define the ratios of the weights of the four Bézier points uniquely.

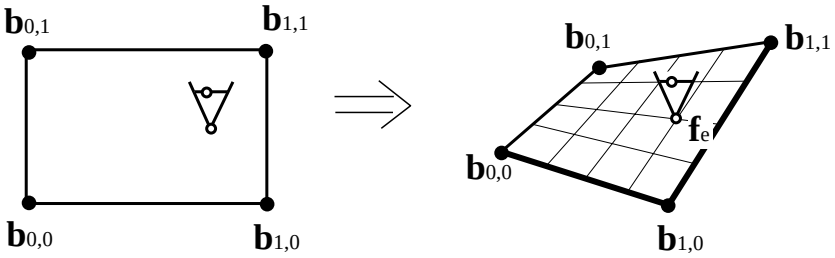


Figure 7.17: The extended Farin point in the 2D domain is mapped onto the bilinear interpolant of $\mathbf{b}_{0,0}, \mathbf{b}_{1,0}, \mathbf{b}_{1,1}, \mathbf{b}_{0,1}$ in 3D.

approach here to determine the weight ratios is to consider the barycentric combination

$$\mathbf{f} = \frac{w_{0,0} \cdot \mathbf{b}_{0,0} + w_{1,0} \cdot \mathbf{b}_{1,0} + w_{1,1} \cdot \mathbf{b}_{1,1} + w_{0,1} \cdot \mathbf{b}_{0,1}}{w_{0,0} + w_{1,0} + w_{1,1} + w_{0,1}}. \tag{7.5}$$

For positive weights, $\mathbf{f}$ lies in the convex hull of $\mathbf{b}_{0,0}$, $\mathbf{b}_{1,0}$, $\mathbf{b}_{1,1}$,$\mathbf{b}_{0,1}$. In general $\mathbf{f}$ is movable in 3D and thus covers the 3 degrees of freedom we have to fix. Unfortunately, this approach fails completely if $\mathbf{b}_{0,0}$, $\mathbf{b}_{1,0}$, $\mathbf{b}_{1,1}$, $\mathbf{b}_{0,1}$ are coplanar, and it fails numerically if $\mathbf{b}_{0,0}$, $\mathbf{b}_{1,0}$, $\mathbf{b}_{1,1}$, $\mathbf{b}_{0,1}$ are almost coplanar. Since in practical applications the case of almost planar subquadrilaterals is common, we have to find another way to describe the weight ratios: we use the concept of the extended Farin points in 2D and map this onto the bilinear interpolant of $\mathbf{b}_{0,0}$, $\mathbf{b}_{1,0}$, $\mathbf{b}_{1,1}$, $\mathbf{b}_{0,1}$ in 3D. Figure 7.17 gives an illustration.

The extended Farin point in 3D is freely movable on the bilinear interpolant. The location $\mathbf{f}_e$ of the extended Farin point on the bilinear interpolant can be computed as

$$\mathbf{f}_e = \frac{\omega_{0,0} \cdot \mathbf{b}_{0,0} + \omega_{1,0} \cdot \mathbf{b}_{1,0} + \omega_{1,1} \cdot \mathbf{b}_{1,1} + \omega_{0,1} \cdot \mathbf{b}_{0,1}}{\omega_{0,0} + \omega_{1,0} + \omega_{1,1} + \omega_{0,1}} \tag{7.6}$$

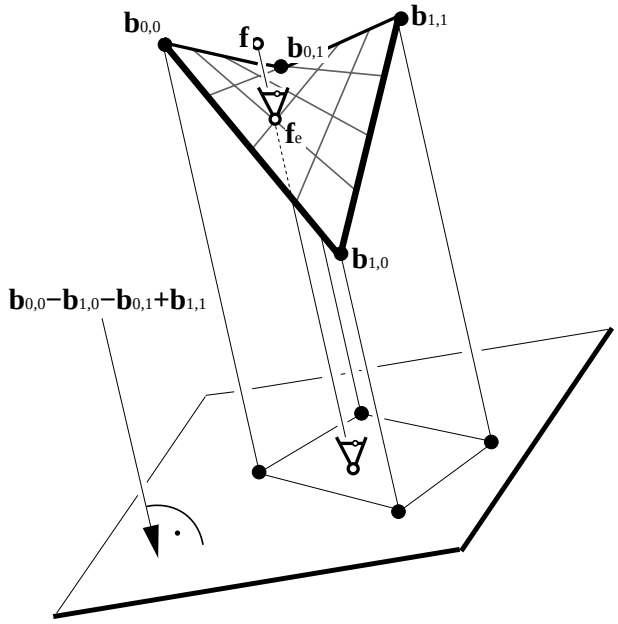


Figure 7.18: Getting the weight ratios from an extended Farin point $\mathbf{f}_e$ on the bilinear interpolation: apply a projection in the twist vector direction $\mathbf{b}_{0,0} - \mathbf{b}_{1,0} - \mathbf{b}_{0,1} + \mathbf{b}_{1,1}$. The projection of the 4 Bézier points gives a parallelogram inside which we can apply all constructions for extended Farin points in the 2D domain. Note that the projections of the location $\mathbf{f}_e$ of the extended Farin point and the barycentric combination $\mathbf{f}$ introduced in (7.5) are identical.

with

$$\omega_{0,0} = (w_{0,0} + w_{1,0}) \cdot (w_{0,0} + w_{0,1}) \quad , \quad \omega_{1,0} = (w_{1,0} + w_{0,0}) \cdot (w_{1,0} + w_{1,1}) \, ,$$
$$\omega_{1,1} = (w_{1,1} + w_{1,0}) \cdot (w_{1,1} + w_{0,1}) \quad , \quad \omega_{0,1} = (w_{0,1} + w_{0,0}) \cdot (w_{0,1} + w_{1,1}) \, .$$

In order to get the weight ratios out of $\mathbf{f}_e$, we apply a parallel projection in the direction of the twist vector $\mathbf{b}_{0,0} - \mathbf{b}_{1,0} - \mathbf{b}_{0,1} + \mathbf{b}_{1,1}$. Doing this the 4 Bézier points happen to form a parallelogram with the projection of $\mathbf{f}_e$ inside it. In this parallelogram we can carry out all constructions introduced in this chapter for the domain case. See figure 7.18 for an illustration.

7.1.2.1 Adjusting adjacent Farin points in rectangular Bézier point schemes

After showing how to handle the weights in a subquadrilateral by using the concept of extended Farin points, we now treat the case of a whole rectangular Bézier point scheme. As in the triangular case there are two ways to keep the system of all Farin points contradiction-free:

(a) Allow every Farin point to be movable and adjust the adjacent Farin points.

(b) Provide a subset of Farin points which are independent of each other and describe the weight ratios uniquely.

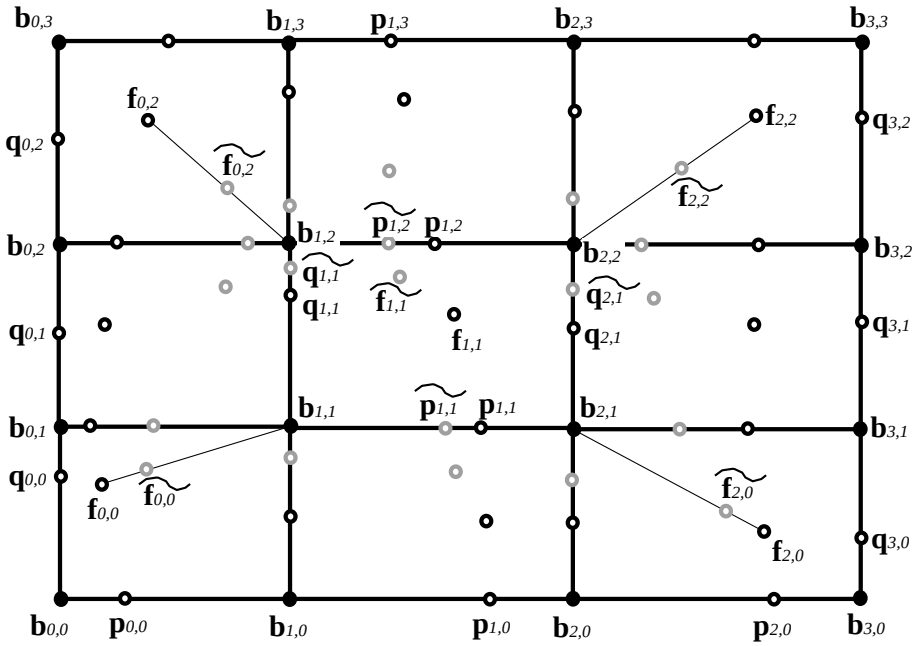


Figure 7.19: Moving the Farin points $\mathbf{f}_{1,1}$, $\mathbf{p}_{1,1}$, $\mathbf{q}_{2,1}$, $\mathbf{p}_{1,2}$, $\mathbf{q}_{1,1}$ to $\widetilde{\mathbf{f}_{1,1}}$, $\widetilde{\mathbf{p}_{1,1}}$, $\widetilde{\mathbf{q}_{2,1}}$, $\widetilde{\mathbf{p}_{1,2}}$, $\widetilde{\mathbf{q}_{1,1}}$. The adjacent Farin points have to be adjusted.

This section treats approach (a); a solution for (b) is given in section 7.1.2.2.

Consider figure 7.19 for a rectangular Farin point scheme in the domain. Suppose the weight ratios in the subquadrilateral $\mathbf{b}_{1,1}$, $\mathbf{b}_{2,1}$, $\mathbf{b}_{2,2}$, $\mathbf{b}_{1,2}$ are changed, for instance using the concepts of an extended Farin point. This means that the points $\mathbf{f}_{1,1}$, $\mathbf{p}_{1,1}$, $\mathbf{q}_{2,1}$, $\mathbf{p}_{1,2}$, $\mathbf{q}_{1,1}$ are moved to the new locations $\widetilde{\mathbf{f}_{1,1}}$, $\widetilde{\mathbf{p}_{1,1}}$, $\widetilde{\mathbf{q}_{2,1}}$,$\widetilde{\mathbf{p}_{1,2}}$, $\widetilde{\mathbf{q}_{1,1}}$. Then all Farin points which depend on at least one of the weights $w_{1,1}$, $w_{1,2}$, $w_{2,1}$, $w_{2,2}$, have to be adjusted: $\mathbf{f}_{0,0}$, $\mathbf{f}_{1,0}$, $\mathbf{f}_{2,0}$, $\mathbf{f}_{2,1}$, $\mathbf{f}_{2,2}$, $\mathbf{f}_{1,2}$, $\mathbf{f}_{0,2}$, $\mathbf{f}_{0,1}$, $\mathbf{q}_{1,0}$, $\mathbf{q}_{2,0}$, $\mathbf{p}_{2,1}$, $\mathbf{p}_{2,2}$, $\mathbf{q}_{2,2}$, $\mathbf{q}_{1,2}$, $\mathbf{p}_{0,2}$, $\mathbf{p}_{0,1}$. The points $\mathbf{p}_{0,0}$, $\mathbf{p}_{1,0}$, $\mathbf{p}_{2,0}$, $\mathbf{q}_{3,0}$, $\mathbf{q}_{3,1}$, $\mathbf{q}_{3,2}$, $\mathbf{p}_{2,3}$, $\mathbf{p}_{1,3}$, $\mathbf{p}_{0,3}$, $\mathbf{q}_{0,2}$, $\mathbf{q}_{0,1}$, $\mathbf{q}_{0,0}$ remain unchanged because they do not depend on the weights $w_{1,1}$, $w_{1,2}$, $w_{2,1}$, $w_{2,2}$.

All we have to show here is how to adjust the points $\mathbf{f}_{0,0}$, $\mathbf{f}_{2,0}$, $\mathbf{f}_{2,2}$,$\mathbf{f}_{0,2}$ to the points $\widetilde{\mathbf{f}_{0,0}}$, $\widetilde{\mathbf{f}_{2,0}}$,$\widetilde{\mathbf{f}_{2,2}}$,$\widetilde{\mathbf{f}_{0,2}}$. The other adjusted points can then be geometrically constructed using the properties described in the figures 7.12b and 7.13b.

The adjusted point $\widetilde{\mathbf{f}_{0,0}}$ lies on $\mathrm{lin}(\mathbf{f}_{0,0}, \mathbf{b}_{1,1})$. Similarly, $\widetilde{\mathbf{f}_{2,0}}$ lies on $\mathrm{lin}(\mathbf{f}_{2,0}, \mathbf{b}_{2,1})$, $\widetilde{\mathbf{f}_{2,2}}$ lies on $\mathrm{lin}(\mathbf{f}_{2,2}, \mathbf{b}_{2,2})$, $\widetilde{\mathbf{f}_{2,0}}$ lies on $\mathrm{lin}(\mathbf{f}_{0,2}, \mathbf{b}_{1,2})$. We present a construction for $\widetilde{\mathbf{f}_{2,2}}$, the points $\widetilde{\mathbf{f}_{0,0}}$, $\widetilde{\mathbf{f}_{2,0}}$, $\widetilde{\mathbf{f}_{0,2}}$ can be constructed in a similar way.

Consider figure 7.20a. First we construct the auxiliary points

$$\mathbf{h}_1 = \frac{w_{3,2} \cdot \mathbf{b}_{3,2} + w_{3,3} \cdot \mathbf{b}_{3,3} + w_{2,3} \cdot \mathbf{b}_{2,3}}{w_{3,2} + w_{3,3} + w_{2,3}},$$

$$\mathbf{h}_2 = \frac{w_{1,1} \cdot \mathbf{b}_{1,1} + w_{2,1} \cdot \mathbf{b}_{2,1} + w_{1,2} \cdot \mathbf{b}_{1,2}}{w_{1,1} + w_{2,1} + w_{1,2}},$$

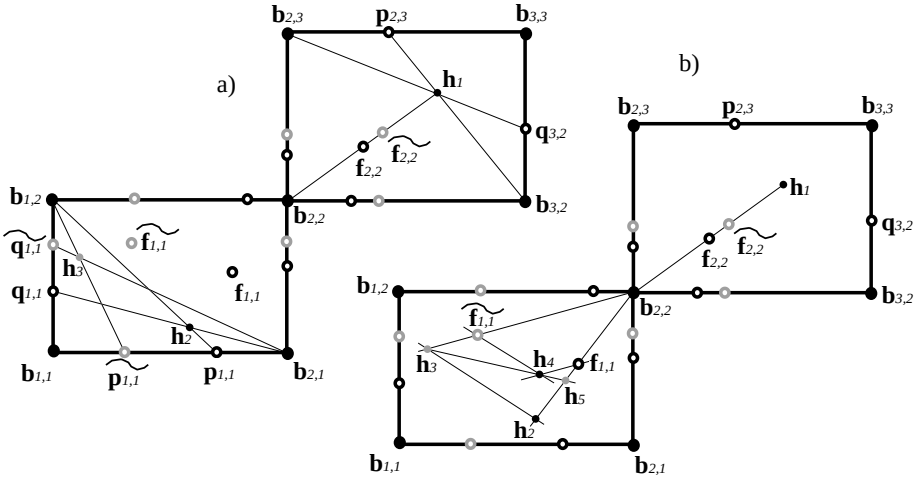


Figure 7.20: Moving the Farin points $\mathbf{f}_{1,1}$, $\mathbf{p}_{1,1}$, $\mathbf{q}_{2,1}$, $\mathbf{p}_{1,2}$, $\mathbf{q}_{1,1}$ to $\widetilde{\mathbf{f}_{1,1}}$, $\widetilde{\mathbf{p}_{1,1}}$, $\widetilde{\mathbf{q}_{2,1}}$, $\widetilde{\mathbf{p}_{1,2}}$, $\widetilde{\mathbf{q}_{1,1}}$: constructing the adjusted Farin point $\widetilde{\mathbf{f}_{2,2}}$. a) constructing the auxiliary points $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$; b) constructing the auxiliary points $\mathbf{h}_4, \mathbf{h}_5$. Then $\mathrm{cr}(\mathbf{b}_{2,2}, \mathbf{f}_{1,1}, \mathbf{h}_5, \mathbf{h}_2) = \mathrm{cr}(\mathbf{b}_{2,2}, \mathbf{f}_{2,2}, \widetilde{\mathbf{f}_{2,2}}, \mathbf{h}_1)$. This gives the location of $\widetilde{\mathbf{f}_{2,2}}$ uniquely.

