ALEXANDER SCHULZE

TOPOLOGY-BASED VECTOR FIELD RECONSTRUCTION

DIPLOMA THESIS

Visual Computing Group
Department of Simulation and Graphics
Faculty of Computer Science
Otto-von-Guericke University Magdeburg

Advisor:
Dipl.-Ing. Alexander Kuhn Dipl.-Ing. Dirk Joachim Lehmann

Assessor:
Prof. Dr.-Ing. habil. Holger Theisel

Hand in date:
April 7, 2012

Matriculation number:
17 26 09

ABSTRACT

This diploma thesis is addressed to the reconstruction of vector fields from their topological structures. In this case, only three-dimensional and parameter independent fields are considered.

Reconstruction means that only with the help of its topological structures, every vector within the field can be computed. This involves two proceedings depending on whether the field is global or local. Each has its own approach and chapter.

The subsequent evaluation shows how well and under which circumstances the approaches can be used for a reconstruction.

This work includes the reconstruction of vectors as well as of so-called *Separation Curves* from critical points of the vector field. Boundary Switch Curve are also taken into account as topological structures. Additionally, it is demonstrated how the information of vectors from one to another point in the domain can be interpolated and diffused.

ZUSAMMENFASSUNG

Diese Diplomarbeit beschäftigt sich mit dem Thema der Rekonstruktion von Vektorfeldern aus ihren topologischen Strukturen. Dabei wird ausschließlich auf dreidimensionale und parameterunabhängige Vektorfelder eingegangen.

Rekonstruktion bedeutet dabei, dass so gut wie möglich aus seinen topologischen Strukturen das Ursprungsvektorfeld errechnet wird. Das Vektorfeld kann lokal oder global vorliegen, beides führt zu unterschiedlichen Rekonstruktionsansätzen, denen jeweils ein Kapitel gewidmet ist.

Abschließend werden die vorgestellten Verfahren und ihre Ergebnisse evaluiert, um zu zeigen inwieweit und unter welchen Bedingungen sie für die Rekonstruktion eingesetzt werden können.

Wichtige Einzelthemen sind die Rekonstruktion von Vektoren und die Konstruktion von sogenannten *Separation Curves* aus kritischen Punkten des Vektorfelds. Berücksichtigung finden als topologische Strukturen auch *Boundary Switch Curves*. Zusätzlich wird gezeigt wie, Vektorinformationen an einem Ort des Definitionsbereichs an andere Punkte interpoliert und diffundiert werden können.

DANKSAGUNG

Den Abschluss meines Studentenlebens stellt diese Diplomarbeit dar. Sie ist das Ergebnis vieler Stunden Arbeit, aber vor allem wäre sie nicht möglich geworden, ohne die tatkräftige Unterstützung meiner beiden Betreuer Dipl.-Ing. Alexander Kuhn und Dipl.-Ing. Dirk Joachim Lehmann. Unsere Diskussionen gaben mir einen großen Einblick in das Thema der Vektorfelder. Dafür gilt ihnen mein Dank.

Bei meinen Freunden, die sich diese Arbeit durchgelesen und wertvolle Verbesserungsvorschläge gegeben haben, möchte ich mich ebenso bedanken, wie bei meinen Eltern, die mich seit fast 30 Jahren in Allem unterstützen.

Zu guter Letzt eine kurze Erwähnung von musikalischen Künstlern, die (unbewusst) für mich den Soundtrack zu dieser Arbeit geschrieben haben: Fehlfarben (*Ich nicht verstehen*), EA80 (*Die Jahre vergehen*) und Zwei Tage: Ohne Schnupftabak (*Abgehakt*).

CONTENTS

1 INTRODUCTION 1
- 1.1 Motivation 1
- 1.2 Goals 2
- 1.3 Tasks 2

2 RELATED WORK 3
- 2.1 2D Construction 3
- 2.2 3D Construction 4
- 2.3 2D Reconstruction as Approximation 6
- 2.4 2D and 3D Reconstruction with Support Vectors Learning 7
- 2.5 Conclusion 9

3 THEORY 11
- 3.1 Basic Notations 11
- 3.2 Vector Fields and Their Topological Structures 11
  - 3.2.1 Critical Points 13
  - 3.2.2 Boundary Switch Curves 15
  - 3.2.3 Separatrices and Saddle Connectors 16
- 3.3 Classification of the reconstruction 17
  - 3.3.1 Global vs. Local 17
  - 3.3.2 Interpolation vs. Approximation 17
  - 3.3.3 Conclusion 18
- 3.4 Interpolation 18
  - 3.4.1 Inverse distance weighted interpolation 18
  - 3.4.2 Trilinear Interpolation 19
  - 3.4.3 Reverse Trilinear Interpolation 21
  - 3.4.4 Reverse Interpolation from Critical Points 21

4 GLOBAL RECONSTRUCTION OF VECTOR FIELDS 23
- 4.1 One Sink and One Source 23
- 4.2 Reconstruction with Critical Points 24
- 4.3 Reconstruction with Boundary Switch Curves 26
- 4.4 Parameters 27
- 4.5 Conclusion 28

5 LOCAL RECONSTRUCTION OF VECTOR FIELDS 29
- 5.1 Preconditions and Input 29
- 5.2 One Sink and One Source in One Cell 29
- 5.3 General Approach 31
- 5.4 Reconstruction of Cells with Topology 31
- 5.5 Reconstruction of Separation Lines 33
- 5.6 Diffusion of Vector Information 36
- 5.7 Conclusion 38

6 IMPLEMENTATION OF A PROTOTYPE 39
- 6.1 Used Software 39

6.2 Implemented Features 40
6.2.1 Algorithms and Reconstruction 40
6.2.2 Visualisation 40
6.2.3 Input and Output 42
6.3 Prototype 44
6.4 Results 46
7 EVALUATION OF THE RESULTS 47
7.1 Introduction and Expectations 47
7.2 Global Reconstruction 48
7.2.1 Test data 48
7.2.2 Results for the GCS data set 49
7.2.3 Results for the VRTD data set 49
7.2.4 Parameters and Runtime 50
7.3 Local Reconstruction 50
7.3.1 Test data 51
7.3.2 Results for the 5CP data set 51
7.3.3 Results for the Perlin data set 52
8 CONCLUSION 59
8.1 Summary of The Results 59
8.2 Interpretation and Comparison 59
8.3 Limitations and Perspectives 60
A APPENDIX 61
A.1 Distribute points equally on sphere surface 61
A.2 Equations 63

BIBLIOGRAPHY 65

1

INTRODUCTION

All processes of hydrodynamics, whether in an ocean or in a glass of water, whether in gases or in fluids, they can be described by and analysed with vector fields. This is done to improve the properties of the substance and the objects which are circulated around. It comes with a great impact of data to store vector fields explicitly. On the other hand, much effort has been done to reveal and to reduce the fields to their important properties.

This work of reducing is well-known. But the opposite direction, generating the original vector field from the reduced properties, still needs some attention.

The diploma thesis at hand provides in chapter 2 an insight into the related work for reconstructing vector fields and establishes in chapter 3 the theoretical backgrounds. The following two chapter, 4 and 5, deal with the global and local reconstruction of vector fields. To put the results of both reconstructions into life, chapter 6 presents the implementation of a prototypical application which handles the reconstruction. The seventh chapter (7) evaluates the reconstruction with regards to proper vector field data. Chapter 8 concludes the thesis.

1.1 MOTIVATION

Speaking of vector fields, they can allocate an huge amount of data if they are stored explicitly. There exist different approaches to lossy compress vector fields. For example, omitting all explicit vectors and instead storing only critical points and other topological data. These still contain the most important information of the field. The opposite direction of decompressing the whole vector field needs special consideration as this involve reconstruction of non-existing vector information.

Taking a vector field with a complex topological structure, one can try to simplify it. Grouping first-order critical points to one higher-order if they are located in a certain radius is one possibility. The result of this simplification must be translated back to vector information in the domain. This needs again a reconstruction to see if the changing of the topology has fundamentally changed the vector field. Comparing of the vectors can reveal such unwanted modifications of the field.

Reconstruction of a vector field can also be thought as a part of a construction. This is discussed in the chapter related work. A user can place certain topological information in the domain which is the first and interac-

tive step in the construction. This is followed by the reconstruction from these information. Reconstruction can be seen as a part of the solution to the construction of vector fields.

This all motivates to the reconstruction of vector fields by their topology.

1.2 GOALS

The goal of this work is to develop a prototype of an application which can reconstruct vector fields. This prototype bases on prior theoretical work presented in this thesis. It is necessary to allow importing and exporting of well-established data formats like Amira or predefined plain formats.

In addition to support the process of development and evaluation of the reconstruction, it is convenient to visualise certain aspects of it. The performance of the reconstruction should be also assessed with the help of vector field visualisation. This includes for example to contrast the original with the reconstructed vector field.

1.3 TASKS

This thesis works on three main tasks. The first is to fully understand the context of the work. That means to repeat or to acquire the theoretical aspects of analytical geometry and of vector fields. Chapter 2 is the result of these efforts. On the other hand, the first task includes also to familarise with existing approaches in the domain of reconstruction of vector fields. With this accumulated knowledge the theoretical consideration of a topology-based reconstruction of vector fields should be done.

The second task is to integrate and implement the theoretical consideration of the reconstruction in a framework which will serve as a software tool and library. The reconstruction should work autonomously, an interaction with the user is only needed by choosing the input data and deciding where to store the output. Nevertheless, an interactive part should be made where the user can explore and compare the results of the reconstruction. This will combine the framework with graphical user interface frameworks and possibilities to draw in three dimensions, both need to be chosen from existing frameworks.

The final task for this thesis is to evaluate the results of the previous tasks. In order to have not only build an application which can produce a vector field, it must been shown how good the application fullfils this task in terms of reconstruction. This needs test data sets which are used to reconstruct a vector field. Also they must come with the original vector field which allows a quantitative measurement of reconstruction.

2

RELATED WORK

There are mainly two approaches in building up vector fields from topological data. The first is the construction in which the vector field will be interactively created by the users. The second is the reconstruction which works in parts autonomously in contrast to the construction. This chapter gives a short summary about the two approaches and also introduces four related works.

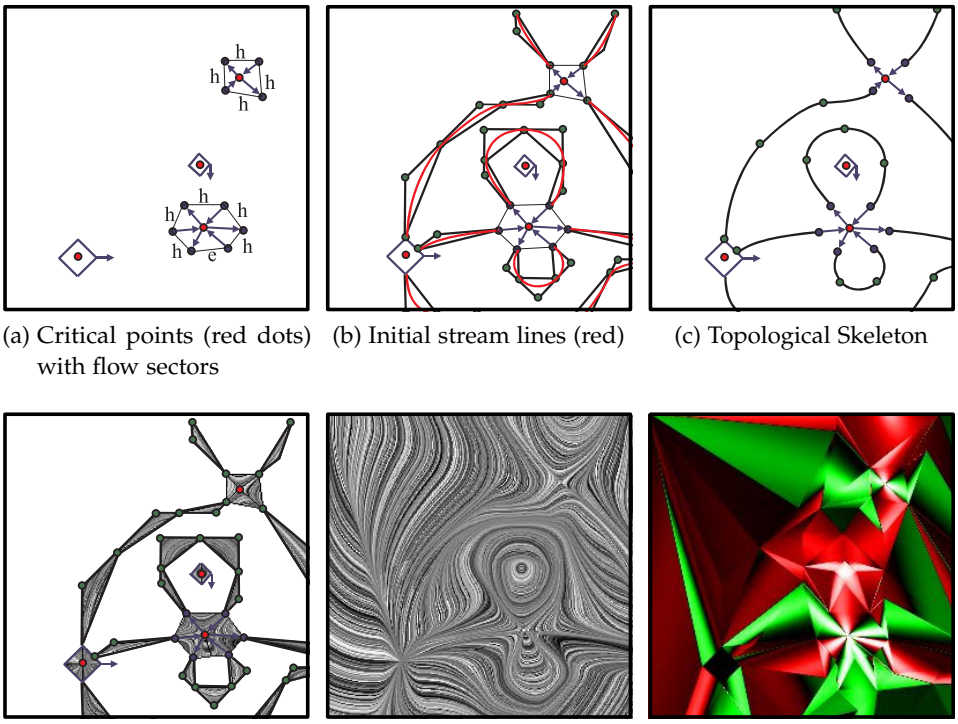
2.1 2D CONSTRUCTION

One of the first works that deals with this concept of constructing two-dimensional vector fields can be found in [11] and it works with two-dimensional vector fields only. The construction is split into two parts. As the topological skeleton is the most significant feature the first step reduces the construction of a vector field to the construction of its skeleton which contains the critical points of the field (see 3.2 for details to topological structures of a vector field). The user will begin with placing control polygons at a plane in whose centre is located a critical point. These control polygons must describe the different sections of flow around the point, see Figure 1a, so the user has to state the flow direction at the vertices of the polygon. The other part of the topological skeleton are the separatrices which can be thought as border curves that separate regions of different flow. They are also modeled by control polygons in a way that exactly one curve can be interpolated within these as pictured in Figure 1b with the red lines. Multiple polygons connect to some kind of a chain which start and end at critical points to form a continuous curve.

After this the interactive part of modeling the topological skeleton is done and it follows the computation of the vector field. The polygons of the critical points and the separatrices can be triangulated as well as the remaining free space. In each of those triangles a linear interpolation will be performed, resulting in a piecewise linear vector field covering the domain. Figure 1e shows the final vector field in an *Integrate And Draw* visualisation which gives a clue how the stream lines of the field unfolds. It is also easy to spot the analogy between modeled critical points and the critical behaviour of the stream lines. Figure 1f on the other hand shows the curvature plot of the constructed vector field.

The main motivation for building those vector fields is the simplification and compression of already existing ones. Simplification should produce

a new vector field which has separated important from less important properties keeping only the first ones. Compression on the other hand should produce a copy of the original vector field which has both the same important properties and a much smaller amount of data. One can also detect the educative effect: an user not only gets passively a vector field by a measurement but interactively affects the result of the measurement. The user is the creator of the field. It can give rich insights in the nature of two-dimensional vector fields if the users explores what happens if he or she puts two sinks together. This critical point experiment also leads to a limitation of the approach. It does not assure that it will not create other critical points. For example the placement of two sinks together will surely create other critical points in their proximity.



(a) Critical points (red dots) with flow sectors

(b) Initial stream lines (red)

(c) Topological Skeleton

(d) Linear interpolation within constructed triangles

(e) Overall vector field rendered with *Integrate and Draw*

(f) Curvature plot (scale from dark green-green-white-red-dark red)

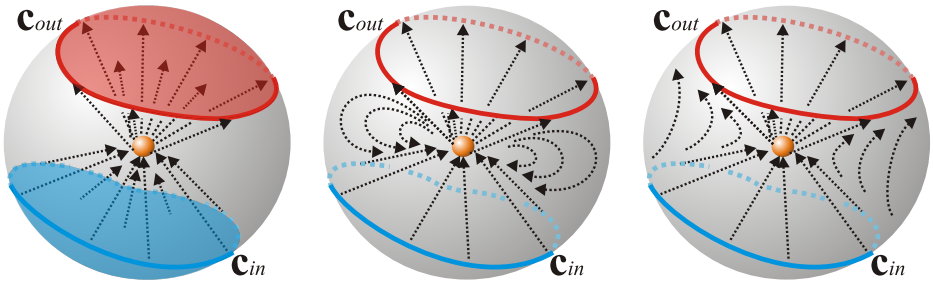
Figure 1: Steps of constructing a 2D vector field (taken from [11])

2.2 3D CONSTRUCTION

A pulling up of the previous technique into the three-dimensional space can be found in [15] which explicitly refers to the previous work of Theisel. Weinkauf et al. state it would be the first algorithm for constructing 3D vector fields based on their topological skeleton.

While the approach of designing 3D vector fields guarantees to contain the user's critical points, however, it cannot prevent the occurrence of new critical points. This may be clear at first sight if we put put two sinks (cf.

3.2) in a vector field. There must be other critical structures which make the flow around the critical points consistent, they cannot exist alone. The paper explains this with the indices of the critical points whose sum is typically almost zero. Another reason for the appearance of new critical points can be found in the approach itself. The tetrahedrisation of the remaining areas or the choice of the vectors at the bounding box can be inappropriate albeit the authors state that theses two problems can be overcome by a more sophisticated strategy.



(a) Parabolic inflow (blue) and outflow (red) sectors

(b) Elliptic flow sector in the middle

(c) Hyperbolic flow sector in the middle

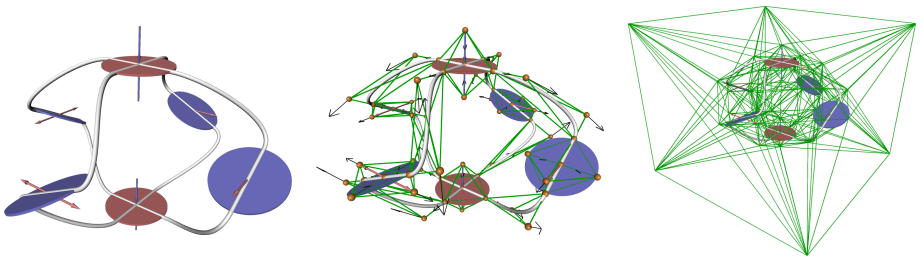
Figure 2: Three types of flow sectors depicted by a sphere icon representing an high-order critical point (taken from [15])

The construction of a three-dimensional vector field is again implemented by user-based interaction. Likewise to the previous work the starting point is to define the high-order critical points and their flow sections. As mentioned in [3], there exist three types of flow section around two- and three-dimensional critical points. They are separated by special stream lines called separatrices. The flow sections characterize a critical point. Figure 2 depicts the three types of flow sections, see section 3.2.1 for a detailed explanation of high order critical points and their flow sections. This iconised depiction is introduced by Weinkauf et al. to visualise a high-order critical point for the first time. The sphere model supports also the design of the flow sections as they can be simply drawn on the surface of the spheres.

After all critical points are modeled they need to be connected. Defining those connectors is done by linking an outflow section of one critical point to the inflow section of another critical point. In general, the connection between critical points does not constitute as straight lines. That is why the approach allows to define some points with vectors which the connector should interpolate in location and direction of their vectors. The connectors are modeled as a piecewise C^1 continuous cubic spline. See Figure 3a for an illustration of a final topological skeleton with six critical points and eight connectors.

The user can place arbitrary stream lines where there are no or not enough topological data. They will serve as support for the construction of the vector field. This concludes the modeling of the vector field.

After the interactive modeling of the topological skeleton the vector field is constructed in an automatic way. At first, to get a piecewise linear vector field the domain needs to be tetrahedrisated. Each vertex of the tetrahedron net is assigned a corresponding vector in order to compute a piecewise trilinear interpolation within them. The tetrahedrisation starts in the regions around critical points.



(a) Final topological skeleton (b) Tetrahedrisation of the skeleton (c) Complete tetrahedrisation

Figure 3: From topological skeleton to final triangulation (all taken from [15])

The regions around the connectors are subsequently tetrahedrisated. As the connectors are piecewise cubic curves for each piece of it, a tetrahedron is made whose vertices match the nodes of the cubic curve. This cannot be done if the curve is nearly planar: the curve is then split into two quadratic curves giving the chance to construct two piecewise linear vector fields for both parabolas. Also if there are tetrahedra from critical points or connectors which intersect each other, the cubic or quadratic curves are subdivided resulting in newly subdivided tetrahedra. This subdivision is done until no tetrahedra intersect each other.

Finally, a bounding box is set to contain the topological skeleton and to limit the final tetrahedrisation of the domain. The approach uses a Delaunay tetrahedrisation whose final result can be seen in Figure 3c. To assign to the vertices of each tetrahedron a vector (if none is present), the weighted average of the vectors of all adjacent vertices is computed. Having done this for each tetrahedron a linear vector field can be computed.

2.3 2D RECONSTRUCTION AS APPROXIMATION

In [5] an algorithm for 2D vector field reconstruction is presented. It uses as input a set of points and their corresponding vectors of the original vector field, called sparse samples. The algorithm mainly depends on an approximation with least squares with two extensions to improve the result. It tries to find a vector field that approximates each support samples best. As output a 2D vector field depending on polynomial functions is generated. These functions are bivariate and have a fixed degree d which can be controlled by the user.

The first step of the algorithm performs its reconstruction as a local one. It solves the problem with the help of the *classical least square* method. This

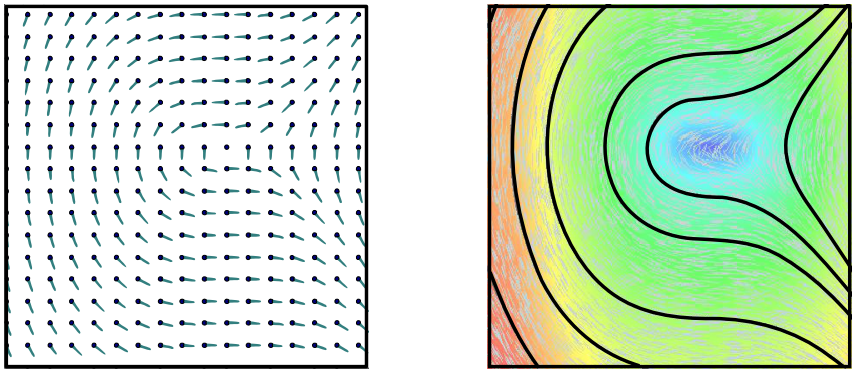
approximation can be improved if the derivatives at the given sample points are known, too. This part is called *acceleration fitting*. As both approaches may tend to numerical instability the authors apply also a *ridge regression* technique which stabilises the solution in a numerical sense. These three parts are interlocked by two user-defined parameters which control their application. Namely if there are no derivatives given, the acceleration fitting cannot be used. In a similar way the ridge regression technique can be controlled, even disabled.

The following second step is to lift up the local approximation to a global one. The local one performs efficiently on small sets of data, it would be counterproductive to adapt it in a global sense. As the input data is scattered, the authors first apply an *adaptive domain subdivision* in order to tune the level of detail on which the previously described least square method can operate. This is done by an adaptive quadtree decomposition in that way that each cell does not contain more points that would be enough for defining a polynomial function of degree d. Each cell that contains enough points will be subdivided if the maximum level of subdivision is not exceeded. This parameter $l_{\max}$ is also user-defined. The reconstruction works then adequately in each cell. For a global reconstruction these separated, local ones are combined by a *multiresolution partition of unity* (MPU). It blends the local approximations to a global vector field which guarantees a smooth behaviour. This blending takes only supporting vector samples being in a certain region around the point which is going to be reconstructed. The authors enable another user-defined parameter with the choice of a kernel function for the MPU. They control the influence of the support regions. The combining of all methods leads to a *global approximation evaluation*. To evaluate the vector field at any point of the region the quadtree is traversed in order to find all support regions that contain that point from which the vector field approximation is computed.