$$\mathbf{h}_3 \;=\; \frac{\widetilde{w_{1,1}} \cdot \mathbf{b}_{1,1} \;+\; \widetilde{w_{2,1}} \cdot \mathbf{b}_{2,1} \;+\; \widetilde{w_{1,2}} \cdot \mathbf{b}_{1,2}}{\widetilde{w_{1,1}} \;+\; \widetilde{w_{2,1}} \;+\; \widetilde{w_{1,2}}} \,.$$

where $\widetilde{w_{1,1}}, \widetilde{w_{2,1}}, \widetilde{w_{1,2}}$ are the new weights after moving $\mathbf{f}_{1,1}$, $\mathbf{p}_{1,1}$, $\mathbf{q}_{2,1}$, $\mathbf{p}_{1,2}$, $\mathbf{q}_{1,1}$. We obtain the constructions

$$\begin{aligned}
\mathbf{h}_1 &:= \mathrm{int}(\mathrm{lin}(\mathbf{b}_{3,2}, \mathbf{p}_{2,3}), \mathrm{lin}(\mathbf{b}_{2,3}, \mathbf{q}_{3,2}))\,, \\
\mathbf{h}_2 &:= \mathrm{int}(\mathrm{lin}(\mathbf{b}_{1,2}, \mathbf{p}_{1,1}), \mathrm{lin}(\mathbf{b}_{2,1}, \mathbf{q}_{1,1}))\,, \\
\mathbf{h}_3 &:= \mathrm{int}(\mathrm{lin}(\mathbf{b}_{1,2}, \widetilde{\mathbf{p}_{1,1}}), \mathrm{lin}(\mathbf{b}_{2,1}, \widetilde{\mathbf{q}_{1,1}}))\,.
\end{aligned}$$

Then the points $\mathbf{b}_{2,2}$,$\mathbf{f}_{2,2}$,$\mathbf{h}_1$ are colinear. The points $\mathbf{b}_{2,2}$,$\mathbf{f}_{1,1}$,$\mathbf{h}_2$ and $\mathbf{b}_{2,2}$, $\widetilde{\mathbf{f}_{1,1}}$,$\mathbf{h}_3$ are also colinear. Now consider figure 7.20b. We construct

$$\begin{aligned}
\mathbf{h}_4 &:= \mathrm{int}(\mathrm{par}(\mathrm{lin}(\mathbf{b}_{2,2}, \widetilde{\mathbf{f}_{1,1}}), \mathbf{f}_{1,1}), \mathrm{par}(\mathrm{lin}(\mathbf{h}_2, \mathbf{h}_3), \widetilde{\mathbf{f}_{1,1}}))\,, \\
\mathbf{h}_5 &:= \mathrm{int}(\mathrm{lin}(\mathbf{b}_{2,2}, \mathbf{f}_{1,1}), \mathrm{lin}(\mathbf{h}_3, \mathbf{h}_4))\,.
\end{aligned}$$

Then it is a straightforward exercise in algebra to show that

$$\mathrm{cr}(\mathbf{b}_{2,2}, \mathbf{f}_{1,1}, \mathbf{h}_5, \mathbf{h}_2) = \mathrm{cr}(\mathbf{b}_{2,2}, \mathbf{f}_{2,2}, \widetilde{\mathbf{f}_{2,2}}, \mathbf{h}_1) = \frac{\widetilde{w_{2,2}}}{w_{2,2}}.$$

From this fact it is a basic construction to get $\widetilde{\mathbf{f}_{2,2}}$ from $\mathbf{b}_{2,2}$, $\mathbf{f}_{2,2}$, $\mathbf{h}_1$, $\mathbf{f}_{1,1}$, $\mathbf{h}_5$, $\mathbf{h}_2$ (see for instance [54]). The adjusted points $\widetilde{\mathbf{f}_{0,2}}$, $\widetilde{\mathbf{f}_{0,0}}$, $\widetilde{\mathbf{f}_{2,0}}$ can be constructed in a similar way. Thus the problem of adjusting the adjacent Farin points geometrically is solved.

7.1.2.2 Independent Farin points in a rectangular Bézier point scheme

In this section we want to establish a subset of independent Farin points for rectangular Bézier point schemes. Here we only consider Bézier patches of the

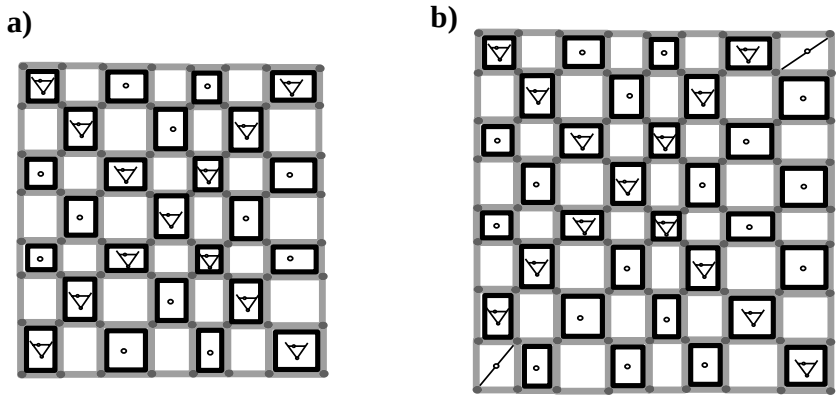


Figure 7.21: A system of independent Farin points for a Bézier patch of the order $n \times n$: a) n odd; b) n even. The system consists of of extended Farin points and normal Farin points. In case b), the weights of two opposite Bézier points have to be fixed by introducing two Farin points on a line segment.

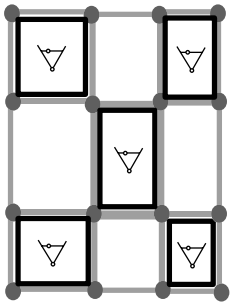


Figure 7.22: Describing the weight ratios in a bicubic rational Bézier patch with 5 extended Farin points.

order $n \times n$. The solution we want to present here consists of a number of extended Farin points and in addition a number of "normal" Farin points $\mathbf{f}_{i,j}$ (i.e. the barycentric combination of the four corners of the subquadrilateral).

Figure 7.21a shows the solution for a patch of the order $n \times n$ where n is odd. We provide the quadrilateral in the middle of the patch with an extended Farin point. The same is done with the quadrilaterals "on the diagonals" of the patch. For the other patches we provide every second one with a "normal" Farin point. In figure 7.21a the patches with a "normal" or extended Farin point are marked with a black closed line. An extended Farin point is marked by the icon similar to figure 7.15, a "normal" Farin point is marked with a hollow dot.

The solution for an even n is more complicated, see figure 7.21b. Here we provide the patches "on one diagonal" with an extended Farin point. The other extended Farin points are next to the other diagonal. The rest is filled with normal Farin points for every second patch. This way, the weights of two opposite corner Bézier points have to be fixed explicitly. This can be done by using two Farin points on a line, as shown in figure 7.21b.

Figure 7.22 shows that the weight ratios in a bicubic Bézier patch can be described in terms of 5 extended Farin points. Since bicubic patches often

occur in practical applications, this example indicates a practical relevance of the approach presented in this section.

Remarks:

- A similar approach as introduced in [3] for the triangular case is possible for the rectangular case. Here we have to search for a system of independent Farin points $\mathbf{p}_{i,j}, \mathbf{q}_{i,j}$ on the line segments of the control net. As in the triangular case, this approach would require more Farin points than the solution introduced in this section.

- The solution introduced in this section is not the only one. In fact, other systems of extended and normal Farin points are thinkable which solve the problem as well. The distinguishing property of our scheme is the symmetry along at least one "diagonal" in the control net.

7.2 Visualizing Curves and Surfaces

From a certain point in the CAGD pipeline up to its end, the data we have to deal with are parametric curves and surfaces, i.e. data of the type $E^{V_2}_{[1]}$, $E^{V_3}_{[1]}$, and $E^{V_3}_{[2]}$ (see section 2.1). This data has to be converted into a graphical representation. To do so, a number of approaches exist which are usually treated under the concept of *curve/surface rendering*. The name "rendering" is more common than the concept "curve/surface visualization". This is due to the fact that the dimensionality both of the underlying grid and of the range of values are that small that there are no further challenges to Scientific Visualization. In particular, the mapping step of the curve/surface visualization is trivial because it is a rather obvious approach to map curves to straight line segments and to map surfaces to polygonal meshes.

In this section we give a brief overview of rendering/visualization methods for curves/surfaces. A more detailed survey on this subject can be found in [19].

To render parametric curves, the usual approach is to compute an appropriate number of sample points and to represent the curve as a set of straight line segments between these points. These line segments can be sent to the output pipeline of a graphical workstation. If the curves live in 3D space, care may be taken to show their spacial behavior. In addition to standard methods of classical 3D computer graphics (such as hidden line removal algorithms), the concept of illuminated stream lines ([213]) may be used for rendering 3D curves: instead of straight line segments, the curve is represented by narrow cylinders which can be rendered using illumination and shading. The 3D curves in figures 3.14 and 3.16 of this work are rendered in such a way to emphasize their spatial behavior.

To render parametric surfaces, three general approaches exist ([19]): isocurve extraction, stepwise refinement, and raytracing. For *isocurve extraction*, the surface rendering is put down to to curve rendering. A family of parametric curves on the surface is chosen, and an appropriate number of representatives of them are rendered. These families of curves may be isoparametric lines u =const or v =const. However, the choice of an appropriate number of representatives

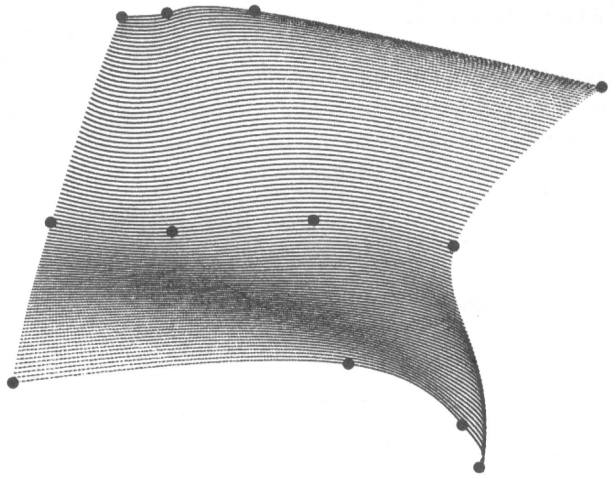


Figure 7.23: Surface rendering by extracting isoparametric lines; shown is a tensor-product B-spline-surface consisting of 3×2 bicubic patches; (image from [55]).

has a strong influence on the result of the rendering process. In case of a local undersampling, important information about the surface behavior gets lost. On the other hand, local oversampling may cause aliasing effects and visual clutter. Figure 7.23 shows an example of surface rendering by extracting a family of isoparametric lines.

Scientific Visualization was faced with a similar problem as the choice of an appropriate number of isoparametric curves for surface rendering. In flow visualization, for visualizing tangent curves, an appropriate number of tangent curves has to be selected (see section 4.3.1.1). There, a number of adaptive tangent curve selection methods ([196], [104]) have been developed which ideas can be applied for isoparametric line extraction on surfaces as well.

To render a surface using a *stepwise refinement*, the surface is approximated by a triangular mesh. To do so, either a number of sample points on the surface is constructed and triangulated, or the surface is stepwise refined by subdivision[5]. The resulting triangular mesh can be rendered by applying standard methods of 3D computer graphics. To do so, 3D computer graphics offers a number of approaches (visibility computation, shading, texturing, shadow computation) to increase the realism of the rendering. Figure 7.24 shows the rendering for a subdivided triangular surface for a number of subdivision steps.

To render a surface using *raytracing*, the intersections between the surface and a straight line have to be computed. For sufficiently complicated surfaces, this is

[5]Subdivision surfaces are another popular class of surfaces. Instead of a parametric description, the surface is given by an initial polygonal mesh and a number of rules to refine this mesh by subdivision. This way a polygonal approximation of the final surface in any resolution can be obtained. A treatment of subdivision surfaces is beyond the scope of this work.

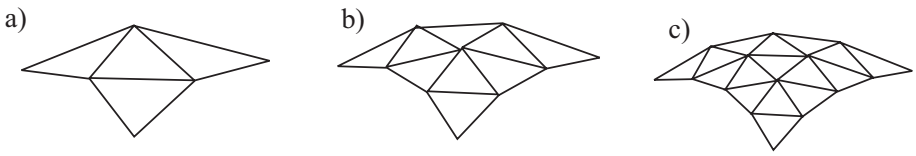


Figure 7.24: Surface rendering by stepwise refinement; a) triangular approximation consisting of 4 triangles; b) 9 triangles; c) 16 triangles.

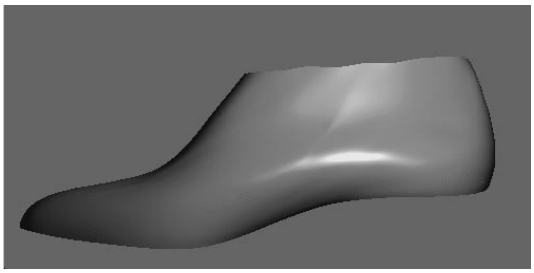


Figure 7.25: Raytracing a surface; shown is a bicubic B-spline surface consisting of 15×10 patches.

only possible by applying numerical integration methods[6]. [19] studies numerical intersection methods for surfaces and straight lines. An example of rendering a surface using raytracing is shown in figure 7.25.

7.3 Visualization for Surface Interrogation

Surface interrogation techniques have the task of evaluating the quality of a designed surface by detecting small imperfections on it. Surface interrogation focuses on a visual analysis of the surface behavior and is therefore a promising candidate for applying methods of Scientific Visualization. Surveys on standard methods of surface interrogation are in [79] and [81].

Most surface interrogation methods focus either on the treatment of characteristic surfaces, or on characteristic surface curves. Examples for characteristic surfaces are focal surfaces ([79]) and stability surfaces ([81]). These surfaces are derived from the designed surface and reflect certain geometric properties of it. This way the visualization of characteristic surfaces may reveal geometric imperfections of the designed surfaces.

[6]For example, a bicubic polynomial patch may have up to 18 intersections with a straight line. To show this, consider the bicubic patch

$$\mathbf{x}(u,v) = \begin{pmatrix} (u-\frac{1}{7})(v-\frac{2}{7})(u-\frac{3}{7})(v-\frac{4}{7})(u-\frac{5}{7})(v-\frac{6}{7}) \\ (u-\frac{2}{7})(v-\frac{1}{7})(u-\frac{4}{7})(v-\frac{3}{7})(u-\frac{6}{7})(v-\frac{5}{7}) \\ z(u,v) \end{pmatrix}$$

with $(u,v) \in [0,1]^2$ and $z(u,v)$ being any bicubic scalar function. Intersecting this surface with the line $\mathbf{l}(t) = (0,0,0)^T + t\,(0,0,1)^T$ gives intersections for the following (u,v)-values: $\left(\frac{1}{7},\frac{1}{7}\right)$, $\left(\frac{3}{7},\frac{1}{7}\right)$, $\left(\frac{5}{7},\frac{1}{7}\right)$, $\left(\frac{1}{7},\frac{3}{7}\right)$, $\left(\frac{3}{7},\frac{3}{7}\right)$, $\left(\frac{5}{7},\frac{3}{7}\right)$, $\left(\frac{1}{7},\frac{5}{7}\right)$, $\left(\frac{3}{7},\frac{5}{7}\right)$, $\left(\frac{5}{7},\frac{5}{7}\right)$, $\left(\frac{2}{7},\frac{2}{7}\right)$, $\left(\frac{4}{7},\frac{2}{7}\right)$, $\left(\frac{6}{7},\frac{2}{7}\right)$, $\left(\frac{2}{7},\frac{4}{7}\right)$, $\left(\frac{4}{7},\frac{4}{7}\right)$, $\left(\frac{6}{7},\frac{4}{7}\right)$, $\left(\frac{2}{7},\frac{6}{7}\right)$, $\left(\frac{4}{7},\frac{6}{7}\right)$, $\left(\frac{6}{7},\frac{6}{7}\right)$.

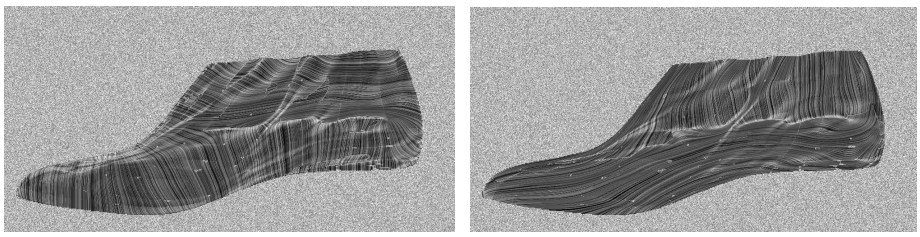


Figure 7.26: Visualization of the two families of lines of curvature using a combination of Integrate&Draw and curvature plot.