Lage et al. give some results of their algorithms which show that the approximation error depends on the polynomial degree and even to a greater extent on the subdivision level. Changing the latter parameter from 0 to 5 will roughly improve the result by factor 10 but further increasing yields no more significant changing. They also show how the ridge regression will prevent the vector field from being too much perturbated when the solution of the least square method is numerical unstable. Applications are given by evaluation of Jacobian matrices, computing integral curves in the domain of the reconstructed vector field and approximating the acceleration field.

2.4 2D AND 3D RECONSTRUCTION WITH SUPPORT VECTORS LEARNING

In [6] the reconstruction of vector fields is described and solved in means of machine learning. The authors have chosen *support vector machines* (SVM) as they appear to be "the most robust" in statistical learning. Adequately to the approach in the previous section, sparse and non regular input values are used in this work as well. The resulting vector field is global and



(a) Initial samples for the reconstruction

(b) Result of the reconstruction with black integral curves

Figure 4: Input and output of the two-dimensional reconstruction, icons in the left picture encode location, direction and magnitude of the vectors, the colours in the right picture encode length of vectors with blue is the shortest and red the longest (both taken from [5])

differentiable, and approximates the given input samples as the authors state.

The learning of the vector field, i. e. the reconstruction, is done for two and three dimensions in learning either the Cartesian or the polar and spherical respectively representation. For example in the 2-d Cartesian case vectors are represented by a (x, y) tuple which defines their positions according to both coordinate axis. On the other hand, in the 2D polar case vectors are represented by a (r, θ) tuple where r represents the length of the vector and θ the deviation to a coordinate axis. The actual reconstruction is strongly abstracted from terms of vector fields as it handles only streams of numbers.

The learning is done by means of a ε-SVR which stands for *support vector regression* and is a learning machine for "solving multidimensional scalar value prediction and estimation problems". SVR uses a training set of the input vectors to build up a affine predication function which is subject to an optimisation. This results in a predication function that approximates values to the searched function best. The reconstruction of a vector field implies a transformation of the vector field into two (in three dimensions into three) predication functions, each for one dimension. They are created with a part of the input samples. These functions are optimised with the rest of the input giving an approximation for the vector field that fits to the input data.

Due to its underlying system this reconstruction offers three parameters that can be controlled by the user. The first parameter that gave the solving method its name is ε and adjusts smoothness and quality of the approximation as two oppositional options. The second parameter is the penalizing constant P which aims for larger values to fit the training data more what is precarious for the prediction at other locations. The used

kernel function is the last parameter. It controls inherently the support vector regression.

Marcos Lage et al. state that their approach gives statistically stable results which are reliable in both dimensionalities. Due to the used machine learning method the obtained vector field is unique and globally optimal.

2.5 CONCLUSION

To sum up the four approaches of this chapter table 1 shows a comparison of important aspects for each approach. The header line of the table refers to each section of this second chapter. The first feature is dimensionality. The second approach explicitly refers to the first in the way that it is the extension to the third dimension. The third approach is only two dimensional but the authors state they plan to extend it to three dimensions. From the four presented approaches only the last reconstruction algorithm is applicable to two and three dimensions. Due to the triangulation and tetrahedrisation in the two first approaches they work in a local sense, i. e. the constructed vector field within one triangle or tetrahedron respectively, has nothing to do with the one in any other. The two latter approaches produce vector fields where the changing of the input in one region of the domain can affect any other region, thus the reconstructions work at a global scope.

The two first approaches were developed to model a vector field by constructing its topology. This first step is followed by a non-interactive construction of the vector field. One can imagine that providing a measured topology can be used to *re*construct a vector field. On the other hand the latter two algorithms are solely intended for executing the reconstruction. If one could model the input of the algorithms they could also be used for a construction. The user must only be able to define samples, i. e. the vectors at certain locations of the vector field.

Finally, again the four presented works fall apart in two categories. The two former aim to interpolate the given topology data, that is to have the critical points and stream lines at the location that the user have modeled. The two approaches by Marcos Lage et al. are per definition only an approximation. In general the input data cannot be found in the resulting vector field but it is at best approximated.

APPROACH	2.1	2.2	2.3	2.4
Dimensionality	2-d	3-d	2-d	2-d & 3-d
Scope	Local	Local	Global	Global
Reconstruction?	Construction	Construction	Reconstr.	Reconstr.
Topology aware	Yes	Yes	No	No
Exact?	Interpolation	Interpolation	Approx.	Approx.

Table 1: The table sums up the features of the four presented approaches

Of course, these four approaches are not the only ones. The first two were chosen because they are strongly related to this diploma thesis as they deal with a topological construction. After a skeleton is made up the two approaches build autonomously a vector field. The two latter approaches are interesting to contrast the two former as they merge calculus with machine learning, especially approach 4. This proves how fertile it can be to mix different fields of computer science to get a new look at an old problem.

There are other possibilities in constructing and reconstructing vector fields which are not covered within this thesis. The features which the resulting approach of this thesis should cover are: reconstruct by interpolation a three-dimensional vector field by its topology in a global and local sense. Before deepening this, general explanations to the theory of vector fields need to be set out.

3

THEORY

The diploma thesis deals mainly with vector fields and interpolation. The following chapter will give an overview of the concepts which are needed for the rest of this course.

3.1 BASIC NOTATIONS

In this thesis scalars and functions are written as small italic letters (s, f) whereas points and vectors are typeset in small bold letters (**v**). Vector fields also used this notation. To denote matrices capital bold letters are used (**M**).

3.2 VECTOR FIELDS AND THEIR TOPOLOGICAL STRUCTURES

A (parameter-independent) *vector field* **v** is a function that maps a vector $\mathbf{x} \in \mathbb{R}^m$ to a point $\mathbf{p} \in D \subset \mathbb{R}^n$ in space: $\mathbf{v} : D \subset \mathbb{R}^n \Rightarrow \mathbb{R}^m$. Throughout this work holds $m = n = 3$ and only parameter-independent vector fields are considered so that we speak of *three dimensional, parameter independent vector fields*. The vector field can be written as in Equation 3.1 where u, v and w are certain trivariate functions and $x, y, z \in \mathbb{R}$.

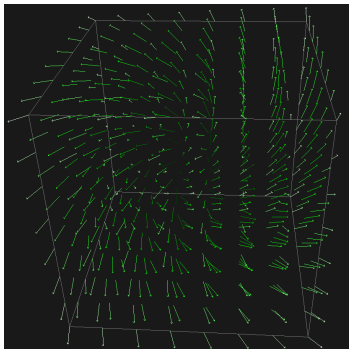
$$\mathbf{v}(\mathbf{x}) = \mathbf{v}(x, y, z) = \begin{pmatrix} u(x, y, z) \\ v(x, y, z) \\ w(x, y, z) \end{pmatrix} \tag{3.1}$$

For example, Figure 5 shows the equation for an analytic vector field. The two images give an illustration on how the vectors of this field behave, they do a helical movement along an axis. Analytic means here that for every point in the domain a vector can be computed. This is not true for measured vector fields as in fluid mechanics. For these fields exists only a finite amount of samples either structured like on a regular grid or unstructured. The samples serve as input for interpolating vectors at unknown points.

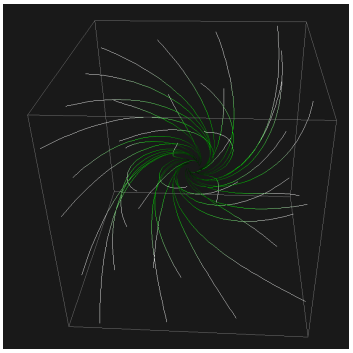
If we look at the vectors at each point as an information of flow, i. e. the vector gives direction and value of a movement at this point, we can build lines of movement for massless particles in the flow which is described by the vector field **v**. These lines are called *stream lines* or tangent lines

$$\mathbf{v}(\mathbf{x}) = \begin{pmatrix} x - y \\ x + y \\ 2 \cdot z \end{pmatrix}$$

(a) Equation



(b) Vector samples



(c) Streamlines

Figure 5: Equation, vector samples and stream lines of an analytic vector field (images made with [2])

as the vectors at these points where the lines go through are also their tangent vectors. Those lines or better curves are solutions of an autonomous ordinary differential equation system, cf. [14]. One important property of stream lines is that given two of them, they do not intersect in one point, except this one is a critical point of $\mathbf{v}$. That also means, given a non-critical point in a vector field $\mathbf{v}$ exact one stream line passes there. In general, one cannot find a parametric description of a stream line.

Stream lines are a very useful visualisation tool because of that non-intersecting property. The two right images in Figure 5 compare the depiction of only vectors versus only stream lines. The stream lines give a much better impression of the vector field behaviour than only the vector samples. With the colour coding for the stream lines it is also possible to infer the direction of the flow. The vector samples do this by all the small arrows.

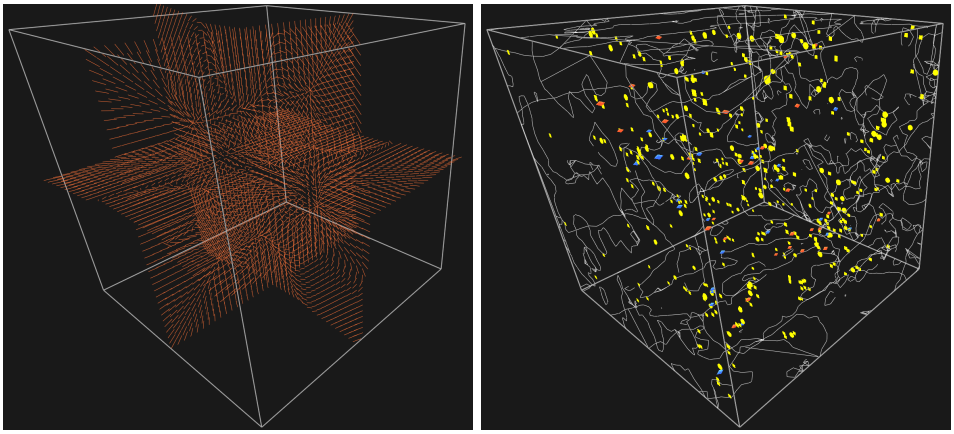


Figure 6: Showing some of the vectors of a field (left) and in contrast only the critical points and boundary switch curves (right) as topological properties. Both images show also the boundary of the domain as wire frame cube in white (Images made with JFlowVisFW)

What are topological structures? Now as there is a definition for vector fields one could try to visualise them. The naive and brute force method

is to draw an arrow at every point of the field in order to visualise the vector information there. This can be an unsatisfying solution as there is too much data to visualise it properly and, of course, efficiently. The viewer cannot interpret the result. To overcome this dilemma one tries to abstract the data, i. e. to build up a topological skeleton which only represents "interesting" parts of the vector field and reduces the visual redundancy. These parts include: *Critical Points*, *Boundary Switch Curves*, *Saddle Connectors*, *Separatrices* starting from the first two ones, *Boundary Switch Connectors*, *Isolated Closed Stream Lines*. This thesis deals only with the first three due to its constrictive length. For details to boundary switch connectors see [16].

3.2.1 *Critical Points*

Every point $\mathbf{p}$ of a vector field where its vector vanishes, i. e. where $\mathbf{v}(\mathbf{p}) = 0$, is called a *critical point*. As there also could be critical curves or critical surfaces where the vector field vanishes, it is also demanded that for a critical point the neighborhood of that point is not null: $\mathbf{v}(\mathbf{p} \pm \varepsilon) \neq 0$ for some ε-vector with positive length. If we take the (parameter-independent) vector field as a flow field those points are stationary islands in that flow. A particle at those points will not be moved by the flow. Every critical point can be assigned a matrix called Jacobian. It contains the first-order derivatives, see Equation 3.2 for the formal notation of the Jacobian of a three dimensional vector field. The variable $\mathbf{c}$ denotes the critical point and the variables u, v and w denote the three trivariate functions which make up the vector field. As this is an analytical definition for simulated or measured vector fields there exist assessments for the Jacobian.

$$\mathbf{J}(\mathbf{c}) = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix} = \begin{pmatrix} \frac{d}{dx}\mathbf{u}(\mathbf{c}) & \frac{d}{dy}\mathbf{u}(\mathbf{c}) & \frac{d}{dz}\mathbf{u}(\mathbf{c}) \\ \frac{d}{dx}\mathbf{v}(\mathbf{c}) & \frac{d}{dy}\mathbf{v}(\mathbf{c}) & \frac{d}{dz}\mathbf{v}(\mathbf{c}) \\ \frac{d}{dx}\mathbf{w}(\mathbf{c}) & \frac{d}{dy}\mathbf{w}(\mathbf{c}) & \frac{d}{dz}\mathbf{w}(\mathbf{c}) \end{pmatrix} \tag{3.2}$$

There are first-order critical points $\mathbf{x}$ for which the determinant of the corresponding Jacobian matrix $\mathbf{J}(\mathbf{x})$ does not vanish: $\det(\mathbf{J}(\mathbf{x})) \neq 0$. Those critical points can be classified by an analysis of the eigenvalues of the Jacobian. Taking the real part of the eigenvalues, there are four different characterizations possible, see Table 2. In general the are more positive eigenvalues the more inflow does exist. These four cases can be subdivided into eight when taking the imaginary part of the eigenvalues into account, see Table 3. If some eigenvalues have also an imaginary part the in- or outflow becomes a swirling movement. In two dimensions there exists also the type *centre* for a critical point which will not be part of the considerations here. This is a point which is surrounded by closed stream lines making it literally an island in the flow.

Besides the relative simple first-order critical points, there exist higher-order critical points which are characterized by an arbitrary amount of

KIND OF CRITICAL POINT	CONDITION FOR EIGENVALUES
Sink	$0 < \Re(\lambda_1) \leqslant \Re(\lambda_2) \leqslant \Re(\lambda_3)$
Attracting Saddle	$\Re(\lambda_1) < 0 < \Re(\lambda_2) \leqslant \Re(\lambda_3)$
Repelling Saddle	$\Re(\lambda_1) \leqslant \Re(\lambda_2) < 0 < \Re(\lambda_3)$
Source	$\Re(\lambda_1) \leqslant \Re(\lambda_2) \leqslant \Re(\lambda_3) < 0$

Table 2: A categorization for first-order critical points according to the real part of the eigenvalues of their Jacobian matrix (from [14]).

KIND OF CRITICAL POINT	CONDITION FOR EIGENVALUES
Node	$\Im(\lambda_1) = \Im(\lambda_2) = \Im(\lambda_3) = 0$
Foci	$\Im(\lambda_1) = 0$ and $\Im(\lambda_2) = -\Im(\lambda_3) \neq 0$

Table 3: A categorization for first-order critical points according to the imaginary part of the eigenvalues of their Jacobian matrix. The eigenvalues are in no particular order (from [14]).

sectors of flow around them. See Figure 7 for an example. There can be three different flows: hyperbolic, elliptical and parabolic. This complexity cannot be handled by the eigenanalysis of the corresponding Jacobian matrix as its determinant is 0 (or not defined?). The characterization likewise to the one of the first-order critical points is also not possible. Instead one uses an explicit enumeration and localization of the different flow sectors.

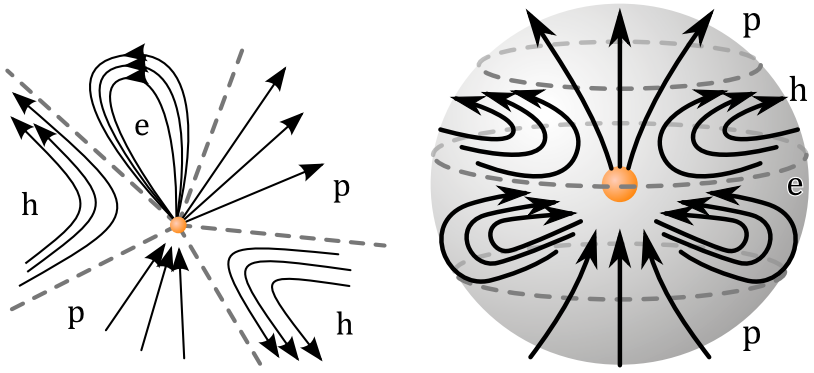


Figure 7: Three different flow sections around a critical point: in two dimensions (left) and in three dimensions (right): *p* parabolic, *e* elliptic, *h* hyperbolic (left inspired by [11], right inspired by [15]

Because of the different flow sectors [15] proposed an icon-based visualisation for high-order critical points in three dimensions. Section 2.2 gives an overview. Each first-order critical point can be considered as a higher-order critical point. In that sense sinks and sources have only one parabolic flow sector. Saddles contain two hyperbolic flow sections in three dimensions.

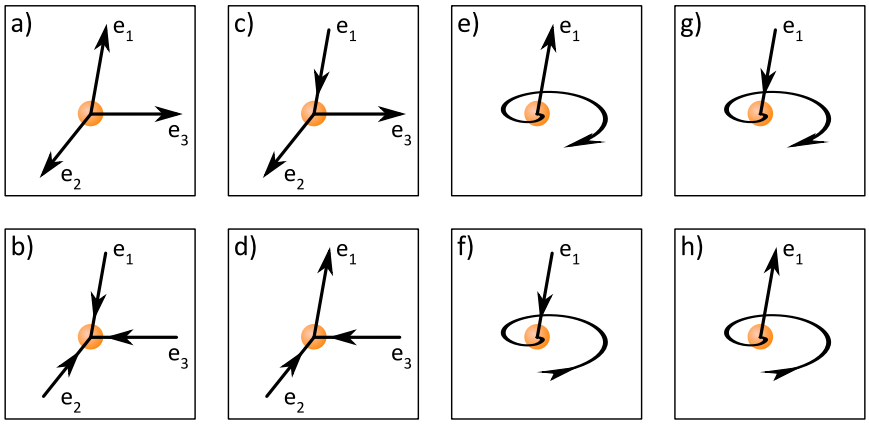


Figure 8: A comparison of first-order critical points in a three-dimensional vector field, e_i denotes a eigenvector (inspired by [14])

Figure 8 juxtapose the eight types of first-order critical points which arise if the four and two conditions for kinds of critical points in tables 2 and 3 are combined. In the figure, *a)* and *e)* denote a source (fully repelling), *b)* and *f)* a sink (fully attracting), *c)* and *g)* a repelling, and *d)* and *h)* an attracting saddle. The four on the left are known as nodes as their eigenvalues have only real parts. In contrast the four critical points on the right are referred to as foci. Here two eigenvalues have also imaginary parts, as shown in table 3. In general, the depicted arrows represent the eigenvectors of the Jacobian of the critical points. If the vector points to the critical point, its eigenvalue is greater than zero. If the vector points away from the critical point, its eigenvalue is less than zero. Therefore, the three eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_3$ in *a)* make the critical point a source, as they all point away from it. The two eigenvectors $\mathbf{e}_2, \mathbf{e}_3$ in *c)* and *d)* make up a separation surface. In *e)* to *h)* the two eigenvectors with eigenvalues that have imaginary parts are not depicted. Instead a spiral is shown which directly corresponds to the imaginary parts of the eigenvalues. The plane which both vectors are span is tangentially to the spiral in the critical point.

3.2.2 *Boundary Switch Curves*

Another important topological feature of vector fields are *Boundary Switch Curves* (BSC). If the domain of a 3D vector field is considered as a cuboid, there are six boundary planes. Let there be also no critical point lying on these boundary planes, every point on these can be declared to be part either of inflow, outflow or part of a boundary switch curve. This means that the vectors at these points are showing into the domain or pointing outside of it. Or, in case of the BSC, the vectors are tangent to the boundary plane. The BSC separate the inflow and the outflow at the boundaries like the flow switches from in to out by crossing the curve. An example is shown in Figure 9. The white BSC shown in the right image of Figure 6 are computed from vector field.

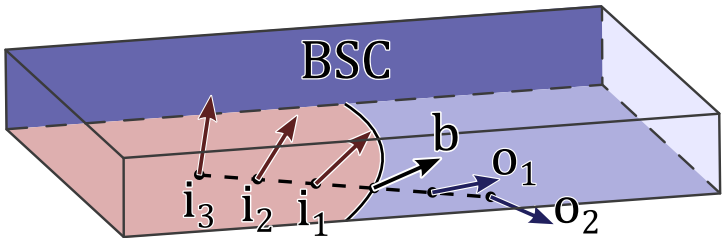


Figure 9: A boundary switch curve (black solid), every vector (e. g. $\mathbf{b}$) on this curve lies in the boundary plane (red/blue), red vectors ($\mathbf{i}_1, \ldots, \mathbf{i}_3$) point inside, blue ones ($\mathbf{o}_1, \mathbf{o}_2$) outside of the domain (inspired by [14])

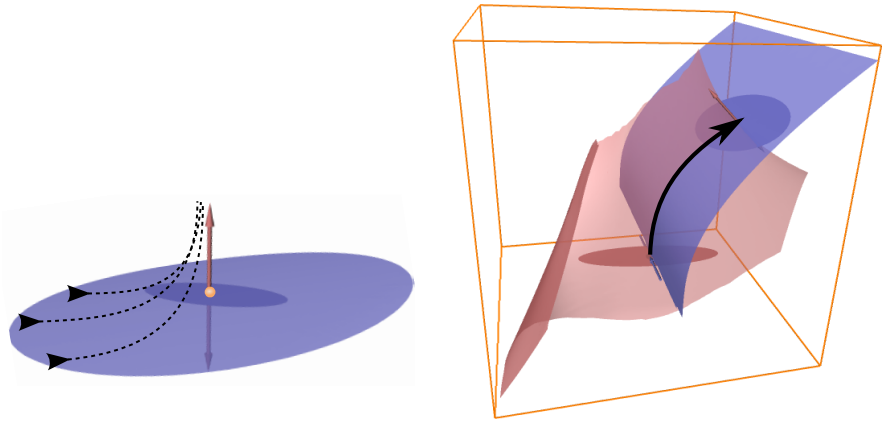
3.2.3 *Separatrices and Saddle Connectors*

This part explains separatrices and the recent concept of *Saddle Connectors* ([10]) which can simplify and support the visualisation of separatrices. Separatrices divide a vector field in regions of different flow. In two dimensions they are curves and in three dimension they appear as curves and surfaces. Critical points are the starting point from which separatrices can emerge. A vector field in two dimensions containing one or more saddles will have separation lines. They start or end in saddles in direction of the eigenvalues in forward and backward integration ([14]). The separation lines can also start or end at boundary switch points which are similar to BSC in three dimensions. If a separation curve is closed it forms a special type.

In three dimensions a vector field with saddles has separation surfaces and curves. Especially the former are difficult to visualise as they can occlude other features and themselves. An attracting saddle in three dimensions maintains three separatrices: two curves for repelling and one surface for attracting. The separatrix curves can end in other critical points or stop at the boundary of the domain. The Figure 10a shows a saddle and its separation surface (blue) and the two separation lines (reddish).