To analyze a designed surface using characteristic surface curves, a number of families of these surface curves exists, such as

- contour lines ([192])
- lines of curvature ([55])
- asymptotic lines ([55])
- isophotes ([151])
- reflection lines ([115], [108]).

In [192] it has been shown that all these families of characteristic surface curves have something in common:

- They reflect geometric properties of the designed surface.
- They react rather sensitively to small bumps and perturbations of the surface. Hence they are promising candidates to reveal imperfection in the surfaces.
- For sufficiently complicated surfaces (such as bicubic polynomial surfaces), these curves cannot be described as closed parametric curves but only as numerical solutions of a system of differential equations.

The last-named property creates a number of problems for visualizing the characteristic surface curves. Fortunately it turned out that all the characteristic surface curves mentioned above can be represented as tangent curves of rather simple[7] vector fields - both in the domains of the surfaces and on the surfaces themselves (see [192]). With this property in mind, virtually all visualization techniques for vector fields can be used to visualize characteristic curves on surfaces. Figure 7.26 shows an example of visualizing the two families of lines of curvature of the surface which was already shown in figure 7.25. Here we used a combination of two techniques of 2D vector field visualization: Integrate&Draw (see section 4.3.3.2) and curvature plot (see section 4.3.3.3). We can clearly see the "flow" behavior of the lines of curvature. The additionally color-coded curvature information reveals the underlying patch structure of the surface.

Another example of applying flow visualization techniques to characteristic curves on surfaces is shown in figure 7.27. Figure 7.27b shows the visualization

[7] simple in the sense that the vector field can easily be obtained from the surface and the configuration of the families of characteristic surface curves.

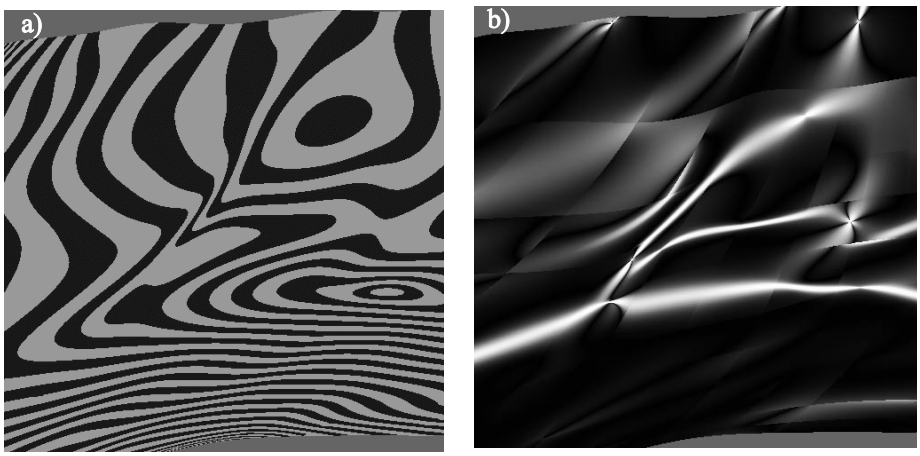


Figure 7.27: A class of isophotes on a fragment of the surface of figure 7.25; b) visualization of the geodesic curvature of the isophotes (images from [192]).

of the geodesic curvature of a class of isophotes shown in figure 7.27a. Figure 7.27b clearly reveals areas of high isophote curvature (represented by bright colors) where a redesign of the surface is necessary. (The surface shown there is a fragment of the surface of figure 7.25.)

The fact that characteristic curves on surfaces can be interpreted as tangent curves of vector fields can not only be used to visualize the characteristic surface curves. It can also be used to prove certain properties of the surface curves. In [186] it was shown that there is – except for some special cases which have to be excluded – a one-to-one correlation between the G^3 continuity of a surface and the G^2 continuity of lines of curvature or the asymptotic lines on this surface. Using this statement, the discontinuities in the curvature plot of the lines of curvature in figure 7.26 show that the designed surface is not G^3 continuous.

In [187] it has been shown that – except for some special cases which have to be excluded – a G^n continuity of a surface can be deduced from a G^{n-1} continuity of a class of isophotes on the surface. This means for instance that the surface in figure 7.27 is not G^3 continuous because the curvature plot of the isophotes shown in figure 7.27b has discontinuities.

Chapter 8

Summary and Future Research

In this work we have investigated the correlations between the disciplines CAGD and Scientific Visualization. Based on a historical analysis, an analysis of the present data, and an analysis of the pipelines of both disciplines, we formulated expectations on where it makes sense to search for applications of one discipline into the other. The detailed treatment of these applications confirmed these expectations. Moreover, following the strategy of investigating the interapplicability of of CAGD and Scientific Visualization, we were not only able to systematize existing approaches, we also developed and introduced a number of new techniques which apply ideas and methods of one discipline to the other.

Up to now, in both disciplines a large amount of research has been done. Also the potential interapplications are so various that we had to consider some restrictions to keep the issue manageable in this work. In particular, in CAGD we focused on parametric curves and surfaces. Other approaches of describing and designing curves and surfaces, such as subdivision surfaces and implicit surfaces, had to be excluded here. In Scientific Visualization we focused on the the treatment of particular visualization techniques more than on aspects of steering the visualization. In particular, approaches of visualization control, collaborate visualization and visualization environments could not be treated here.

For the future it seems to be an interesting task to investigate the application of subdivision surfaces to Scientific Visualization. The idea of defining subdivision surfaces can also be applied to describe vector- and volume data sets. [204] gives a first example of describing vector fields by subdivision.

A further challenge is the systematic treatment of CAGD methods in disciplines which are related to Scientific Visualization. For example, in medical imaging, there are already a number of applications of curves and surfaces. Another candidate for a systematic application CAGD methods is visual data mining.

Also in the issues treated in this work there are still a number of open questions which we want to summarize in the following:

- There are unsolved problems concerning a topology-preserving interpolation of vector fields. In this work we have shown that a 2D vector field of any topology can be described as a piecewise linear vector field, i.e. a vector field of a C^0 continuity. So it is a natural next question to ask for interpolation schemes which preserve the topology of the piecewise linear vector field but yield a global C^1 (or higher) continuity.

- The concepts of distances of first order critical points of 2D vector fields should be extended to 2D critical points of general topology.

- Although not treated in this work,there may be further applications of Scientific Visualization in the upper parts of the CAGD pipeline. For instance, the search for an appropriate parameterization of a B-spline curve consisting of n cubic segments can be considered as an optimization process in the n-dimensional space of all parameterizations. Here visualization approaches of multidimensional data may be applied to support the designer in this search.

Bibliography

[1] S. Abramowski and H. Müller. *Geometrisches Modellieren*. Wissenschaftsverlag, Mannhein, 1991. (in German).

[2] S. Abramowski and H. Müller. Searching connected components in very large grid graphs. In *Lecture Notes in Computer Science*, 246, pages 118–130. Springer, Berlin, 1991.

[3] G. Albrecht. A note on Farin points for rational triangular Bézier patches. *Computer Aided Geometric Design*, 12:507–512, 1995.

[4] P. Alfeld. Scattered data interpolation in three or more variables. In L.L Schumaker and T. Lyche, editors, *Mathematical Methods in Computer Aided Geometric Design*. Academic Press, 1989.

[5] F. Allamandri, P. Cignoni, C. Montani, and R. Scopigno. Adaptively adjusting marching cubes output to fit a trilinear reconstruction filter. In D. Bartz, editor, *Visualization in Scientific Computing '98*, pages 25–34. Springer, Wien and New York, 1998.

[6] T. Avidan and S. Avidan. Parallax - a data mining tool based on parallel coordinates. *Computational Statistics*, 14:79–89, 1999.

[7] C. L. Bajaj, V. Pascucci, and D. R. Schikore. Fast Isocontouring For Improved Interactivity. In *Symposium on Volume Visualization*, pages 39–46, Los Alamitos, October 1996. IEEE Computer Society Press.

[8] C.L. Bajaj, V. Pascucci, and D.R. Schikore. Visualization of scalar topology for structural enhancement. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 51–58, Los Alamitos, 1998. IEEE Computer Society Press.

[9] G. Barequet, D. Shapiro, and D. Tal. History consideration in reconstructing polyhedral surfaces from parallel slices. In *Proc. IEEE Visualization '96*, pages 149–173, Los Alamitos, 1996. IEEE Computer Society Press.

[10] R. Barnhill. Smooth interpolation over triangles. In R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 45–70. Academic Press, 1974.

[11] R. Barnhill and G. Farin. C^1 quintic interpolation over triangles: two explicit representations. *Int. J Numer. Methods in Eng.*, 17:1763–1778, 1981.

[12] R. Batra and L. Hesselink. Feature comparisons of 3-d vector fields using earth mover's distance. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 105–114, Los Alamitos, 1999.

[13] R. Batra, K. Kling, and L. Hesselink. Topology based vector field comparison using graph methods. In A. Varshney, C.M. Wittenbrink, and H. Hagen, editors, *Proc. IEEE Visualization '98, Late Breaking Hot Topics*, pages 25–28, 1999.

[14] H. Battke, D. Stalling, and H. Hege. Fast line integral convolution for arbitrary surfaces in 3d. *Preprint SC-96-59*, 1996. Konrad-Zuse-Zentrum für Informationstechnik, Berlin.

[15] J. Beddow. Shape coding of multidimensional data on a microcomputer display. In *Proc. IEEE Visualization '90*, pages 238–246, Los Alamitos, 1990. IEEE Computer Society Press.

[16] S. Benford, D. Snowdon, C. Greenhalgh, R. Ingram, I. Knox, and C. Brown. VR-VIBE: A virtual environment for co-operative information retrieval. In *Proc. Eurographics '95*, pages C349 – C360. Blackwell Publishers, 1995.

[17] R.D. Bergeron and G. Grienstein. A reference model for the visualization of multidimensional data. In *Proc. Eurographics '89*, pages 393–399, Hamburg, 1989.

[18] C. Beshers and S. Feiner. Autovisual: Rule-based design of interactive multivariate visualizations. *IEEE Computer Graphics and Applications*, 13(4):41–49, 1993.

[19] M. Boldt. *Visualisierung für Freiformflächen*. PhD thesis, University of Rostock, 1994. (in German).

[20] G.P. Bonneau. Variational design of rational Bézier curves. *Computer Aided Geometric Design*, 8:393–399, 1991.

[21] E. Boring and A. Pang. Interactive deformations from tensor fields. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 297–304, Los Alamitos, 1998. IEEE Computer Society Press.

[22] K. Brodlie and P. Mashwama. Controlled interpolation for scientific visualization. In G.M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization*, pages 253–276. IEEE Computer Society, 1997.

[23] K.W. Brodlie et al. *Scientific Visualization*. Springer, Berlin, 1992.

[24] G. Brunnett, H. Hagen, and P. Sntarelli. Variational design for curves and surface. *Surveys on Mathematics for Industry*, 3:1–27, 1993.

[25] T. Bürger. Magic Eye View: Eine neue Focus + Context - Technik zur Darstellung von Graphen. Diplomarbeit, University of Rostock, Computer Science Department, 1999. (in German; also available at http://wwwicg.informatik.uni-rostock.de/Projekte/koan.html).

[26] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. *Computer Graphics*, 27:263–272, 1993.

[27] S.K. Card, J.D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization - Using Vision to Think*. Morgan Kaufman, San Francisco, CA, 1999.

[28] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of American Statistical Association*, 68:361–368, 1973.

[29] M. S. Chong, A. E. Perry, and B. J. Cantwell. A general classification of three-dimensional flow fields. *Physics of Fluids A*, 2(5):765–777, 1990.

[30] S.Y. Chou, S.W. Lin, and C.S. Yeh. Cluster identification with parallel coordinates. *Pattern Recognition letters*, 20:565–572, 1999.

[31] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computers & Graphics*, 24:399–418, 2000.

[32] W.S. Cleveland. *Visualizing Data*. Hobart Press, Summit New Jersey, 1993.

[33] B.M. Collins. Data visualization - has it all been seen before? In *Animation and Scientific Visualization*. British Computer Society, State of the Art Report, 1992.

[34] S. Coons. Surfaces for computer aided design of space forms. *Technical report*, 1967. MIT, Project MAC-TR 41.

[35] B.H. Mc Cormick, T.A. DeFanti, and M.D. Brown. Visualization in scientific computing. *Computer Graphics*, 21(6):1–14, 1987.

[36] M. Cox. The numerical evaluation of b-splines. *Journals Inst. Math. Applics.*, 10:134–149, 1972.

[37] A. Dai and A. Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE Trans. Commun. Elec. Inf. Syst. E-74*, 1:214–224, 1991.

[38] H. Davis. *Introduction to vector analysis*. Allyn and Bacon, Inc., Boston, 1967.

[39] C. de Boor. On calculating with b-splines. *Journals Approximation Theory*, 6(1):50–62, 1972.

[40] P. de Casteljau. Outillages méthodes calcul. *Technical report, Citroen*, 1959. (in French).

[41] P. de Casteljau. Courbes et surfaces poles. *Technical report, Citroen*, 1963. (in French).

[42] W. de Leeuw and R. van Liere. Comparing LIC and spot noise. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 359–365, Los Alamitos, 1998. IEEE Computer Society Press.

[43] W. de Leeuw and R. van Liere. Collapsing flow topology using area metrics. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 149–354, Los Alamitos, 1999.

[44] W. de Leeuw and J. van Wijk. A probe for local flow field visualization. In *Proc. IEEE Visualization '93*, pages 39–45, Los Alamitos, 1993. IEEE Computer Society Press.

[45] T.A. Defanti and M.D. Brown. Foreword. In L. Rosenblum et al., editors, *Scientific Visualization. Advances and Challenges*. Academic Press, London, 1994.

[46] T. Delmarcelle and L. Hesselink. Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics and Applications*, 13(4):25–33, 1993.

[47] M.J. Dürst. Additional references to marching cubes. *ACM Computer Graphics*, 22(2):72–73, 1988.

[48] M. Eck and H. Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *Proc. Siggraph '96*, pages 325–334, 1996.

[49] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical morse complexes for piecewise linear 2-manifolds. In *Proc. 17th Sympos. Comput. Geom. 2001*, 2001. to appear.

[50] J. Encarnacao, W. Strasser, and R. Klein. *Graphische Datenverarbeitung 1. Band*. Oldenbourg Verlag, München, Wien, 1996. (in German).

[51] J. Encarnacao, W. Strasser, and R. Klein. *Graphische Datenverarbeitung 2. Band*. Oldenbourg Verlag, München, Wien, 1997. (in German).

[52] K. Engel, R. Westermann, and T. Ertl. Isosurface extraction techniques for web-based volume visualization. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 139–146, Los Alamitos, 1999.

[53] Euklid. *Die Elemente*. Akademische Verlagsgesellschaft m.b.H., Leipzig, 1933. (German translation).

[54] G. Farin. *NURB Curves and Surfaces*. A K Peters, Wellesley, MA, 1995. Second edition 1999.

[55] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, Boston, 4th edition, 1997.

[56] G. Farin and D. Hansford. *The Essentials of CAGD*. A K Peters, Natick, MA, 2000.

[57] P.A. Firby and C.F. Gardiner. *Surface Topology*, chapter 7, pages 115–135. Ellis Horwood Ltd., 1982. Vector Fields on Surfaces.

[58] J. Foley and B. Ribarsky. Next-generation data visualization tools. In Rosenblum et al., editors, *Scientific Visualization*, pages 103–127. Academic Press, London, 1994.

[59] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley Publishing Company, Reading Massachusetts, 1996.

[60] K.L. Forsell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In *Proc. IEEE Visualization '94*, pages 240–247, Los Alamitos, 1994. Computer Society Press.

[61] L.K. Forsell and S.D. Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, 1995.

[62] R. Franke. Scattered data interpolation: tests and some methods. *Math. Computation*, 38(157):181–200, 1982.

[63] R. Franke. Recent advances in the approximation of surfaces from scattered data. *Topics in Multivariate Approximation*, pages 79–98, 1987.

[64] T. Frühauf. *Graphisch-Interaktive Strömungsvisualisierung*. Springer-Verlag, Berlin, 1997. (in German).

[65] I. Fujishiro, Y. Maeda, and H. Sato. Interval volume: A solid fitting technique for volumetric data display and analysis. In G. Nielson and D. Silver, editors, *Proc. IEEE Visualization '95*, pages 151–158, Los Alamitos, 1995.

[66] R.S. Gallagher and J.C. Nagtegaal. An efficient 3-d visualization technique for finite element models and other coarse volumes. In *Proc. Siggraph '89*, 1989.

[67] H. Garcke, T. Preusser, M. Rumpf, A. Telea, U. Weikardt, and J. van Wijk. A continuous clustering method for vector fields. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proc. IEEE Visualization 2000*, pages 351–358, 2000.

[68] M.G. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *Proc. Siggraph '97*, pages 209–216, 1997.

[69] A. Globus and C. Levit. A tool for visualizing of three-dimensional vector fields. In *Proc. IEEE Visualization '91*, pages 33–40, Los Alamitos, 1991. IEEE Computer Society Press.

[70] K.U. Graw, S. Lange, N. López, and H. Schumann. Konzept und Realisierung einer intelligenten Visualisierungshilfe. In *Preprint CS-08-97*, University of Rostock, 1997. (in German).

[71] J. Gregory. Smooth interpolation without twist constraints. In R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 71–88. Academic Press, 1974.

[72] G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum*, 13(3):143–154, 1994.

[73] C.M. Grimm and J.F. Hughes. Smooth isosurface approximation. In B.Wyvill and M.P. Gascuel, editors, *Implicit Surface '95, Eurographics Workshop*, pages 57–76. Blackwell Publishers, 1995.

[74] M.H. Gross, T.C. Sprenger, and J. Finger. Visualizing information on a sphere. In *Proc. of Information Visualization '97 Symposium (Infovis '97)*, pages 11–16, 1997.

[75] R. Grosso and T. Ertl. Progessive Isosurface Extraction from Hierarchical 3D Meshes. *Computers Graphics Forum (EUROGRAPHICS '98)*, 17(3), September 1998.

[76] B. Guo. Interval set: A volume rendering technique generalizing isosurface extraction. In G. Nielson and D. Silver, editors, *Proc. IEEE Visualization '95*, pages 3–10, Los Alamitos, 1995.

[77] B. Guo. Surface reconstruction: from points to splines. *Computer Aided Design*, 29(4):269–277, 1997.

[78] R.B. Haber and D.A. Mc Nabb. Visualization idioms: a conceptual model for scientific visualization systems. In B. Shriver, G. Nielson, and L. Rosenblum, editors, *Visualization in Scientific Computing*. IEEE Computer Society Press, Los Alamitos, 1990.

[79] H. Hagen et al. Surface interrogation algorithms. *IEEE Computer Graphics and Applications*, 12(5):53–60, 1992.

[80] H. Hagen, S. Hahmann, and G.P. Bonneau. Variational surface design and surface interrogation. *Computer Graphics Forum*, 12(3):447–460, 1993.

[81] S. Hahmann. *Stabilitäts- und Qualitätsanalyse von Freiformflächen*. PhD thesis, University of Kaiserslautern, 1994. (in German).

[82] M. Hall and J. Warren. Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, pages 33–42, 1990.

[83] B. Hamann. Visualisierungstechniken zur Darstellung dreidimensionaler Datenmengen. *CAD und Computergraphik*, 13(5):129–139, 1991. (in German).

[84] B. Hamann, I. J. Trotts, and G. Farin. Approximating contours of the piecewise trilinear interpolant using triangular rational quadratic Bézier patches. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):215–227, 1997.

[85] D. Hearn and P. Baker. Scientific visualization tutorial notes. In *Eurographics '91*, Wien, 1991.

[86] B. Heckel, G.H. Weber, B. Hamann, and K.I.Joy. Construction of vector field hierarchies. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 19–26, Los Alamitos, 1999.

[87] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, August 1989.

[88] J. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11:36–46, May 1991.

[89] G.T. Hermann and H.K. Lin. Three dimensional display of human organs from computed tomograms. *Computer Vision, Graphics and Image Processing*, 9:1–21, 1979.

[90] L. Hesselink, Y. Levy, and Y. Lavin. The topology of symmetric, second-order 3D tensor fields. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):1–11, 1997.

[91] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley. DNA visual and analytic data mining. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 437–441, 1997.

[92] D.J. Holliday and G.M. Nielson. Progressive volume models for rectilinear data using tetrahedral coons volumes. In W. de Leeuw and R. van Liere, editors, *Data Visualization 2000*, pages 83–92. Springer, Wien and New York, 2000.

[93] H. Hoppe. Progressive meshes. In *Proc. Siggraph '96*, pages 99–108, 1996.

[94] H. Hoppe. New quadric metric for simplifying meshes with appearance attributes. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 59–66, Los Alamitos, 1999.

[95] J. Hoschek and D. Lasser. *Grundlagen der Geometrischen Datenverarbeitung*. B.G. Teubner, Stuttgart, 1989. (in German; English translation: Fundamentals of Computer Aided Geometric Design, AK Peters, 1993).

[96] J. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *Proc. IEEE Visualization '92*, pages 171–177, Los Alamitos, 1992. IEEE Computer Society Press.

[97] A. Inselberg. Visual data mining with parallel coordinates. *Computational Statistics*, 13:47–63, 1998.

[98] A. Inselberg. Don't panic ... just do it in parallel! *Computational Statistics*, 14:53–77, 1999.

[99] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proc. IEEE Visualization '90*, pages 361–375, Los Alamitos, 1990. IEEE Computer Society Press.

[100] A. Inselberg, M. Reif, and T. Chomut. Convexity algorithms in parallel coordinates. *Journal ACM*, 34:765–801, 1987.

[101] V. Interrante and C. Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 421–425, 1997.

[102] T. Itoh, Y. Yamaguchi, and K. Koyamada. Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):319–327, December 1995.

[103] T. Itoh, Y. Yamaguchi, and K. Koyamada. Volume Thinning for Automatic Isosurface Propagation. In R. Yagel and G. M. Nielson, editors, *Proc. IEEE Visualization '96*, pages 303–310, Los Alamitos, 1996.

[104] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proceedings 8th Eurographics Workshop on Visualization in Scientific Computing*, pages 57–66, Boulogne, 1997.

[105] J.K. Johnstone and K.R. Sloan. Tensor product surfaces guided by minimal surface area triangulations. In G. Nielson and D. Silver, editors, *Proc. IEEE Visualization '95*, pages 254–261, Los Alamitos, 1995.

[106] M.W. Jones. A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13:75–84, 1994.

[107] A. Kaufman. *Volume Visualization. Tutorial Notes*. IEEE Computer Society Press, Los Alamitos, 1990.

[108] E. Kaufmann and R. Klass. Smoothing surfaces using reflection lines for families of splines. *Computer Aided Design*, 10(6):312–316, 1988.

[109] D. Keim, M. Ankerst, and H.P. Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In G. Nielson and D. Silver, editors, *Proc. IEEE Visualization '95*, pages 279–287, Los Alamitos, 1995.

[110] D. Keim and H.P. Kriegel. Visdb: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, pages 40–49, September 1994.

[111] D. Keim, H.P. Kriegel, and T. Seidel. Visual feedback in querying large data bases. In *Proc. IEEE Visualization '93*, pages 158–165, Los Alamitos, 1993. IEEE Computer Society Press.

[112] D.N. Kenwright, C. Henze, and C. Levit. Feature extraction of separation and attachment lines. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):135–144, 1999.

[113] G.D. Kerlick. Moving iconic objects in scientific visualization. In A. Kaufman, editor, *Proc. IEEE Visualization '90*, pages 124–130, Los Alamitos, 1990. IEEE Computer Society Press.

[114] G. Kindlmann and D. Weinstein. Hue-balls and lit-tensors for direct volume rendering of diffusion tensor fields. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 183–189, Los Alamitos, 1999.

[115] R. Klass. Correction of local surface irregularities using reflection lines. *Computer Aided Design*, 12(2):73–77, 1980.

[116] J.J. Koendering and A.J. van Doorn. Relief: pictorial and otherwise. *Image and Vision Computing*, 13(5):321–334, 1995.

[117] M. Köller. Ein Testbed zur Untersuchung von Higher Order Parallel Coordinates. Studienarbeit, University of Rostock, Computer Science Department, 1999. (in German).

[118] M. Köller. Schwellwertunabhängiges Preprocessing für Marching Cubes. Diplomarbeit, University of Rostock, Computer Science Department, 2000. (in German).

[119] M.v. Krefeld, R.v. Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour Trees and Small Seed Sets for Isosurface Transversal. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, New York, June 1997. ACM Special Interest Groups for Graphics and Algorithms and Computation Theory, ACM Press.

[120] M. Kreuseler. Visualization of geographically related multidimensional data in virtual 3d scenes. *Computers & Geosciences, special issue on Geoscientific Visualization*, 1999.

[121] M. Kreuseler, H. Schumann, and N. Lopez. A scalable framework for information visualization. In *Proc. Information Visualization 2000 (INFOVIS 2000)*, Salt Lake City, Utah, 2000.

[122] Y. Lavin, R.K. Batra, and L. Hesselink. Feature comparisons of vector fields using earth mover's distance. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 103–109, Los Alamitos, 1998. IEEE Computer Society Press.

[123] Y. Lavin, Y. Levy, and L. Hesselink. Singularities in nonuniform tensor fields. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 59–66, 1997.

[124] J. Leblanc, M.O. Ward, and N. Wittels. Exploring n-dimensional databases. In *Proc. IEEE Visualization '90*, pages 230–237, Los Alamitos, 1990. IEEE Computer Society Press.

[125] H. Levkowitz. Color icons: Merging color and texture perception for integrated visualization of multiple parameters. In *Proc. IEEE Visualization '91*, pages 164–170, Los Alamitos, 1991. IEEE Computer Society Press.

[126] M. Levoy, P. Hanrahan, A. Kaufman, and W. Lorensen. Volume visualization. In *Tutorial notes, Visualization '90*. IEEE Computer Society Press, Los Alamitos, 1990.

[127] Y. Livnat, H.W. Shen, and C. R. Jonson. A Near Optimal Isosurface Extraction Algorithm Using the Span Space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, 1996.

[128] S.K. Lodha, J.C. Renteria, and K.M. Roskin. Topology preserving compression of 2d vector fields. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proc. IEEE Visualization 2000*, pages 343–350, 2000.

[129] A. Lopez and K.W. Brodlie. Accuracy in 3d particle tracing. In H.C. Hege and K. Polthier, editors, *Mathematical Visualization: Algorithms, Applications and Numerics*. Springer, 1998.

[130] W.E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3d surface reconstruction algorithm. *Computer Graphics*, 21:163–169, 1987.

[131] X. Mao, M. Kikukawa, N. Fujeta, and N. Imamiya. Line integral convolution for 3d surfaces. In W. Lefer and M. Grave, editors, *Visualization in Scientific Computing '97*, pages 57–69. Springer, Berlin, 1997.

[132] R. Mencl and H. Müller. Interpolation and approximation of surfaces from three-dimensional scattered data points. In A. Augusto de Sousa and B. Hopgood, editors, *Eurographics '98, State of the Art Reports*, pages 51–67, 1998.

[133] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In *Proc. IEEE Visualisation '94*, Los Alamitos, 1994. Computer Society Press.

[134] C. Montani, R. Scateni, and R. Scopigno. A modified look-up table for implicit disambiguities of marching cubes. *The Visual Computer*, 10(6):353–355, 1994.

[135] H. Müller and A. Klingert. Surface interpolation from cross sections. In H. Hagen, H. Müller, and G.M. Nielson, editors, *Focus on Scientific Visualization*, pages 139–189. Springer, Berlin, 1993.

[136] H. Müller and M. Stark. Adaptive generation of surfaces in volume data. *The Visual Computer*, 9:182–199, 1993.

[137] T. Munzner. H3: laying out large directed graphs in 3D hyperbolic space. In *Proc. of Information Visualization '97 Symposium (Infovis '97)*, pages 2–10, 1997.

[138] B.K. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11(1):52–62, 1994.

[139] G. M. Nielson and B. Hamann. The asymptotic decider: Resolving the ambiguity in marching cubes. In *Proc. IEEE Visualization '91*, pages 83–91, Los Alamitos, 1991. IEEE Computer Society Press.

[140] G.M. Nielson. A method for interpolating scattered data based upon a minimum norm network. *Mathematics of Computation*, 40:253–271, 1983.

[141] G.M. Nielson. Tools for computing tangent curves and topological graphs for visualizing piecewise linearly varying vector fields over triangulated domains. In G.M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization*, pages 527–562. IEEE Computer Society, 1997.

[142] G.M. Nielson. Tools for triangulations and tetrahedrizations and constructing functions defined over them. In G.M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization*, pages 429–525. IEEE Computer Society, 1997.

[143] G.M. Nielson and B. Hamann. Techniques for the interactive visualization of volumetric data. In *Proc. IEEE Visualization '90*, pages 45–50, Los Alamitos, 1990. IEEE Computer Society Press.

[144] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, 1993.

[145] K.A. Olsen, R.R. Korfhage, K.M.Sochats, M.B.Spring, and J.G. Williams. Visualization of a document collection: The VIBE system. *Information Processing and Management*, 29(1):69–81, 1993.

[146] H.L. Ong and H.Y. Lee. Software report WInViz - a visual data analysis tool. *Computation & Graphics*, 20(1):83–84, 1996.

[147] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.P. Sloan. Interactive ray tracing for isosurface rendering. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 233–238, Los Alamitos, 1998.

[148] P. A. Philippou and R. N. Strickland. Vector field analysis and synthesis using three dimensional phase portraits. *Graphical Models and Image Processing*, 59:446–462, November 1997.

[149] R.M. Pickett and G. Grienstein. Iconographic displays for visualizing multidimensional data. In *Proc. IEEE Conference on Systems, Man and Cybernetics*, pages 514–519, Beijing and Shenyang, 1988. PRC.

[150] L. Piegl and W. Tiller. *The Book of Nurbs*. Springer-Verlag, 1995.

[151] T. Poeschl. Detecting surface irregularities using isophotes. *Computer Aided Geometric Design*, 1(2):163–168, 1984.

[152] F. Post and T. van Walsum. Fluid flow visualization. In H. Hagen, H. Müller, and G.M. Nielson, editors, *Focus on Scientific Visualization*, pages 1–40. Springer, Berlin, 1993.

[153] M. Powell and M. Sabin. Piecewise quadratic approximation on triangles. *ACM Trans. Math. Software*, 3(4):316–325, 1977.

[154] K.J. Renze and J.H. Oliver. Generalized unstructured decimation. *IEEE Computer Graphics and Applications*, 16(6):24–32, 1996.

[155] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. Interactive Exploration of Volume Line Integral Convolution Based on 3D–Texture Mapping. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 233–240, Los Alamitos, 1999.

[156] C. Perez Risquet. Visualizing 2D flows: Integrate and Draw. In *Proceedings 9th Eurographics Workshop on Visualization in Scientific Computing*, pages 57–67, 1998.

[157] G. Robertson, J. Mackinlay, and S. Card. Cone trees: Animated 3D visualizations of hierarchical information. In *Proc. ACM CHI International Conference of Human Factors in Computing (CHI '91)*, pages 189–194, 1991.

[158] A. Rockwood and P. Chambers. *Interactive Curves and Surfaces*. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[159] R.M. Rohrer, D.S. Ebert, and J.S. Sibert. The shape of Shakespeare: Visualizing text using implicit surfaces. In G. Wills and J. Dill, editors, *Proc. Information Visualization '98*, pages 121–129, 1998.

[160] M. Roth and R. Peikert. A higher-order method for finding vortex core lines. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 143–150, Los Alamitos, 1998. IEEE Computer Society Press.

[161] D. Ruprecht and H. Müller. A scheme for edge-based adaptive tetrahedron subdivision. In H.C. Hege and K. Polthier, editors, *Mathematical Visualization*. Springer, 1997.

[162] G. Scheuermann, H. Hagen, H. Krüger, M. Menzel, and A. Rockwood. Visualization of higher order singularities in vector fields. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 67–74, 1997.

[163] G. Scheuermann, H. Krüger, M. Menzel, and A. Rockwood. Visualizing non-linear vector field topology. *IEEE Transactions on Visualization and Computer Grapics*, 4(2):109–116, 1998.

[164] G. Scheuermann, X. Tricoche, and H. Hagen. C1-interpolation for vector field topology interpolation. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 271–278, Los Alamitos, 1999.

[165] W. Schroeder, K. Martin, and B. Lorenzen. *The Visualization Toolkit*. Prentice Hall PTR, 2nd edition, 1998.

[166] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.

[167] H. Schumann and W. Müller. *Visualisierung - Grundlagen und allgemeine Methoden*. Springer-Verlag, 2000. (in German).

[168] C.E. Shannon and W. Weaver. *Mathematische Grundlagen der Informationstheorie*. Oldenbourg-Verlag, München, Wien, 1976. (in German).