[10] presents a more intuitional way of visualise separatrix surfaces called *saddle connectors*. They are computed from the intersection of two separatrix surfaces if this forms a curve. Note that this intersection can degenerate to a new surface which will then not be considered as a saddle connector. They always form as a stream line of the vector field and connect an attracting and a repelling saddle. As a result a saddle connectors gives a good clue how the flow appears and where separatrices intersect. The common approach is to hide all separatrices and to show only the possible saddle connectors. The user can then activate the separatrices in which he or she is interested avoiding visual clutter by showing too much information at once.

Figure 10b shows a saddle connector between an attracting (top) and a repelling saddle. The black arrow also reveals the orientation of the stream line.



(a) Saddle (orange dot) with separation surface (blue) and curves (reddish, depicted as vector) (from [14]), three streamlines are shown

(b) Saddle Connector (black): intersection of separation surfaces of two saddles (taken from [14])

Figure 10: Separatrices and saddle connector

3.3 CLASSIFICATION OF THE RECONSTRUCTION

The following section will classify different approaches of a reconstruction. It contains the benefits and drawbacks which are used to choose the best methods.

3.3.1 *Global vs. Local*

Speaking of global and local means how the vector field topology was generated and how it should be interpreted.

In the local case one has given a grid which sub-divides the domain consisting of a vast amount of grid cells. They exist independently from each other, i. e. the vectors in one cell has no influence to the flow in another cell no matter how far or near it is. Reconstructing the vectors with a linear method, all this can also imply that the field contains abrupt changes when crossing the boundaries between two cells. Only the vectors on the boundary of two adjacent cells share information of these two.

On the other hand global means that the critical structures and the flow are bound together so that they influence each other over long distances. This is mostly given in analytical fields (c. f. Figure 5).

3.3.2 *Interpolation vs. Approximation*

As mentioned in section 3.4, interpolation is used to reconstruct values at data points with the help of known values at surrounding data points. This includes also that the result will contain the given values and not generate other. In the context of vector field reconstruction this means an interpolation will not generate new critical structure and will also retain existing ones. This advantage comes with the drawback of being computa-

tionally intensive. For example, if one tries to solve the interpolation with polynomials, the amount of critical points will be directly the degree of the polynomials that needs to be solved.

In contrast to the interpolation an approximation does not try to find the exact solution. This is most commonly seen while solving a system of linear equations where there are more equations than unknowns, i.e. the system is overdetermined. With the help of the method of the least squares an approximative solution can be found that fits best for every equation. In general, such a overdetermined system has no solution. For a reconstruction using only approximation this bears the problem that the result will not resemble to the original data. For the reconstruction of a vector field that means new critical structures can appear and old ones vanish.

3.3.3 *Conclusion*

For the course of this thesis the choice is to reconstruct a vector field by methods of interpolation. Depending on topological features the reconstruction must map them to the result, so all input critical points should occur in the result at the same position and no other critical points should emerge. As pointed out before, approximation must not be used.

The other aspect of the reconstruction is its scope. Chapter 2 gives an overview for different approaches both local and global. For the topological based reconstruction in this thesis local and global approaches should be developed allowing to choose adaptively based on the input.

3.4 INTERPOLATION

Interpolation in general is a method to find values at arbitrary data points by looking at the surrounding data points and their known values. As demonstrated in section 3.3 interpolation should be the preferred method for the vector reconstruction. The following subsections describe the interpolation methods used in this thesis.

3.4.1 *Inverse distance weighted interpolation*

The inverse distance weighted interpolation is a simple and most commonly used technique for interpolating scattered data. The main idea is that the values which are supposed to be interpolated should be less influenced by known values if these are farer away, i. e. their influence is inverse to the distance of each other. The interpolated values are an average of the scattered values weighted with the inverse of their distance, see Equation (3.3) for a principal formula. Every unknown value at a certain location can be interpolated with the N known, scattered values $f(x_i)$ and their locations

x_i ([9]). These values are weighted with the inverse of their distance to the unknown value.

$$f(x) = \sum_{i=1}^{N} \frac{f(x_i)}{\sqrt{x_i^2 + x^2}} \tag{3.3}$$

There exist some extensions, like a power of the distance. Only the simple previous equation will be used in this work.

As this method uses all values, it is called global. Taking only a certain subset of values, preferably those which are closest, makes it a local method. This would be useful if there is a huge amount of scattered data and the computation will take rather much time. This imposes the problem to choose the right subset of values. Moreover, using all values makes the resulting function continuous.

3.4.2 *Trilinear Interpolation*

In the one dimensional case trying to compute values $f(x)$ for $x_0 < x < x_1$ where $f(x_0)$ and $f(x_1)$ are given, one of the fastest way is to use a linear interpolation. The corresponding formulation is given by the formula:

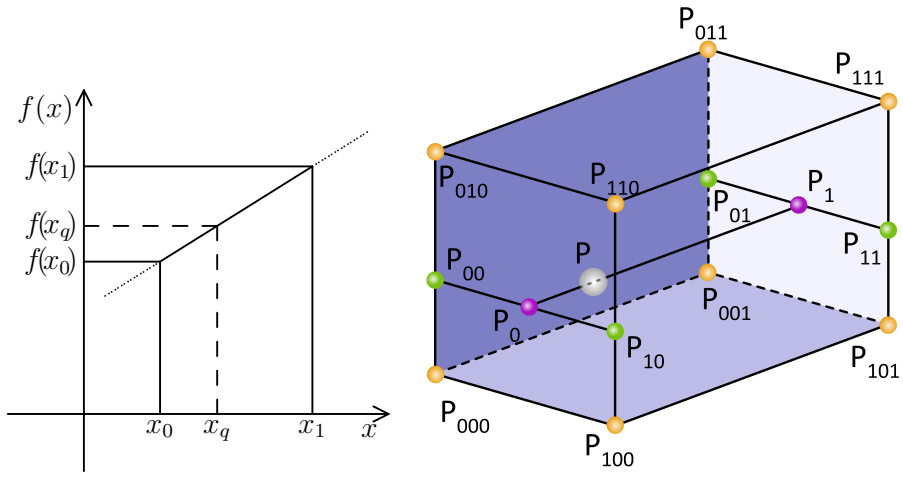
$$f(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0} \cdot (x - x_0). \tag{3.4}$$

As seen in Figure 11a the values between $f(x_0)$ and $f(x_1)$ form a straight line when computed by linear interpolation, in fact this computation is strongly related to linear functions. This interpolation can be extended to higher dimensions. Figure 11b shows a three times repeated linear interpolation resulting in a trilinear interpolation in a cuboid.

A simple formulation for a trilinear interpolation is given in Equation (3.5). It is an extension of a linear interpolation to three dimensions meaning that one can interpolate values in volumetric data sets. To use this formula the eight values $a_7, \ldots, a_0$ must reside at the eight corners of a cuboid.

$$\mathbf{v}_{xyz} = a_7xyz + a_6xy + a_5xz + a_4yz + a_3x + a_2y + a_1z + a_0 \tag{3.5}$$

Consider now a unit cube, i. e. a regular cuboid with side length 1 and let the corners be denoted by $\mathbf{p}_{000}, \mathbf{p}_{100}, \mathbf{p}_{010}, \ldots, \mathbf{p}_{111}$ which means that the first point is located at $(0, 0, 0)$ and so on. If there are values given at each of those eight corners, e. g. a vector, one can interpolate at any location inside (and outside if desired) the cube a vector with the following



(a) Linear interpolation, computing $f(x)$ at x_q by knowing x_0 and x_1 and their function values

(b) Trilinear interpolation in a cuboid, value P is a linear interpolation of P_0 and P_1 which are in turn a linear interpolation of P_{ij}, which are finally a linear interpolation of P_{ijk} values at the eight vertices of the cuboid

Figure 11: Linear and trilinear interpolation in a geometric sense.

Equation (3.6) where $\mathbf{v}_{000}, \ldots, \mathbf{v}_{111}$ denote the values at the corresponding corners.

$$\begin{aligned}\mathbf{v}_{xyz} = \mathbf{v}_{000} &\cdot (1-x) \cdot (1-y) \cdot (1-z) \\ &+ \mathbf{v}_{100} \cdot x \cdot (1-y) \cdot (1-z) \\ &+ \mathbf{v}_{010} \cdot (1-x) \cdot y \cdot (1-z) \\ &+ \mathbf{v}_{001} \cdot (1-x) \cdot (1-y) \cdot z \\ &+ \mathbf{v}_{110} \cdot x \cdot y \cdot (1-z) \\ &+ \mathbf{v}_{101} \cdot x \cdot (1-y) \cdot z \\ &+ \mathbf{v}_{011} \cdot (1-x) \cdot y \cdot z \\ &+ \mathbf{v}_{111} \cdot x \cdot y \cdot z\end{aligned} \tag{3.6}$$

If that cube is neither of unit size nor located at the origin, one can translate and scale it to conform that condition. The computation above is applied and the transformation can be inverted to get the result. The formulation in (3.6) can be changed to the following:

$$\begin{aligned}\mathbf{v}_{xyz} = &(-\mathbf{v}_{000} + \mathbf{v}_{001} + \mathbf{v}_{010} + \mathbf{v}_{100} - \mathbf{v}_{011} - \mathbf{v}_{101} - \mathbf{v}_{110} + \mathbf{v}_{111})xyz \\ &+ (\mathbf{v}_{000} - \mathbf{v}_{010} - \mathbf{v}_{100})xy + (-\mathbf{v}_{000} + \mathbf{v}_{001})z \\ &+ (\mathbf{v}_{000} - \mathbf{v}_{001} - \mathbf{v}_{100})xz + (-\mathbf{v}_{000} + \mathbf{v}_{010})y \\ &+ (\mathbf{v}_{000} - \mathbf{v}_{010} - \mathbf{v}_{001})yz + (-\mathbf{v}_{000} + \mathbf{v}_{100})x \\ &+ \mathbf{v}_{000}\end{aligned} \tag{3.7}$$

One important property of a trilinear interpolation is that it can generate only up to six zeros ([13]). Equation (3.8) shows another form of Equation

(3.5). One can see by a simple expand that the formulas are equivalent. Now consider that all zeros are searched. It is obvious that each one of the three trivariate functions u, v, w can take $-a_i$ for x, $-b_i$ for y and $-c_i$ for z. Combining these so that each equation is zero, there are at most six combinations, namely: $(-a_1, -b_2, -c_3)$, $(-a_1, -b_3, -c_2)$, $(-a_2, -b_1, -c_3)$, $(-a_2, -b_3, -c_1)$, $(-a_3, -b_1, -c_2)$, and $(-a_3, -b_2, -c_1)$ which are the six possible zeros. Expressing this in combinatoric terms, one tries to permute elements without repetition, yielding the formula $n!$ which gives for the three elements the value 6.

$$\mathbf{V}(x,y,z) = \begin{pmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{pmatrix} = \begin{pmatrix} (a_1+x)(b_1+y)(c_1+z) \\ (a_2+x)(b_2+y)(c_2+z) \\ (a_3+x)(b_3+y)(c_3+z) \end{pmatrix} \tag{3.8}$$

3.4.3 *Reverse Trilinear Interpolation*

This section deals with the inversion of the previous described trilinear interpolation. That means, vectors are searched at the unit cells' corners from given input vectors which may or may not be inside the unit cell. This unit cell is located at the coordination origin. Therefore eight vectors must be given to uniquely reconstruct the vectors at the corners.

$$\begin{aligned} \mathbf{v}_1 = {} & \mathbf{v}_{000}(1-x_1)(1-y_1)(1-z_1) + \mathbf{v}_{100}x_1(1-y_1)(1-z_1) + \\ & \mathbf{v}_{010}(1-x_1)y_1(1-z_1) + \mathbf{v}_{001}(1-x_1)(1-y_1)z_1 + \\ & \mathbf{v}_{110}x_1y_1(1-z_1) + \mathbf{v}_{101}x_1(1-y_1)z_1 + \\ & \mathbf{v}_{011}(1-x_1)y_1z_1 + \mathbf{v}_{111}x_1y_1z_1 \\ \vdots = {} & \vdots \\ \mathbf{v}_8 = {} & \mathbf{v}_{000}(1-x_8)(1-y_8)(1-z_8) + \ldots + \mathbf{v}_{111}x_8y_8z_8 \end{aligned} \tag{3.9}$$

This gives the system of linear equations shown in (3.9) which can be uniquely solved if the locations of the eight known vectors are not coplanar. $\mathbf{v}_1, \ldots, \mathbf{v}_8$ are just the given vectors and $\mathbf{v}_{000}, \ldots, \mathbf{v}_{111}$ are the vectors at the eight corners of the unit cube. The values x_i, y_i, z_i form the coordinates of vector $\mathbf{v}_i$. The vectors are forming a linear combination.

Every non-unit cuboid can be scaled and translated to be a unit cell at the coordination origin. After the reverse interpolation the cell and the result can be translated and scaled back to the original location and size.

3.4.4 *Reverse Interpolation from Critical Points*

Considering again an unit cell and an amount of given (first-order) critical points strictly inside it, the following will show when and how one can reconstruct the vectors at the eight corners, assuming a trilinear interpolation. The values $\mathbf{v}_{000}, \ldots, \mathbf{v}_{111}$ indicate the vectors at the eight corners of the cell:

$\mathbf{v}_{000}$ is located at $(0,0,0)$ and so on. For every critical point the location and its Jacobian are given, resulting in twelve information which can be used as in Equation (3.10) to form a system of linear equations. Note that the equations form for vectors, e. g. the left side of the first equation is the zero vector. Furthermore, the $\mathbf{J}_x, \ldots, \mathbf{J}_z$ are the three rows of the Jacobian.

The first equation just mentions that the eight vectors must interpolate a critical point at the given position (x, y, z). That equation can be differentiated in three dimensions, resulting in the three latter equations. The left sides are the entries of the Jacobian, the right sides are the derivatives.

$$\begin{aligned}
0 &= \mathbf{v}_{000}(1-x)(1-y)(1-z) + \ldots + \mathbf{v}_{011}xy(1-z) + \mathbf{v}_{111}xyz \\
\mathbf{J}_x &= -\mathbf{v}_{000}(1-y)(1-z) + \ldots - \mathbf{v}_{011}yz \qquad + \mathbf{v}_{111}yz \\
\mathbf{J}_y &= -\mathbf{v}_{000}(1-x)(1-z) + \ldots + \mathbf{v}_{011}(1-x)z + \mathbf{v}_{111}xz \\
\mathbf{J}_z &= -\mathbf{v}_{000}(1-x)(1-y) + \ldots + \mathbf{v}_{011}(1-x)y + \mathbf{v}_{111}xy
\end{aligned} \tag{3.10}$$

One critical point with its Jacobian results in four equations as seen before. On the other hand, the eight vectors are made up by 24 ($= 3 \times 8$, dimension times amount) unknown information. To get a system of linear equations which is neither underdeterminated nor overdeterminated, eight equations are needed. Exactly two critical points with their Jacobian will form a unique solution for this problem.

Again, if the cell is not the unit cell, it can be translated and scaled to be so.

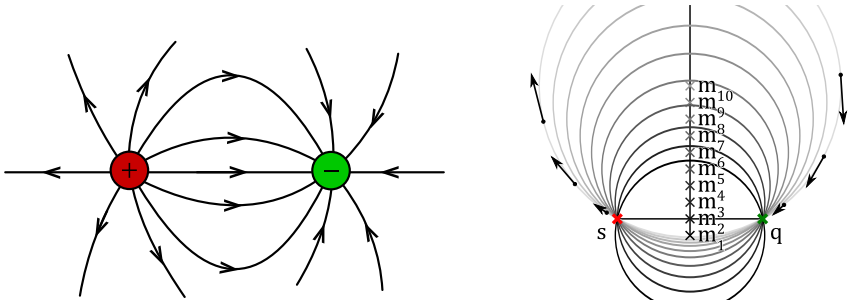
4

GLOBAL RECONSTRUCTION OF VECTOR FIELDS

This chapter describes the approaches to reconstruct vector fields in a global sense which I have developed in this thesis.

4.1 ONE SINK AND ONE SOURCE

To give a short introduction, this section shows how a reconstruction of a quite simple vector field in two and three dimensions can be done. This vector field consists only of one source and one sink. Such a vector field models for example the magnetic field of a bar magnet or two point charges of opposite polarity, as visualized in Figure 12a. Both point charges are visualised by two dots of different colour. The magnetic field lines are visualised by curves which start and end in the dots. They can be represented by stream lines. One important difference between point charges and the model is that they have an actual size and are not only points.



(a) Magnetic field lines created by two point charges of opposite polarity (taken from [7])

(b) Abstracting the point charges and their field lines

Figure 12: Illustrating the idea for reconstructing a vector field with one source and one sink

One can mathematically model this two dimensional magnetic field by a two dimensional vector field with the following approach. Using circles which go through both critical points, i. e. both poles, and any other point in the domain, that circle is uniquely defined. Moreover the centre of each circle has the same distance to both critical points. By convention the negative pole is the sink and the positive pole the source. Each vector of the field is parallel to the tangential vector of the circle that goes through

this point and both critical points. The direction of the vector is determined by the direction following the circle to the south pole. See Figure 12b for a visual explanation. It shows several constructed circles with their centre points $m_1, \ldots, m_{10}$. The red marked point s is the source and the green one the sink. Also for the circle with the centre point m_{10} six reconstructed vectors are shown.

The last missing piece is the length of the vectors which is computed in terms of the distance to both critical points. Moving along a stream line the length should increase until the distance to both critical points is equal and then decrease. One should define a maximum length l_{max} which can bound the computation in equation (4.1). The operator dist gives the Euclidean norm for its two arguments. For every point p in the length of its vector in the field can be computed.

$$l(p) = \arctan\Big(\min\Big(\operatorname{dist}(s;p); \operatorname{dist}(q;p)\Big)\Big) \cdot \frac{2}{\pi} \cdot l_{max} \tag{4.1}$$

Essentially all circles represent two stream lines. The construction guarantees that these stream lines will not intersect except in the critical points. The approach can also be used in three dimensional space. Again a vector is constructed with the help of a circle which goes through both critical points and the point where the vector should be computed. The vector lies in the plane at this location which is spanned by all three points. This leaves two possibilities for its direction. It must be that direction which points away from the source and points towards to the sink. The computation of the vector length is similar to equation (4.1).

This reconstruction is made in a global sense. At every point in the domain the two critical points are taken into account.

4.2 RECONSTRUCTION WITH CRITICAL POINTS

To extend the previous considerations for arbitrary many first-order critical points, I developed the following two approaches, in this and the next section.

Given are all n first-order critical points of the vector field with their respective Jacobians. These matrices carry information about the derivatives of the field at the locations of the critical points, namely the first-order partial derivatives (c. f. section 3.2.1). It is possible to sample, or better approximate vectors with the Jacobian, as equation (4.2) demonstrates. This pictures just a matrix multiplication. $\overrightarrow{C,P}$ is the vector between the critical point C and the location P. This should be taken with caution. The farther the sample location, the worse the result of the vector compared to the one in the original vector field. On the other hand, the smaller the ε gets, the shorter also the sampled vector get as they get nearer to the critical point where the vector field is null. This evaluation is similar to the one of the linear parts of the Taylor Series. One should use a fairly small distance to the critical point to get at least a good approximation for the directions of

the vectors surrounding the critical points. This first step provides a bridge between critical points and vectors, used although it is a approximation.

$$\mathbf{v}_{\text{sample}} = \mathbf{J}_{\mathbf{v}} \cdot (\overrightarrow{C, P}) \tag{4.2}$$

For the use of this equation one needs sample locations around the critical point. For symmetry considerations it is the best when these location have the same distance to the critical point and also are nearly equally distributed around it. This leads to considerations how to equally distribute k points on a sphere to which appendix A.1 gives some clues. Figure 13 shows a result of the distribution as it is used in this approach. The black dots represent $k = 1280$ nearly equally distributed locations on that sphere. The colored tori are the sections of the surface with the three planes $x = 0$, $y = 0$ and $z = 0$, respectively, considered the sphere's centre is located at the coordination origin.

An extension is to take more than one sphere surface. Generating m layers, needs m different ε values. They can be generated equally in decreasing order: $\varepsilon \cdot i/m$ for $i = m \ldots 1$. This improves the sample density around the critical point while still having the previous mentioned symmetry consideration.

Having all these k locations around the critical point, one can compute a vector between a location and the critical point. This serves as input for equation (4.2) giving a sampled vector around a critical point. All sampled vectors are assumed to represent in parts the flow around the critical point as they are sampled from the Jacobians.

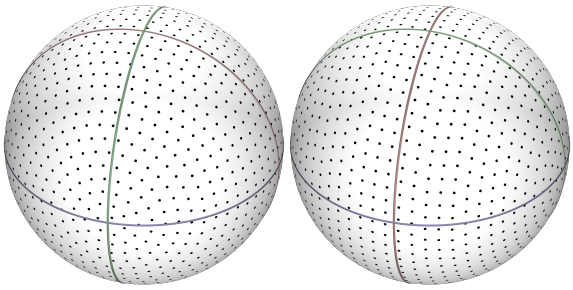


Figure 13: Two views of the same sphere with approximately equally distributed points on its surface, generated by the approach used in this work and rendered in the ray tracing program POV-Ray.

At this point there exist $n \cdot k$ so-called support vectors where k is the number of samples around each critical point. With those support vectors I now use a inverse distance weighted interpolation (c. f. 3.4.1) to compute other vectors that should make up the reconstructed vector field. This is done in global sense in that way that every vector has an influence on the result. The interpolation interweaves the local information generated near the critical points to one unit, making it possible to generate vectors at every position in the domain. Figure 14 illustrates again the two steps. Critical points as input are used to sample vectors in their surrounding.

The following step uses these as support vectors for a global interpolation generating vectors at arbitrary locations.

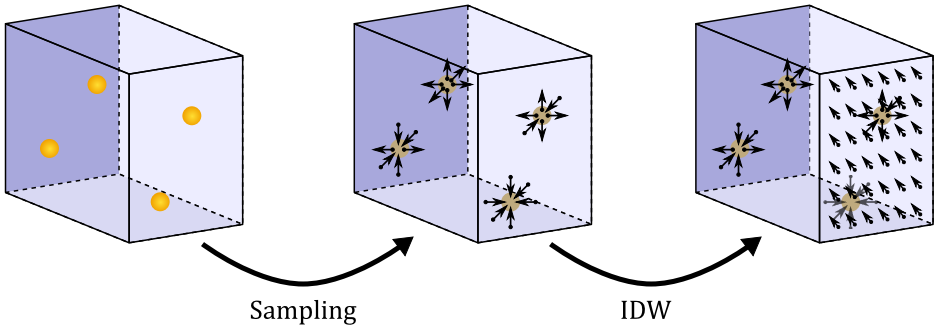


Figure 14: From four critical points to support vectors to a reconstructed vector field: the approach of the section 4.2

This first approach sketch can take the following parameters: the amount of points which are sampled around a critical point, the distance between the samples and the critical points.

While sampling very near to the critical points, one generates fairly short supporting vectors which also makes very short result vectors. Despite that, the supporting vectors are not normalised.