[169] C.D. Shaw, J. Hall, C. Blahut, D.S. Ebert, and D.A. Roberts. Using shape to visualize multivariate data. In D.S. Ebert and C.D. Shaw, editors, *Proc. Workshop on New Paradigms in Information Visualization and Manipulation (NPIVM '99)*, pages 17–20. ACM press, 1999.

[170] R. Shekhar, E. Fayyad, R. Yagel, and J. F. Cornhill. Octree-based decimation of marching cubes surfaces. In *Proc. IEEE Visualization '96*, pages 335–342, Los Alamitos, 1996. Computer Society Press.

[171] H.W. Shen. Isosurface extraction in time-varying fields using a temporal hierarchical index tree. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proc. IEEE Visualization '98*, pages 159–164, Los Alamitos, 1998.

[172] H.W. Shen, C. D. Hansen, Y. Livnat, and C. R.Jonson. Isosurfacing in Span Space with Utmost Efficiency (ISSUE). In R. Yagel and G. M. Nielson, editors, *Proc. IEEE Visualization '96*, pages 287–294, Los Alamitos, 1996.

[173] H.W. Shen and D. Kao. Uflic - a line integral convolution algorithm for visualizing unsteady flows. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 317–323, 1997.

[174] W.H. Shen and C.R. Johnson. Sweeping simplices: A fast iso-surface extraction algorithm for unstructured grids. In G. Nielson and D. Silver, editors, *Proc. IEEE Visualization '95*, pages 143–150, Los Alamitos, 1995.

[175] R. Spence, L. Tweedie, H. Dawkes, and H. Su. Visualization for functional design. In *Proc. Int. Symposium on Information Visualization (Info-Vis '95)*, pages 4–9, 1995.

[176] S. Speray and S. Kennon. Volume probes: Interactive data exploration on arbitrary grids. *Computer Graphics (San Diego Workshop on Volume Visualization)*, 24(5):5–12, 1990.

[177] T.C. Sprenger, R. Brunella, and M.H. Gross. H-blob: A hierarchical clustering method using implicit surfaces. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proc. IEEE Visualization 2000*, pages 61–68, 2000.

[178] D. Stalling and H. Hege. Fast and resolution independent line integral convolution. *Proceedings Siggraph '95*, pages 249–256, 1995. Los Angeles.

[179] M. Stark and H. Müller. Variations of the splitting box scheme for adaptive generation of contour surfaces in volume data. In G.M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization*, pages 337–356. IEEE Computer Society, 1997.

[180] J. Stolk and J. van Wijk. Surface particles for 3d flow visualization. In F. Post and A. Hin, editors, *Advances in Scientific Visualization*, pages 119–130. Springer, 1992.

[181] P. Sutton and C.D. Hansen. Isosurface extraction in time-varying fields using a temporal branch-on-need tree (t-bon). In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 147–153, Los Alamitos, 1999.

[182] P.M. Sutton, C.D. Hansen, H.W. Shen, and D. Schikore. A case study of isosurface extraction algorithm performance. In W. de Leeuw and R. van Liere, editors, *Data Visualization 2000*, pages 259–268. Springer, Wien and New York, 2000.

[183] A. Telea and J.J. van Wijk. Simplified representation of vector fields. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 35–42, Los Alamitos, 1999.

[184] H. Theisel. Analyse und Visualisierunghilfe für mehrdimensionale wissenschaftliche Daten. *Informatik Forschung und Entwicklung*, 10(2):91–98, 1995. (in German).

[185] H. Theisel. *Vector Field Curvature and Applications*. PhD thesis, University of Rostock, November 1995.

[186] H. Theisel. Geometric conditions for G^3 continuity of surfaces. *Computer Aided Geometric Design*, 14:719–729, 1997.

[187] H. Theisel. On geometric continuity of isophotes. In A.L.Mehaute, C. Rabut, and L.L. Schumaker, editors, *Curves and Surfaces with Applications in CAGD*, pages 419–426. Vanderbilt University Press, Nashville, 1997.

[188] H. Theisel. Visualizing the curvature of unsteady 2D flow fields. In *Proceedings 9th Eurographics Workshop on Visualization in Scientific Computing*, pages 47–56, 1998.

[189] H. Theisel. Using Farin points for rational Bézier surfaces. *Computer Aided Geometric Design*, 16:817–835, 1999.

[190] H. Theisel. Higher order parallel coordinates. In B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Proc. Vision, Modeling and Visualization 2000*, pages 119–125, Berlin, 2000. Aka.

[191] H. Theisel. On properties of contours of trilinear scalar fields. In P.J. Laurent, P. Sablonniere, and L.L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 99*. Vanderbilt University Press, 2000.

[192] H. Theisel and G. Farin. The curvature of characteristic curves on surfaces. *IEEE Computer Graphics and Applications*, 17(6):88–96, 1997.

[193] H. Theisel and M. Kreuseler. An enhanced spring model for information visualization. *Computer Graphics Forum*, 17(3):335–344, 1998. (Proceedings Eurographics 98).

[194] X. Tricoche, G. Scheuermann, and H. Hagen. A topology simplification method for 2d vector fields. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proc. IEEE Visualization 2000*, pages 359–366, 2000.

[195] I. Trotts, D. Kenwright, and R. Haimes. Critical points at infinity: a missing link in vector field topology. In *Proc. NSF/DoE Lake Tahoe Workshop on Hierarchical Approximation and Geometrical Methods for Scientific Visualization*, 2000. available at http://graphics.cs.ucdavis.edu/ hvm00/program.html.

[196] G. Turk and D. Banks. Image-guided streamline placement. In *Proc. Siggraph '96*, pages 453–460, 1996.

[197] A. van Gelder and J. Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13(4):337–375, 1994.

[198] J. van Wijk. Spot noise: Texture synthesis for data visualization. *Computer Graphics*, 25(4):309–318, 1991.

[199] J. van Wijk and R. van Liere. Hyperslice. In *Proc. IEEE Visualization '93*, pages 119–125, Los Alamitos, 1993. IEEE Computer Society Press.

[200] J.J van Wijk and R. van Liere. An environment for computational steering. In G.M. Nielson, H. Hagen, and H. Müller, editors, *Scientific Visualization*, pages 103–124. IEEE Computer Society, 1997.

[201] C. Ware. *Information Visualization. Perception for Design*. Morgan Kaufmann Publishers, San Francisco, 2000.

[202] R. Wegenkittl and E. Gröller H. Löffelmann. Visualizing the behavior of higher dimensional dynamical systems. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 119–125, 1997.

[203] R. Wegenkittl, H. Löffelmann, and E. Gröller. Fast oriented line integral convolution for vector field visualization via the internet. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 309–316, 1997.

[204] H. Weimer and J. Warren. Subdivision schemes for fluid flow. In *Proc. Siggraph '99*, pages 111–120, 1999.

[205] T. Weinkauf. Krümmungsvisualisierung für 3D-Vektorfelder. Diplomarbeit, University of Rostock, Computer Science Department, 2000. (in German).

[206] D. Weinstein, G. Kindlmann, and E. Lundberg. Tensorlines: Advection-diffusion based propagation through diffusion tensor fields. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. IEEE Visualization '99*, pages 249–253, Los Alamitos, 1999.

[207] R. Westermann, C. Johnson, and T. Ertl. A level-set method for flow visualization. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proc. IEEE Visualization 2000*, pages 147–154, 2000.

[208] J. Wihelms and A.v. Gelder. Octrees for Faster Isosurface Generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.

[209] J. Wilhelms. Visualizing sampled volume data. In Thalmann, editor, *Scientific Visualization and Graphics Visualization*. Wiley, 1990.

[210] 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*, pages 3–33. IEEE Computer Society Press, Los Alamitos, 1997.

[211] P.C. Wong and R.D. Bergeron. Multivariate visualization using metric scaling. In R. Yagel and H. Hagen, editors, *Proc. IEEE Visualization '97*, pages 111–118, 1997.

[212] Y. Zhou, W. Chen, and Z. Tang. An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes. *Computers & Graphics*, 19(3):355–364, 1995.

[213] M. Zöckler, D. Stalling, and H.C. Hege. Interactive visualization of 3d-vector fields using illuminated stream lines. In *Proc. IEEE Visualization '96*, pages 107–113, Los Alamitos, 1996. IEEE Computer Society Press.

Holger Theisel: CAGD and Scientific Visualization
Habilitationsarbeit, eingereicht im Juli 2001

Thesen

1. CAGD (Computer Aided Geometric Design) und wissenschaftliche Visualisierung sind zwei Disziplinen, welche starke Bezüge zur Computergraphik haben, sich ab einer gewissen Zeit aber als eigenständige Gebiete entwickelt haben.

2. Beide Disziplinen haben in relativer Unabhängigkeit zueinander eine Vielfalt von Methoden, Techniken und Anwendungen entwickelt.

3. Die Untersuchung der Anwendbarkeit von Ideen und Methoden der einen Disziplin auf die andere stellt einen hoffnungsvollen Ansatz dar, in beiden Disziplinen Fortschritte zu erzielen.

4. Die Untersuchung der Anwendung von Ideen und Methoden der einen Disziplin in der anderen stellt stellt den Kerngedanken der vorliegenden Arbeit dar.

5. Im CAGD ist nur eine geringere Anzahl von Anwendungen der wissenschaftlichen Visualisierung zu erwarten, da die Daten, die in CAGD vorliegen, nicht explizit in der Visualisierung behandelt werden.

6. Wegen der Ähnlichkeit der behandelten Daten können die meisten Anwendungen von CAGD auf dem Teilgebiet der Strömungsvisualisierung erwartet werden.

7. Da zu Beginn der Entwicklung der wissenschaftlichen Visualisierung das Gebiet des CAGD bereits weitestgehend entwickelt war, ist eine grössere Anzahl von Anwendungen von CAGD-Methoden in der Visualisierung zu erwarten.

8. Da die meisten wichtigen Entwicklungen im CAGD zu einer Zeit gemacht wurden, als die wissenschaftliche Visualisierung als eigenständige Disziplin noch nicht existierte, sind weniger Anwendungen von Visualisierung im CAGD zu erwarten.

9. Die ablaufenden Prozesse in beiden Disziplinen können in Pipelines beschrieben werden. Dies ist ein Standard-Ansatz für die Visualisierung, für CAGD musste eine solche Pipeline eingeführt werden.

10. Die meisten Anwendungen der wissenschaftlichen Visualisierung im CAGD sind in den unteren Teilen der CAGD-Pipeline zu erwarten, während CAGD-Methoden in allen Teilen der Visualisierungspipeline Anwendung finden.

11. Durch die Untersuchung der gegenseitigen Anwendbarkeit von CAGD und wissenschaftlicher Visualisierung können eine Anzahl neuer Ansätze und Techniken für beide Disziplinen gefunden werden. Diese werden in den nachfolgenden Thesen charakterisiert.

12. Eine Klassifikation der Isoflächen eines trilinearen Skalarfeldes nach der Anzahl der Flächenelemente wird eingeführt.

13. Eine Isofläche eines trilinearen Skalarfeldes kann als getrimmte Fläche von rationalen kubischen Flächenstücken exakt modelliert und visualisiert werden.

14. Eine neue Klassifikation für kritische Punkte erster Ordnung von 2D-Vektorfeldern wird eingeführt. Diese Klassifikation basiert auf den Konzepten Rotation und Skalierung in eine Richtung des Vektorfeldes. Basierend auf dieser Klassifikation kann eine Abstandsfunktion auf der Menge aller kritischen Punkte erster Ordnung aufgebaut werden.

15. Ähnlich wie Flächen können Vektorfelder durch geeignete Systeme von Kontrollpunkten modelliert werden. Diese Systeme von Kontrollpunkten werden in stückweise lineare Vektorfelder überführt. Auf diese Weise kann ein 2D-Vektorfeld beliebiger Topologie als stückweise lineares Vektorfeld beschrieben werden.

16. Zur Visualisierung von mehrdimensionalen Daten wird die Technik ShapeVis eingeführt, welche die mehrdimensionalen Daten durch geeignete Freiformflächen Visualisiert. Hierbei wird die gesamte Information eines Datensatzes eindeutig in Ort, Grösse und Form dieser Flächen kodiert.

17. Die Technik der parallelen Koordinaten wird erweitert zu parallelen Koordinaten höherer Ordnung. Durch den Einsatz von Kurven anstelle von Liniensegmenten können Korrelationen von mehr als zwei Dimensionen erkannt werden.

18. Zur Nutzbarmachung von Farinpunkten für Bézierflächen werden mehrere Ansätze gezeigt. Speziell für die eindeutige und intuitive Beschreibung der Gewicht eines Subquadrilaterals für Tensoprodukt-Bézierflächen wird eine geeignete Ikone eingeführt und diskutiert.

Selbständigkeitserklärung

Hiermit erkläre ich, dass die eingereichte Habilitationsarbeit von mir selbständig und ohne fremde Hilfe verfasst wurde. Andere als die angegebenen Hilfsmittel und Quellen habe ich nicht benutzt. Die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen sind als solche kenntlich gemacht.

Weiterhin erkläre ich, dass ich mich nicht zuvor an der Universität Rostock und auch nicht an einer anderen Universität um einen Habilitationsgrad beworben habe.

Holger Theisel
01. Juli 2001

Tabellarischer Lebenslauf von Holger Theisel

16.08.1969	geboren in Jena, Mutter: Ärztin; Vater: Ingenieur
09/76 - 08/84	Polytechnische Oberschule in Jena
09/84 - 08/88	Spezialschule Mathematisch-Naturwissenschaftlicher Richtung "Carl Zeiss", Jena
08/88	Abitur, Notendurchschnitt 1.0, mit Auszeichnung
09/88 - 10/88	Praktikum am Fachbereich Informatik der Universität Rostock
11/88 - 08/89	Wehrdienst
09/89 - 08/94	Studium der Informatik an der Universität Rostock, Vertiefungsgebiete Computergraphik und Mathematik
10/91-11/91	Praktikum an der Enter AG, Zürich, Schweiz
08/94	Diplom, Universität Rostock, Thema: Automatische Auswahl geeigneter Visualisierungstechniken für allgemeine wissenschaftliche Daten, Note 1.0, mit Auszeichnung
10/94 - 09/95	Visiting Scholar an der Arizona State University, Computer Science Department, Tempe, USA; Zusammenarbeit mit Prof. Gerald Farin; unterstützt durch ein Stipendium der Studienstiftung des deutschen Volkes
seit 10/95	wissenschaftlicher Assistent am Fachbereich Informatik der Universität Rostock; Arbeit in der Gruppe Computergraphik von Frau Prof. Heidrun Schumann
01/96	Promotion, Universität Rostock, Thema: Vector Field Curvature and Applications, Prädikat summa cum laude
03/99 - 04/99	Gastdozentur an der Universidad Central de Las Villas, Cuba, gefördert durch Deutschen Akademischen Austauschdienst (DAAD)

Appendix A

Color Images

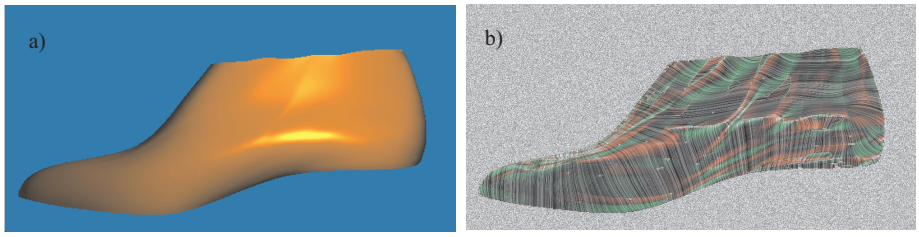


Figure A.1: (color version of figure 1.1): a) piecewise bicubic B-spline surface; b) visualizing one class of lines of curvature using methods of flow visualization.

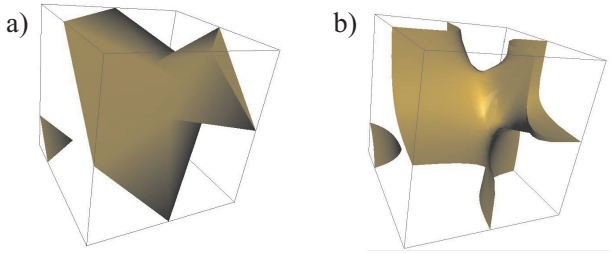


Figure A.2: (color version of figure 1.2): a) piecewise triangular approximation of an isosurface of a trilinear volume data set using Marching Cubes; b) computation of the exact isosurface as trimmed piecewise rational cubic surface.

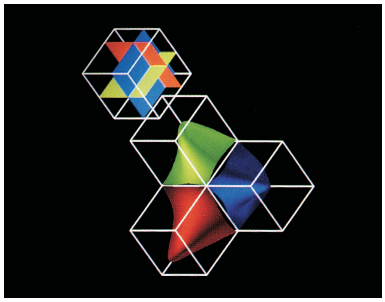


Figure A.3: (color version of figure 3.1): 2D slices for a point of interest in the volume; its three 2D scalar fields are visualized as height fields; the point of interest can be moved interactively (from [143]).

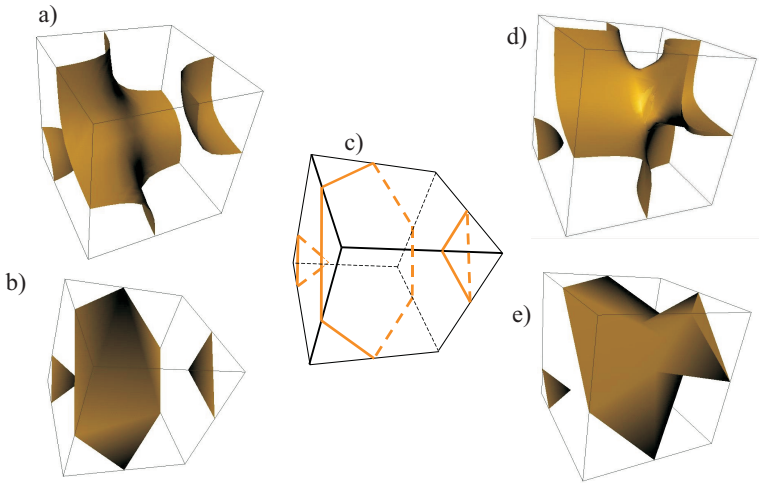


Figure A.4: (color version of figure 3.19): a) and d): two possible contours of (3.4) and (3.5) where the MC algorithm of [130] and [139] gives the same set of closed polygons on the faces of the cell shown in c); depending on certain inner points, the exact triangulation is either b) or e).

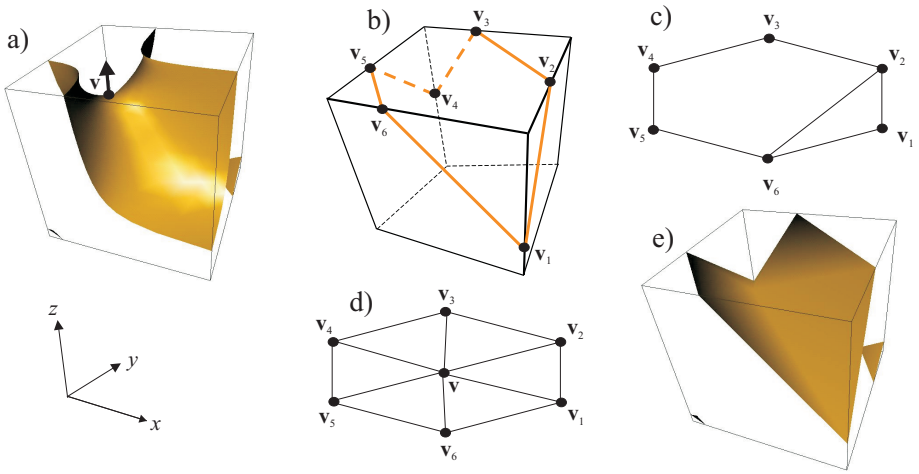


Figure A.5: (color version of figure 3.20): a) contour and point $\mathbf{v}$ with normal in $z-$direction on it; b) closed polygons resulting from the MC algorithm of [130] and [139]; c) part of a wrong triangulation of b); d) triangulation applied here; e) triangulation of d) in 3D.

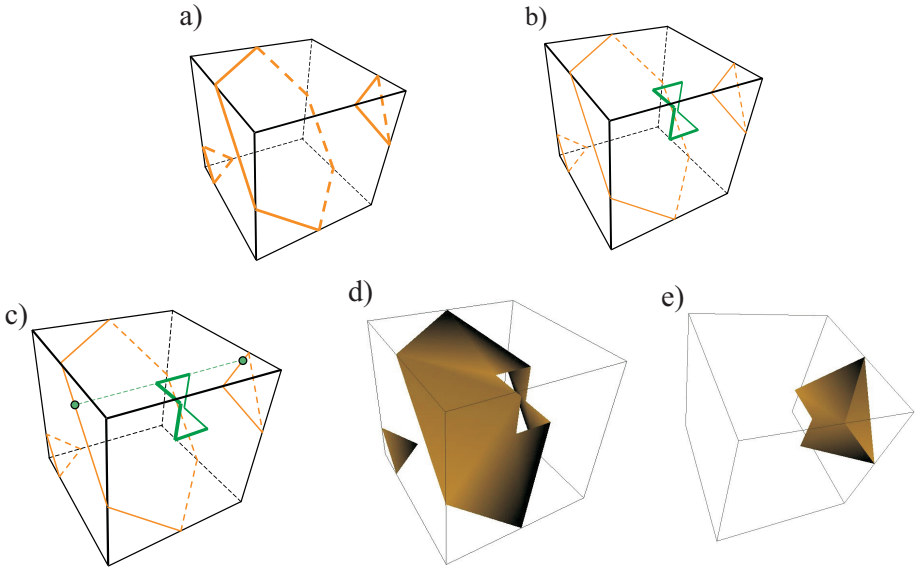


Figure A.6: (color version of figure 3.21): Illustration of a topologically exact MC algorithm; a) create outer rings following [130] and [139]; b) compute inner ring; c) check connectivity between inner ring and outer rings by intersecting the lines of the inner ring with all faces of the cell; here the inner ring is connected to two outer rings; d), e) triangulate the areas between inner ring and outer rings.

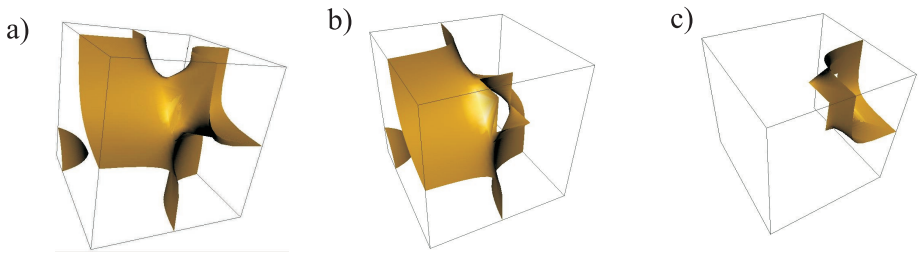


Figure A.7: (color version of figure 3.22): a) Exact contour over the triangulation shown in figure 3.21d) and 3.21e); b) exact contour over triangulation in figure 3.21d); c) exact contour over triangulation in figure 3.21e). We can clearly see that inner ring is part of the contour.

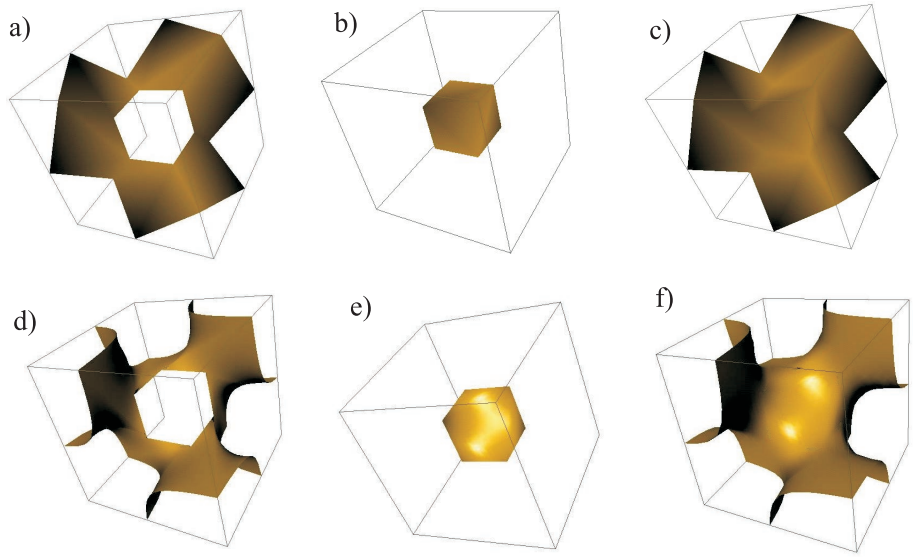


Figure A.8: (color version of figure 3.23): Example of a contour with one outer ring consisting of 12 edges and the inner ring being completely inside the cell; a) triangulation between inner ring and outer ring; b) triangulation of inner ring; c) whole triangulation; d)-f) exact contours over the triangulations a)-c).

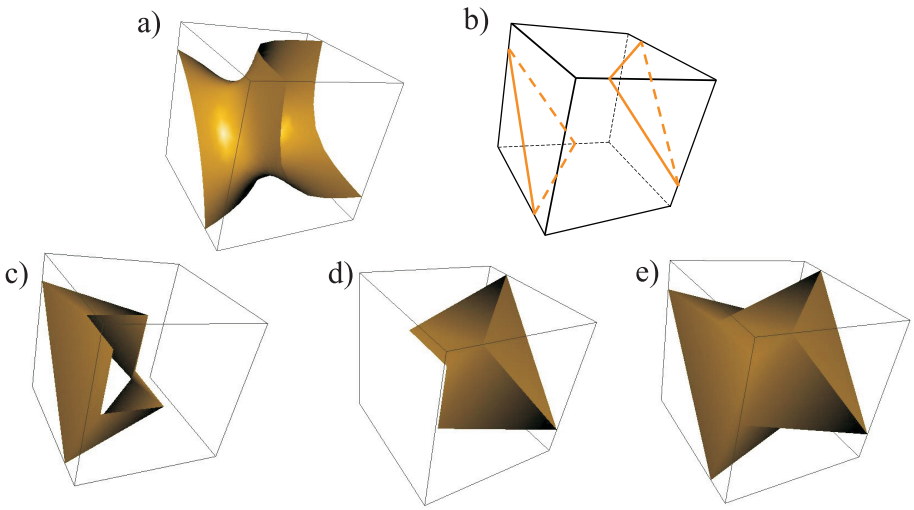


Figure A.9: (color version of figure 3.24): a) Contour which gives two outer rings and the inner ring completely inside the cell; b) outer rings; c) triangulation between inner ring and one outer ring; d) triangulation between inner ring and the other outer ring; e) whole triangulation.

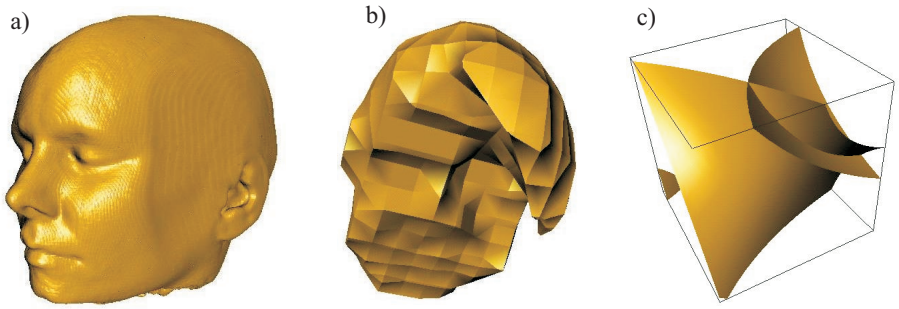


Figure A.10: (color version of figure 3.25): a) CT head consisting of 423.963 triangles - a candidate for mesh reduction algorithms; b) detail inside the same CT head - the triangular mesh is too coarse; c) the contour of a triquadratically interpolated scalar field may have self- intersections and complicated topologies.

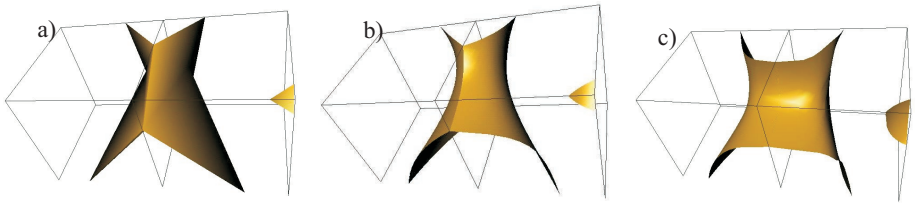


Figure A.11: (color version of figure 3.27): a) two cells and the triangular approximation of a certain contour using MC; b) the exact contour represented by a number of trimmed surfaces of rational cubic triangular patches; c) the global G^1 modification of the contour without changing its topology.

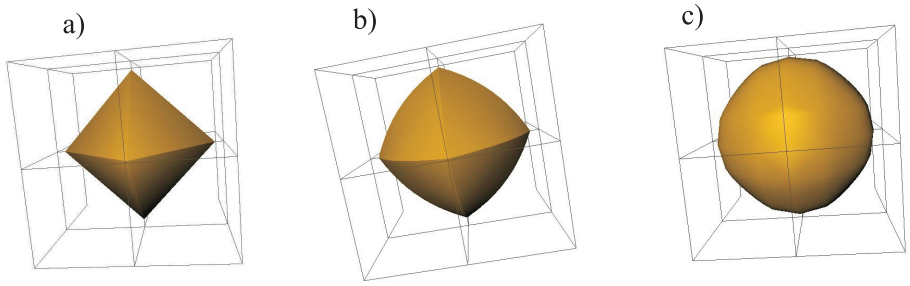


Figure A.12: (color version of figure 3.33): Scalar field $s(x, y, z) = x^2 + y^2 + z^2$, sampled by a $3 \times 3 \times 3$ grid in the domain $[-1, 1]^3$, $r = 0.9$; a) Marching cubes; b) exact contours; c) globally G^1 contours.

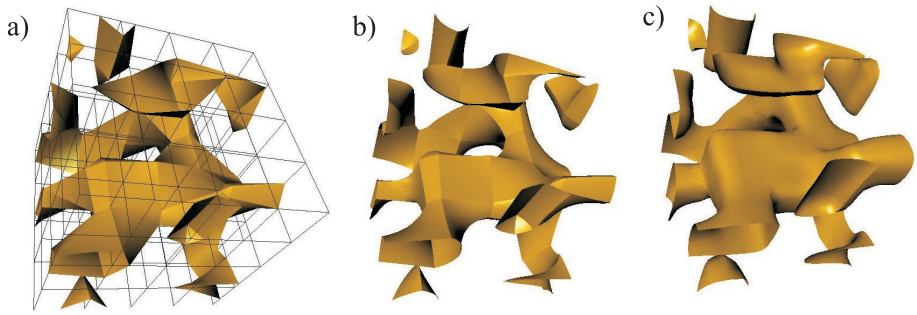


Figure A.13: (color version of figure 3.34): $5 \times 5 \times 5$ random volume data set; a) Marching cubes; b) exact contours; c) globally G^1 contours.

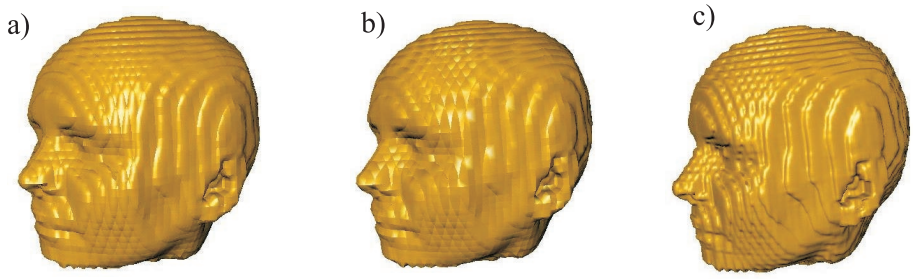


Figure A.14: (color version of figure 3.35): Downsampled data set of figure 3.25a ; a) Marching cubes; b) exact contours; c) globally G^1 contours.

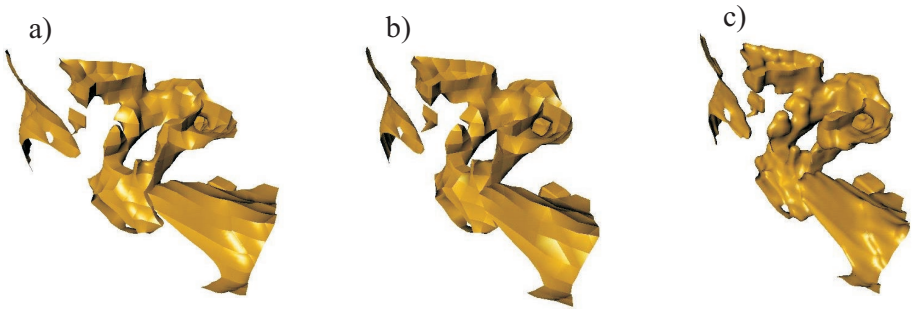


Figure A.15: (color version of figure 3.37): Inner detail of the data set of figure 3.25a; a) Marching cubes; b) exact contours; c) globally G^1 contours.