It is inherent to this approach that it at best only reconstruct the orientation of the vectors but not their length. Another remark needs the fidelity: one should not expect to get good results at locations where topological information (i. e. a critical point) is far away. Therefore, one can take the boundary switch curves as another topological structure into account.

4.3 RECONSTRUCTION WITH BOUNDARY SWITCH CURVES

The previous method lacked topological information to reach a better fidelity in reconstruction. The farther a reconstructed vector is away from topological data the poorer is it compared to the original. To reach a better result, I have added the *Boundary Switch Curves* (BSC, c. f. 3.2.2) as further topological feature to the method. This gives the information where vectors at the boundary are located that neither point in nor outside corresponding to the boundary.

After generating the support vectors from the critical points as described in the previous section, this extended method will try to reconstruct the vectors at the BSC as a next step. The reconstructed vectors of the BSC must lay in their corresponding boundary plane. In this example, each BSC is given as a polyline. At every control point of the polyline a vector is reconstructed. In general those vectors will not have the property being tangent to the boundary plane, so an iterative fitting is used: The reconstructed vectors at the BSC are transformed into the boundary to fulfil this feature and then in turn being used as support vectors together with the original vectors sampled around the critical points for the next iteration. This is done until the angle of the reconstructed vectors at the BSC and

the ideally BSC vectors is sufficiently small, i.e. until the vectors lay nearly perfect within the boundary. The iteration will not change the vectors that are sampled from the critical points. Again the inverse weighted distance interpolation is used. For the locations at the BSC a converging is used which will not return the constructed vector from the previous step but consume all support vectors of the current step to generate a new vector. However, if there is already a vector at this location it should have the biggest influence to the interpolation. Neglecting this convergence will prevent the vectors at the BSC from moving into the boundary plane.

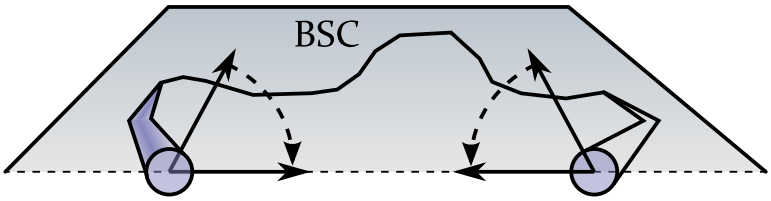


Figure 15: Moving vectors on a BSC into the boundary

The first obvious method to handle the vectors at the BSC is to rotate them into the plane. The rotation axis is the "tangent" of the BSC at that point and the rotation is done in that direction which takes the smaller rotation angle. If the BSC are taken as a polyline, it is not differentiable at their control points. The tangent at a control point of the polyline should be defined as the line that is coplanar to both line segments of the polyline at this point, goes through this point and has the same angle to both segments. This can be computed by central difference. See Figure 15 for an example, showing a BSC and two vectors which are rotated into the boundary plane.

Note that it is possible to generate vectors at the boundary switch curves that lay in the boundary plane but between two vectors can be a huge difference in their angles compared to their other neighbors. This is because they are rotated into the plane by the smallest angle. Giving that one vector is rotated by, for example 89°into the plane, its neighbor would be rotated by 91°in the same direction into the plane. But actually it is also rotated by 89°in the opposite direction just because the threshold is 90°.

Figure 16 outlines the three steps for the reconstruction mentioned in this section. Note the additional *Boundary* step in contrast to the previous section and Figure 14. The generated vectors at the BSC will serve as additional support vectors for the final step. Vectors from critical points and boundary switch curves are not distinguished. It could be further examined if this would necessary to rely more on the vectors of the critical points, i. e. weight these vectors more.

4.4 PARAMETERS

This sections outlines important parameters of the global reconstruction. The amount of samples which are generated for every critical point is the first one. Assuming it to large can slow down the whole reconstruction. A

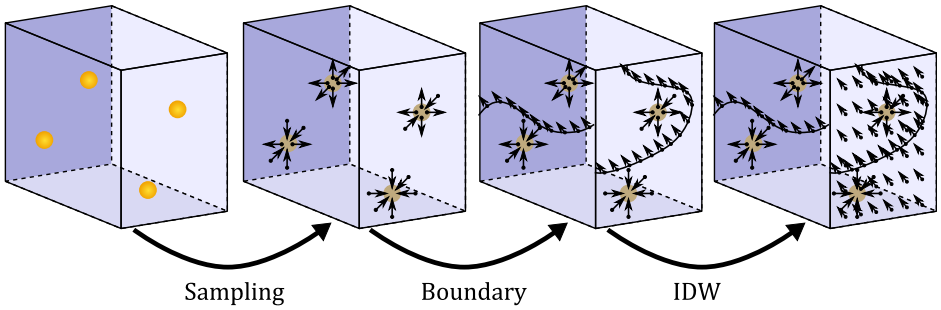


Figure 16: From four critical points to support vectors to a reconstructed vector field: the approach of the section 4.3

good intermediate value for not more than 50 critical points is 100. If there are more critical points, this value should be decreased.

The second parameter is the distance between the samples and the critical point, called *epsilon*. It correlates directly to the length of the reconstructed vectors. Making it too small can cause numerical problems. A value of 0.001 seemed to work well. Also, it should be considered that the value must not be larger than the half of the minimum distance between all critical points. Breaking this rule would generate samples for one critical point which are nearer to another critical point. This makes the vectors dependent from a Jacobian which is farther away than another. It would be a good guess to have this value at most at a fourth of this minimum distance.

4.5 CONCLUSION

This chapter showed two possibilities of reconstructing a vector field when given only its first-order critical points and its boundary switch curves. The reconstruction generated a global vector field. A local reconstruction is discussed in the next chapter 5. The results of this chapter are evaluated in chapter 7.

5

LOCAL RECONSTRUCTION OF VECTOR FIELDS

The previous chapter dealt with the reconstruction of vector fields in a global sense. In addition, the following chapter is dedicated to a local approach. This means that the space is divided into small subcells in which the reconstruction is done. The reconstruction in one cell is independent from the reconstruction in any other non-adjacent cell.

5.1 PRECONDITIONS AND INPUT

The reconstruction in a local sense gets as input data the critical points with their corresponding Jacobian matrices and the Cartesian or regular grid at whose vertices the vectors should be reconstructed. This is similar to the global reconstruction. As this approach works local the underlying grid needs to be known in beforehand. This is mainly done by specifying the width, height and length of one cell and the location of a corner of one cell. These six numbers define a Cartesian grid if each cell is a unit cube and the grid points are at integer locations, otherwise it is a regular grid. Considerations are not made for rectilinear and other grids. Also, the interpolation within each cell is assumed to be trilinear and depends therefore on the vectors at the eight vertices.

It is demanded that a critical point does not coincide with a vertex of any cell or lies on an edge of any cell.

5.2 ONE SINK AND ONE SOURCE IN ONE CELL

Reconstruction in one cell means that the vectors at the eight corners of the cell need to be found from the only input data, the critical points. Therefore one needs to know when the reconstruction in one cell has an unique solution. Clearly, if there is no critical point, the reconstruction cannot find any vector out of thin air.

As an introduction like in section 4.1, this section describes a simple example of reconstructing the eight vectors of a cell which contains one source and one sink as critical points. The interpolation within this cell is considered to be trilinear. The two Jacobians of both points are given. Figure 17 shows an example of such a cell.

To find the eight vectors at the cell's vertices, one can make up a system of linear equation where each critical point with its Jacobian matrix contributes

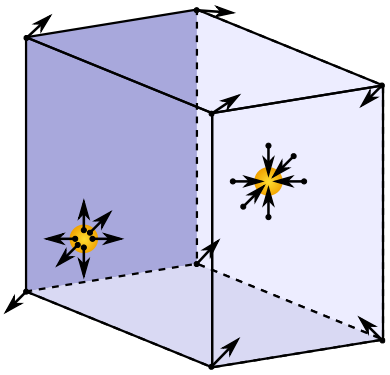


Figure 17: A cell with one source and one sink (orange dots, with prototypical vectors).

with four equations, see Equation (5.1) and section 3.4.4, where x, y, z denote the position of the critical point and $\mathbf{J}_x, \mathbf{J}_y, \mathbf{J}_z$ are the row vectors of the Jacobian matrix $\mathbf{J}$. The function $f(n, m)$ returns the value m if $n = 1$, otherwise $(1 - m)$ is returned. Likewise for f', if $n = 1$ the value 1 is returned, otherwise -1. This function f is introduced for brevity. See equation (A.1) for a matrix orientated form of this system of linear equations.

$$\begin{aligned}
\mathbf{0} &= \sum_{ijk \in \{0,1\}} \mathbf{v}_{ijk} \cdot f(i,x) \cdot f(j,y) \cdot f(k,z) \\
\mathbf{J}_x &= \sum_{ijk \in \{0,1\}} \mathbf{v}_{ijk} \cdot f'(i,x) \cdot f(j,y) \cdot f(k,z) \\
\mathbf{J}_y &= \sum_{ijk \in \{0,1\}} \mathbf{v}_{ijk} \cdot f(i,x) \cdot f'(j,y) \cdot f(k,z) \\
\mathbf{J}_z &= \sum_{ijk \in \{0,1\}} \mathbf{v}_{ijk} \cdot f(i,x) \cdot f(j,y) \cdot f'(k,z)
\end{aligned} \tag{5.1}$$

The first equation denotes the property of the critical point: it is the location of a vector with length zero of the vector field. And as it is a trilinear interpolation, the vector at the critical point is trilinear combined by the vectors at the eight corners. The last three equations are made up by the three directional derivatives at the critical point. These are also result of a weighted summation of the eight vectors at the corners but now in a differential sense.

With this tuple of four equations it is clear that two 4-tuples of them are needed to recover the eight vectors at the corners. As one critical point gives four equations as mentioned above, one cell needs two critical points to have a single solution and therefore an unique reconstruction. In general, if the matrix of coefficients of the linear equation system has a the same rank as the matrix of coefficients extended by the solution vector (i. e. extended matrix of coefficients), the system is solvable. If in turn the latter matrix' rank is equal to the number of unknowns (in this case 8), then there exists an unique solution.

With the help of formula 5.1 it is a straightforward exercise to calculate the eight vectors at the corners of the cell. These eight vectors can be used to interpolate every vector within this cell trilinear. Equation (3.6) performs this operation. As demanded the interpolation generates critical points at the given locations.

5.3 GENERAL APPROACH

From the previous example, I conclude to the following general approach. For the reconstruction, n critical points with their Jacobian are given and the underlying grid is known. At all vertices of the grid the vectors should be reconstructed to make up a trilinear interpolated vector field which matches the given critical points. Moreover no additional critical points should be generated. The local reconstruction implies that all cells share topological information only if they are directly connected, i. e. that they share at least one vertices. Given two cells that are not connected by that means, one cannot conclude from the one cell's topological information to the topological information of the other.

One simple and brute force approach could be to generate a system of linear equations. As shown in equation 5.1, each critical point makes up four equations which can recover four vectors at the cell's vertices. Let n be the number of critical points and k the amount of cells for the reconstruction. All k cells form up to a cuboid, i. e. $k = l \cdot w \cdot h$, and have together $m = (l+1) \cdot (w+1) \cdot (h+1)$ vertices. Then, one could make up a system of $4 \cdot n$ linear equations and m unknowns. This could generate a very huge system of equations which does not entirely fits into memory. For example, one used data set has 64,000 vertices and 482 critical points. The matrix of coefficients would be sparse only if there are not too much critical points but on the other hand, the less critical points, the less topology information can be relied on which gives in turn a less confidential result. In general, one does not get a system of equations which has a unique solution. Mostly only approximations will generate a solution, like least squares. Therefore, it was necessary to search for alternatives. The next sections will give an approach which follows a *divide and conquer* strategy.