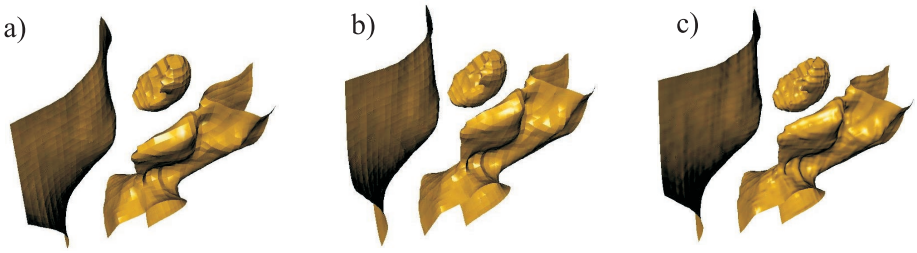


Figure A.16: (color version of figure 3.38): Inner detail of the data set of figure 3.25a; a) Marching cubes; b) exact contours; c) globally G^1 contours.

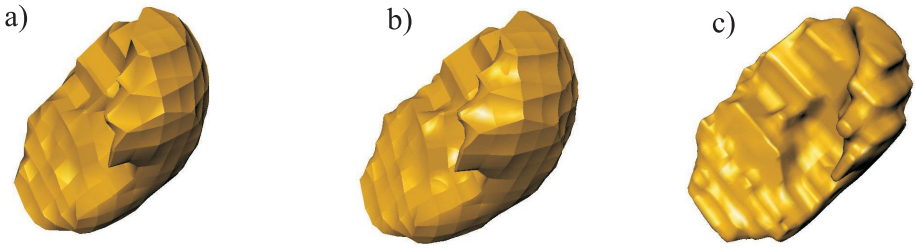


Figure A.17: (color version of figure 3.39): Detail of figure 3.38; a) Marching cubes; b) exact contours; c) globally G^1 contours.

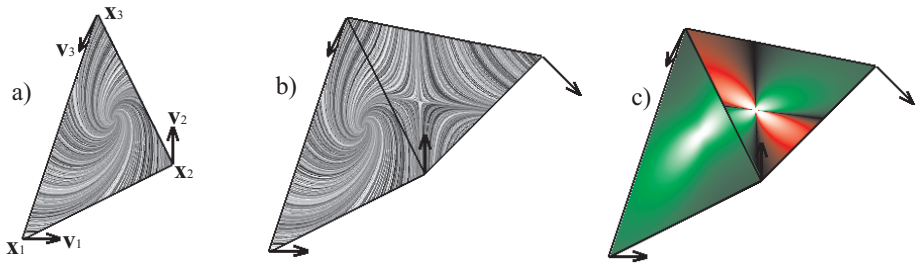


Figure A.18: (color version of figure 4.20): a) linear vector field inside the triangle $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$; b) piecewise linear vector field consisting of two domain triangles; c) curvature plot of b) reveals that the tangent curves in b) are not curvature continuous.

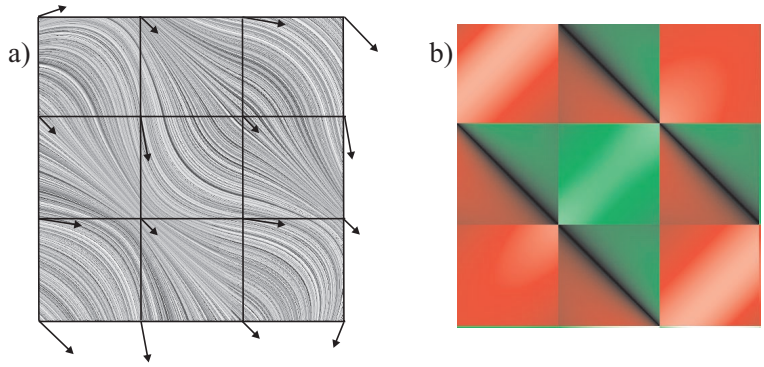


Figure A.19: (color version of figure 4.24): Piecewise bilinear vector field on a regular 4×4 grid; a) Integrate and Draw; b) curvature plot.

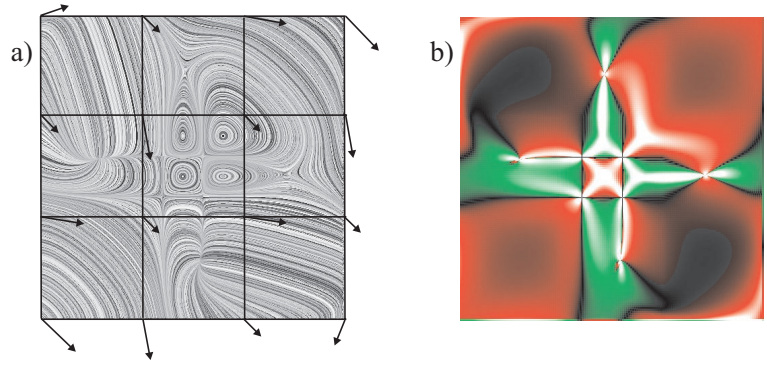


Figure A.20: (color version of figure 4.25): Piecewise biquadratic vector field on a regular 4×4 grid; a) Integrate and Draw; b) curvature plot.

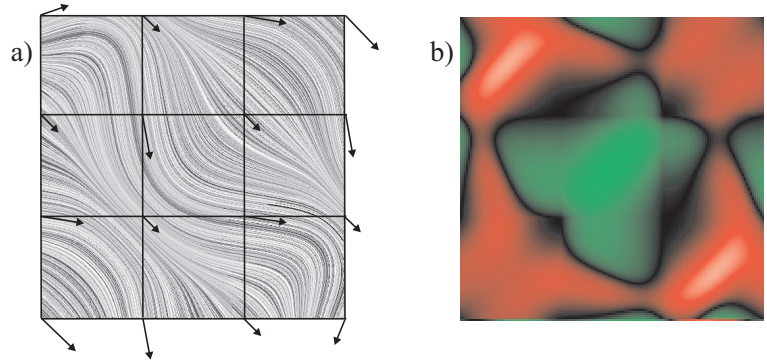


Figure A.21: (color version of figure 4.26): Piecewise bicubic vector field on a regular 4×4 grid; a) Integrate and Draw; b) curvature plot.

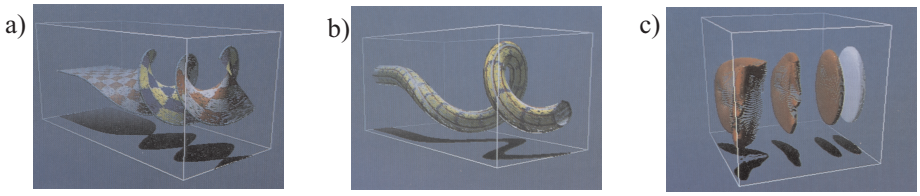


Figure A.22: (color version of figure 4.28): a) stream surface; b) stream tube; c) stream objects; (from [180]).

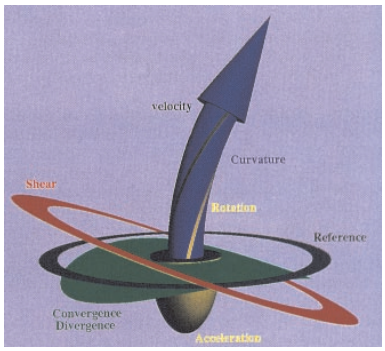


Figure A.23: (color version of figure 4.30): A local probe which represents – among other measures – the curvature of the tangent curve in the selected locations of the vector field (from [44]).

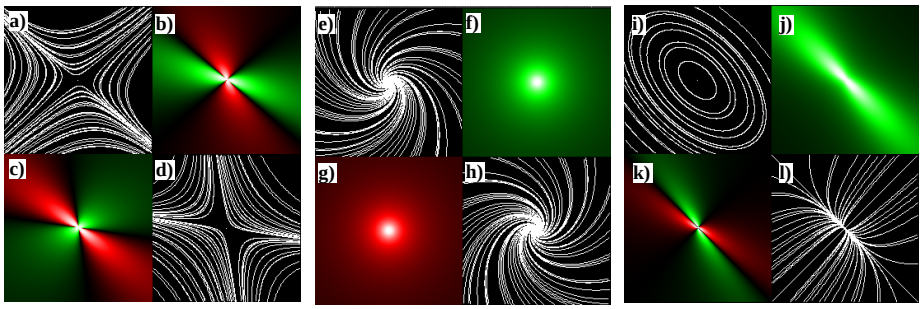


Figure A.24: (color version of figure 4.33): Linear vector field with saddle point (a..d); linear vector field with repelling focus (e..h); linear vector field with center (i..l).

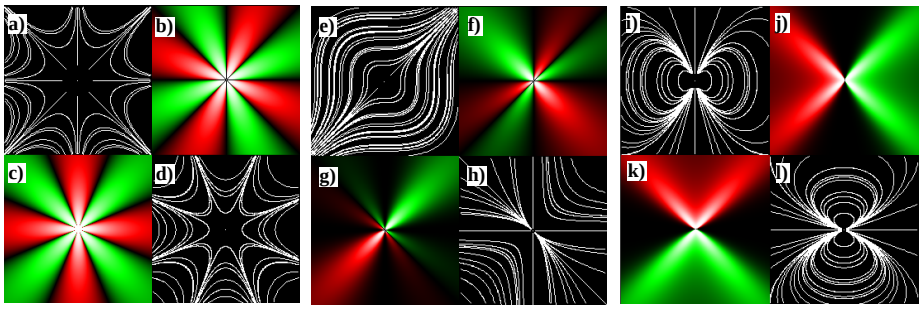


Figure A.25: (color version of figure 4.34): Higher order saddle point (a..d); critical point with two elliptic sectors (e..h); dipole (i..l).

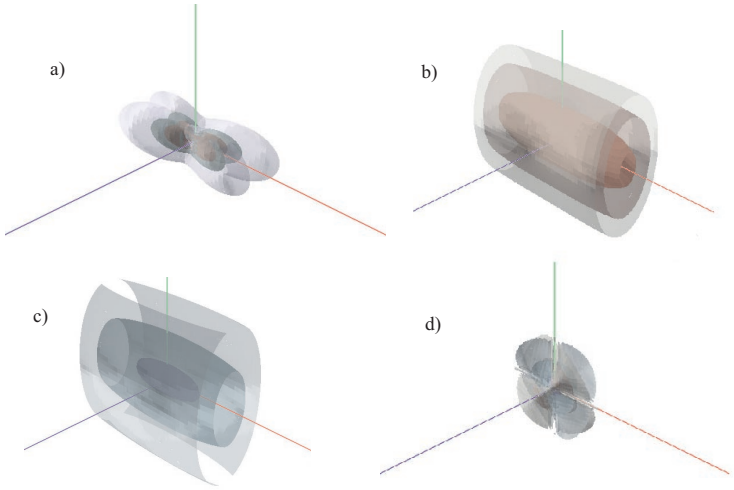


Figure A.26: (color version of figure 4.35): Linear vector field with a first order critical point of the type (RN,RN,RN); a) curvature $\kappa(\mathbf{v})$; b) Gaussian curvature $K(\mathbf{v})$; c) Mean curvature $H(\mathbf{v})$; a) torsion $\tau(\mathbf{v})$; (from [205]).

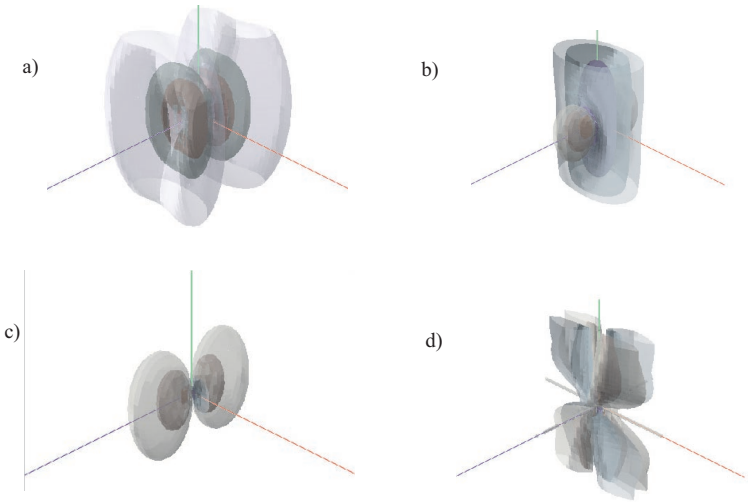


Figure A.27: (color version of figure 4.36): Linear vector field with a first order critical point of the type (AN,Sa,Sa); a) curvature $\kappa(\mathbf{v})$; b) Gaussian curvature $K(\mathbf{v})$; c) Mean curvature $H(\mathbf{v})$; a) torsion $\tau(\mathbf{v})$; (from [205]).

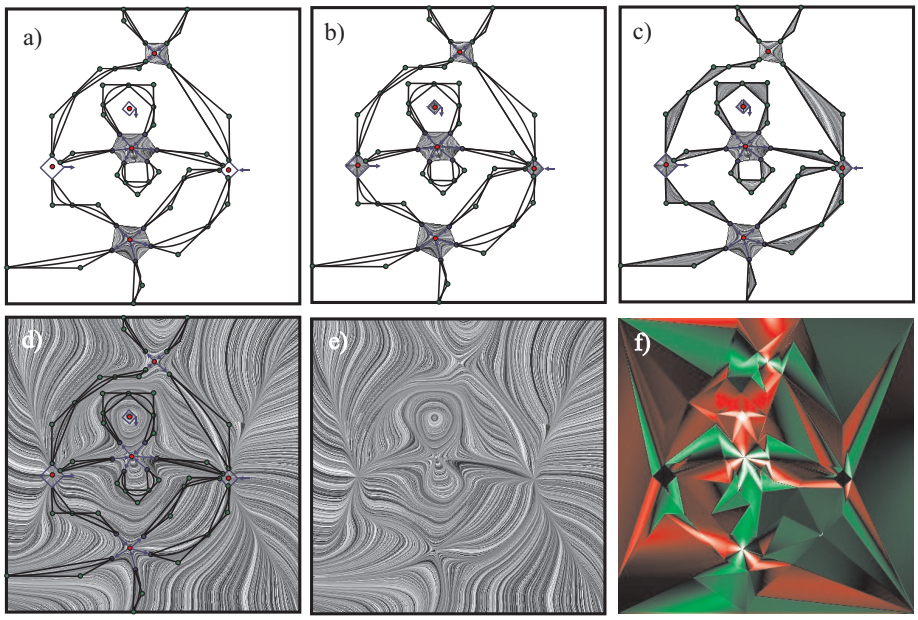


Figure A.28: (color version of figure 4.45): Constructing the piecewise linear vector field for the topological skeleton of figure 4.39; a) construct piecewise linear vector field for general critical points; b) construct piecewise linear vector field for first order critical points of index +1; c) construct piecewise linear vector field for separatrices; d) Delaunay triangulate remaining parts and apply piecewise linear interpolation; e) final vector field consists of 79 vertices and 138 triangles; f) curvature plot of e).

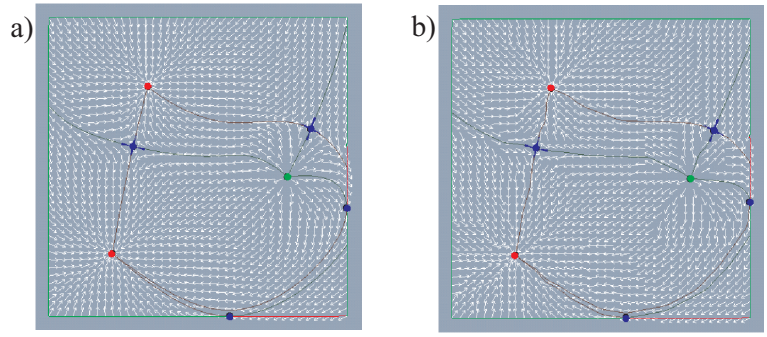


Figure A.29: (color version of figure 4.46): a) test data set (4.77) on a regular 38 x 38 grid; b) compressed version of the same topology, compression ratio 90%; (images from [128]).

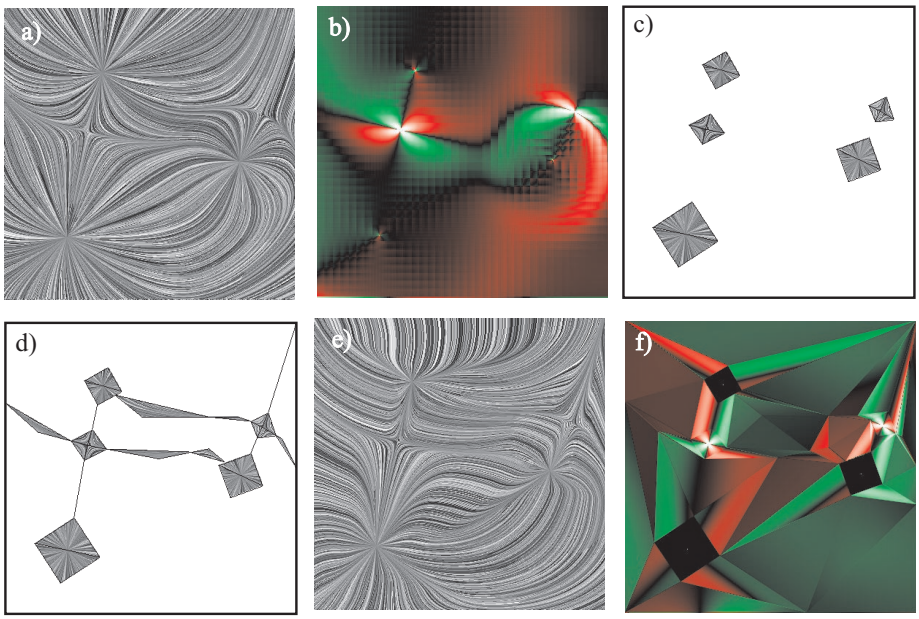


Figure A.30: (color version of figure 4.47): a) vector field (4.77) on a 38 x 38 regular grid: 1444 grid points; b) curvature plot of a); c) remodeling the critical points as piecewise linear vector field; d) remodeling the separatrices as piecewise linear vector field; e) complete remodeled piecewise linear vector field consists of 40 vertices and 68 triangles: compression ratio 95%; f) curvature plot of e).

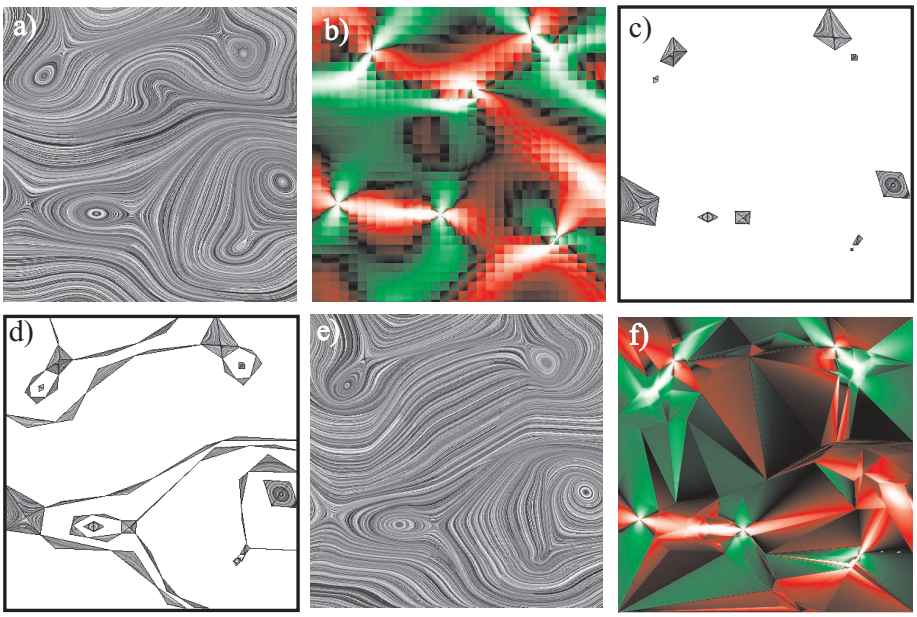


Figure A.31: (color version of figure 4.48): a) fragment of the vector field shown in figure 4.27 on a 34 x 34 regular grid: 1056 grid points; b) curvature plot of a); c) remodeling the critical points as piecewise linear vector field; d) remodeling the separatrices as piecewise linear vector field; e) complete remodeled piecewise linear vector field consists of 124 vertices and 226 triangles: compression ratio 79%; f) curvature plot of e).

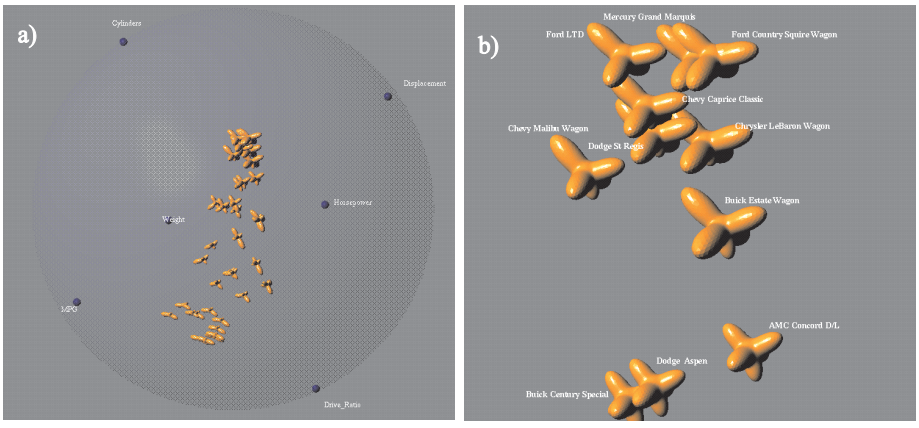


Figure A.32: (color version of figure 5.9): Visualization of the car data set (38 observation cases, 6 dimensions); a) overview; b) zoom into the cluster.

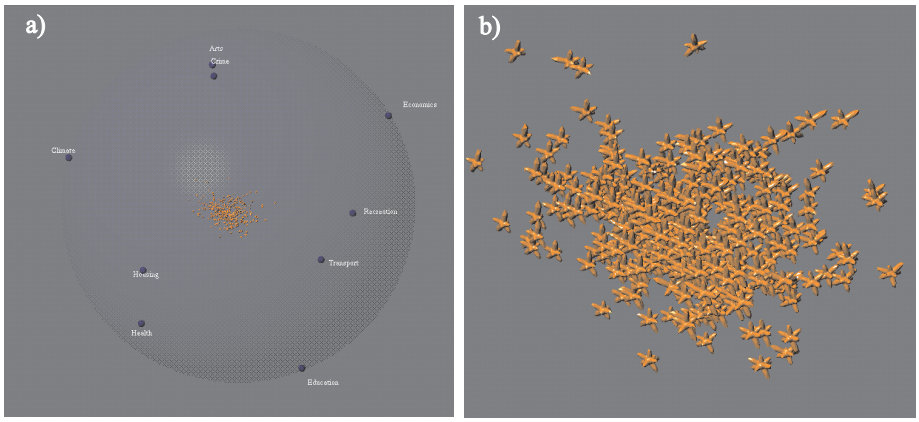


Figure A.33: (color version of figure 5.10): Visualization of the city data set; a) overview; b) zoom into the cluster.

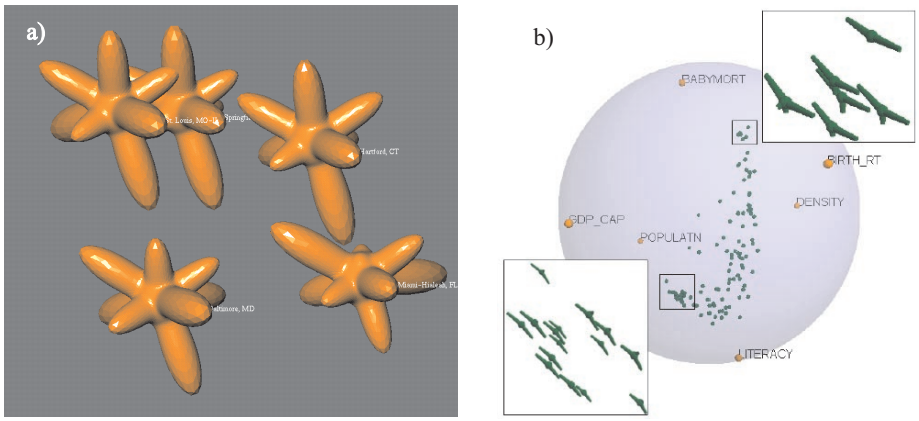


Figure A.34: (color version of figure 5.11): a) city data set - zoom of the 5 objects lower right in figure 5.10b; b) 6-dimensional demographic data set using simplified ShapeVis (image from [121]).

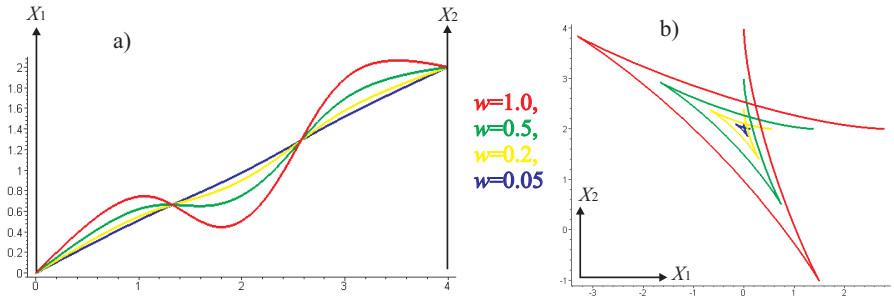


Figure A.35: (color version of figure 5.28): Influence of the weight w; a) higher order parallel coordinates; b) dual curve of a) in cartesian coordinates.

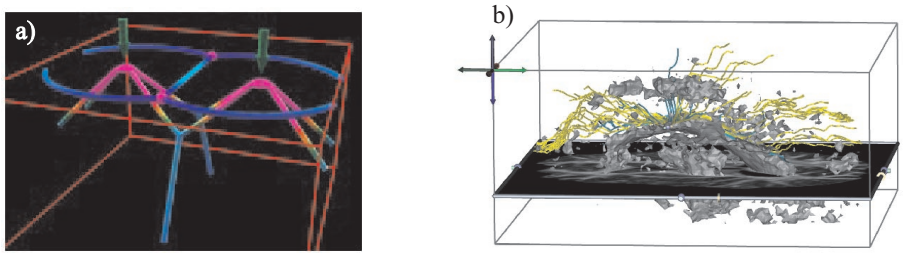


Figure A.36: (color version of figure 6.2): a) minor hyperstreamlines for a stress tensor field (image from [123]); b) tensor lines (yellow) and hyperstreamlines (cyan) for a diffusion data set (image from [206]).

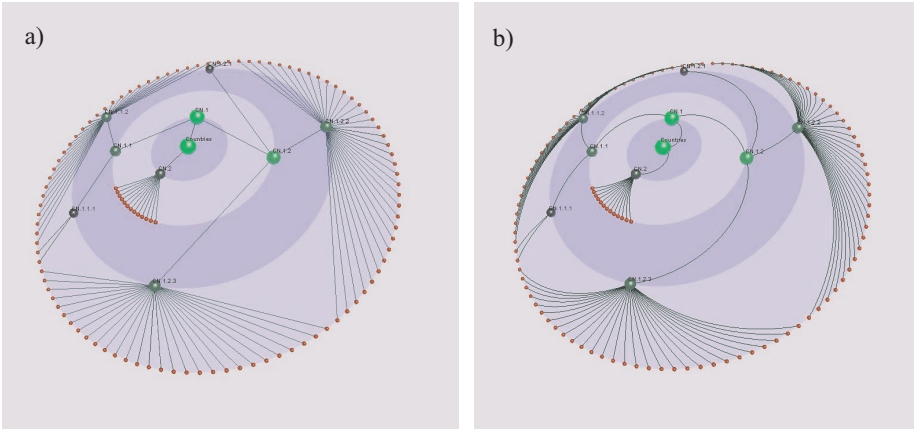


Figure A.37: (color version of figure 6.3): Magic Eye View for hierarchical data; the rings on the sphere represent the levels of hierarchy; a) edges of the graph represented as straight lines; b) edges of the graph represented as rational quadratic Bzier curves.

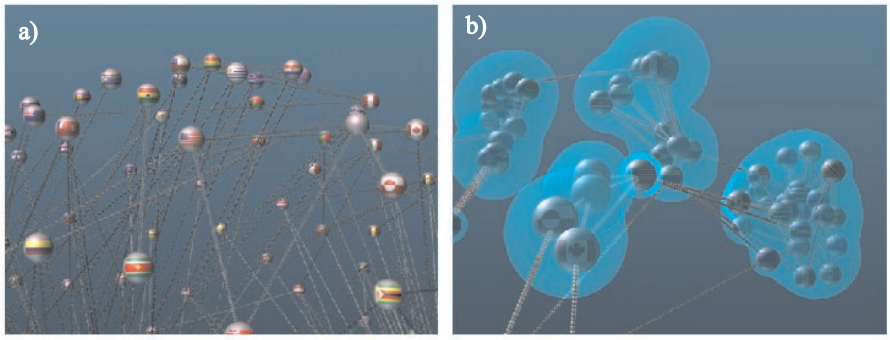


Figure A.38: (color version of figure 6.4): Blobs; a) initial object layout; b) Blob surfaces denoting the clusters (images from [177]).

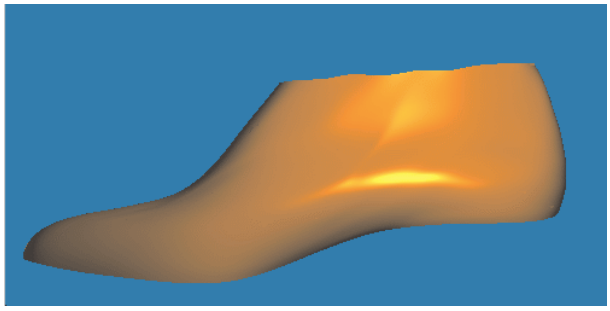


Figure A.39: (color version of figure 7.25): Raytracing a surface; shown is a bicubic B-spline surface consisting of 15×10 patches.

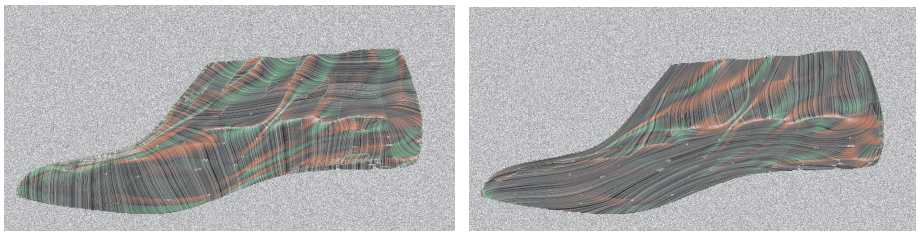


Figure A.40: (color version of figure 7.26): Visualization of the two families of lines of curvature using a combination of Integrate&Draw and curvature plot.

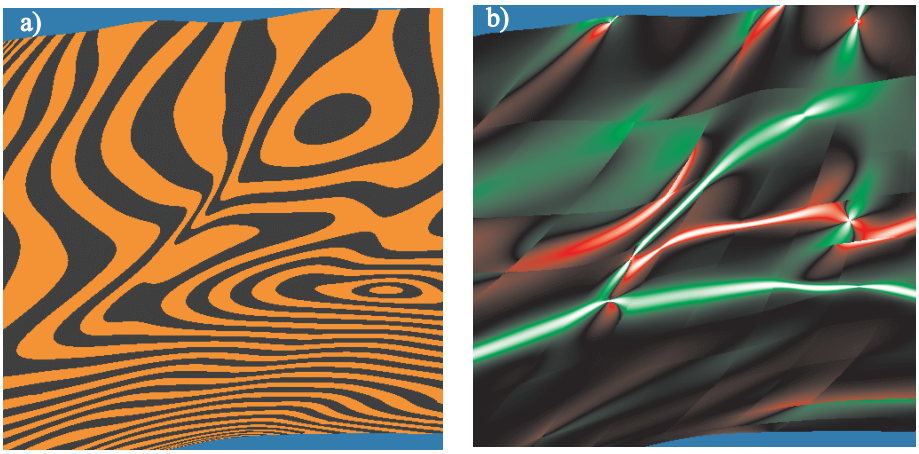


Figure A.41: (color version of figure 7.27): A class of isophotes on a fragment of the surface of figure 7.25; b) visualization of the geodesic curvature of the isophotes (images from [192]).