I separate the local reconstruction in three general steps. The first is the reconstruction of cells which contain at least one critical point. This step is followed by the reconstruction of separation lines which start at *saddle* critical points. This will also generate vector information in cells where the lines pass. The last step is to reconstruct the vector information at the remaining cells with the help of a diffusion. The reconstructed information in one step is never overwritten by a following step.

5.4 RECONSTRUCTION OF CELLS WITH TOPOLOGY

The reconstruction of cells with topology is the first step in reconstructing vector fields locally. For this purpose, all cells which contain critical points are separated into four different types: isolated cells with one, two and

more than two critical points and clustered cells. The first three types label cells which are not proximate to other cells with critical points, i. e. they do not share any vertex with other such cells. This coins the term isolated. The fourth type is for those cells which are connected and have critical points, therefore clustered. This connection is established if two cells with critical point share one, two, or four vertices with each other. Each type will be considered separately.

ISOLATED CELLS WITH ONE CRITICAL POINT As the explanations in 5.3 cannot be applied to only one critical point, another approach is needed. No direct reconstruction is done, but instead, a sampling technique is used again. At the eight cell's vertices vectors are sampled with the help of equation (4.2). These eight vectors approximate the original best when the eigenvectors of the Jacobian are pairwise orthogonal and each parallel to one coordinate axis. With eleven such critical points an average angle difference between reconstructed vectors and original ones of less than 0.0037°could be generated. In another test the difference was about 18.221°. This value was the observation of testing 482 critical points from the Perlin data set (c. f. 7.3.1).

This approach can later be extended. One central question is, why the eigenvectors of a critical points must be pairwise orthogonal to get that nearly perfect result. In general, the eigenvectors are not pairwise orthogonal. For now, the approach is used as described but is left for further research.

ISOLATED CELLS WITH TWO CRITICAL POINTS If a cell contains two critical points, the approach from section 5.2 comes to turn.

ISOLATED CELLS WITH THREE OR MORE CRITICAL POINTS Cells with more than two critical points could be handled with a system of linear equations like in section 5.3. In general, this would yield a system which has no exact solution as there are more equations than unknowns. Only an approximation can be applied. Another possibility is to sample again vectors around each critical point and interpolate these support vectors to the cell's vertices.

For the course of this thesis, the latter possibility was implemented. But no data set hold this critical point configuration in order to test the result. This is left for further examination. Explanations in [13] show that a maximum of six critical points can be possible for a trilinear interpolation.

CLUSTERS OF CELLS WITH CRITICAL POINTS For clusters of cells where each cell contains at least one critical point, one can set up a system of linear equations. This system contains four equations per critical point (c. f. 5.2) and up to $m = n \cdot 8 - (n - 1)$ unknowns where n is the number of cells and greater than 1. This number is an upper bound where all cells are connected only by one vertex and two

cells are only connected to one other cell. The lower bound for m is $m = (l+1) \cdot (w+1) \cdot (h+1)$ where l, w, h are the amount of cells in length, width and height which make up a cuboid.

If the system of linear equations is too huge, it can be split into subsystems. For the computation of one vertex all adjacent cells with critical points need to be considered. One could convert the reconstruction from cell-orientated to vertex-orientated giving the possibility for parallel computing.

Again, for the Perlin data set the great majority of cells with critical points formed clusters. Implemented was the approach using a system of linear equations.

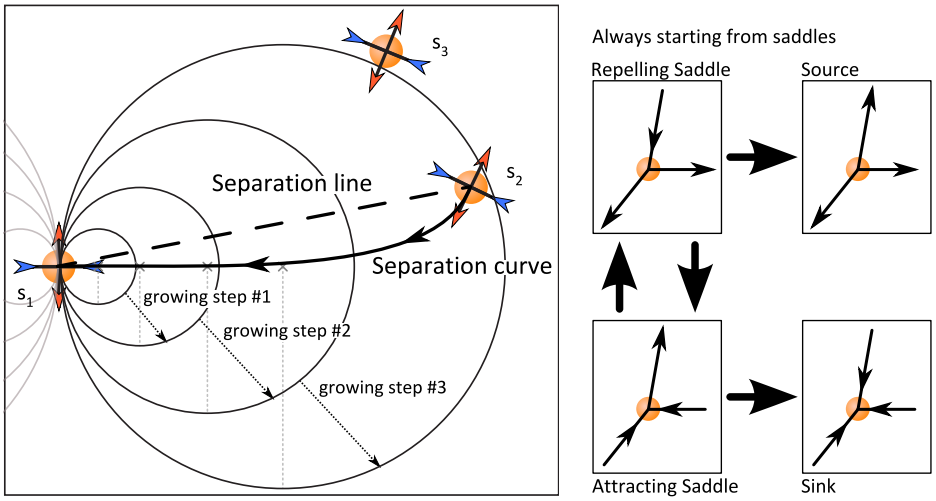
After this step, at all vertices of cells with critical points a vector is given which will be also part of the final result.

5.5 RECONSTRUCTION OF SEPARATION LINES

This section deals with a global strategy which supports the local reconstruction. As the critical points are given, one can try to find separation curves between them. These are stream lines which start and end in critical points in direction of a eigenvector, c. f. 3.2.3. The only starting points for this search are saddles which have in general one separation surface and two separation curves. This makes it possible to search for them, sinks and saddles can have arbitrarily many. One separation curve starts or ends in the direction of the eigenvector whose corresponding eigenvalue has a different sign than the other two, short *sole eigenvector*. The other starts or ends in the opposite direction (eigenvector multiplicated by scalar -1). Talking of starting and ending, this curve has a direction. The separation curves are a global feature as they spread potentially over several cells.

The search for separation lines is as follows. From every saddle of the topological skeleton the sole eigenvector is taken. A line is constructed that goes through the current saddle point in direction of that eigenvector. The following is done in both directions from the critical point. A sphere is made up whose centre is on that line and whose surface contains the critical point. One can see that there exist two possible spheres. The initial distance between centre of sphere and the saddle must be less than the minimal distance between all critical points to ensure that the sphere does not initially contain any critical points.

This sphere is iteratively growing from the minimal distance, i. e. the centre moves along the previous line away from the initial critical point, see Figure 18a. If the sphere contains one or more critical points, the one that is nearer becomes the ending point candidate for the separation curve. The found critical point must be analysed if it can be the other end of the separation curve in respect to the starting critical point. See Figure 18b for the conditions. If a connection is not possible, the next nearer critical points are considered or the iteration is continued. It is possible that the search will leave the convex hull of all critical points, i. e. no other critical points



(a) Search and construction of separation curve between saddles s_1 and s_2, red and blue arrows depict eigenvectors (repelling and attracting, resp.), circles represent the growing steps

(b) Possible connections between critical points (foci and nodes are not distinguished, arrows represent eigenvectors)

Figure 18: Constructing the separation line between critical points

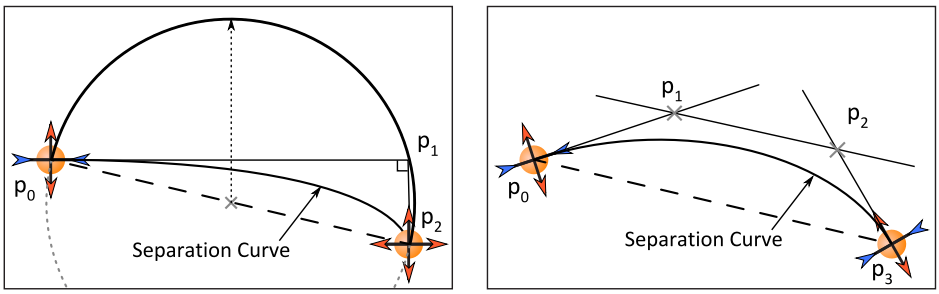
can be found. This is prevented by stopping the growing when diameter of the sphere reached a maximum, e. g. the diameter of the domain.

Having the connection between two critical points, one can reconstruct a separation curve. This is done by a quadratic or cubic Bézier curve whose first and last control points (p_0 and p_2 or p_3) are just the critical points, p_0 is always the starting one. If the ending critical point is not a saddle the quadratic curve is used and p_1 is constructed with the help Thales' theorem, see Figure 19a. p_1 is that point which makes with p_0 and p_2 a right triangle and the side $\overline{p_0p_1}$ is in direction of the sole eigenvector. If this line contains p_2, the curve degenerates to a line. The separation curve between two saddles is a cubic Bézier curve where p_1 lies on the line through the first critical point in direction of the sole eigenvector and p_2 lies on the line through the second critical point in direction of its sole eigenvector towards the first critical point. The two points will be placed that the three line segments $\overline{p_0p_1}$, $\overline{p_1p_2}$ and $\overline{p_2p_3}$ have the same length, see Figure 19b. Bézier curves were chosen because they are wide-spread in computer graphics and allow a simple definition and construction. The placement of their control points is done in a most simple way.

After having computed those separation curves for all critical points, there can exist cells which have no critical points but one or more such curves going through them. Every curve can be used to generate sampled vector, i. e. tangential vectors, within the cell. As the curves are Bézier curves, a differentiation at a curve point gives the tangent direction. The approach is to have at least eight sampled vectors which serve as n supporting vectors for a system of n linear equations where the eight vectors at the cells vertices are the unknown, c. f. 3.4.3.

The previous step can depend either on the computed curves or only on the direct connecting lines. However, both alternatives must generate

support vectors at locations which are not coplanar. Otherwise the system of linear equations will have no solution. For example, for the data set 5CP (c. f. 7.3.1, 19a) only straight lines as separation curves were generated. Comparing with the original vector field, the vectors are aligned to the lines. This case was handled or implemented.

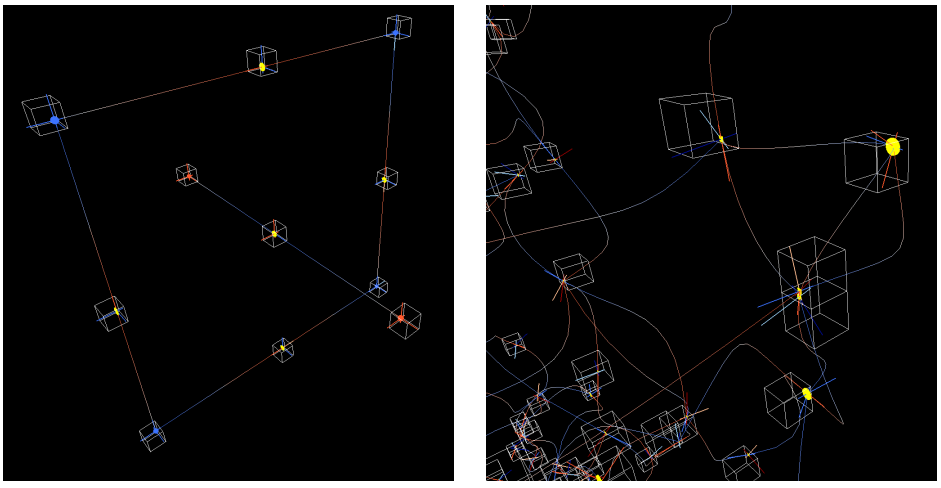


(a) Separation curve constructed between saddle and source

(b) Possible connections between two saddles (foci and nodes are not distinguished, arrows represent eigenvectors)

Figure 19: Constructing the separation curve between critical points (projected into two dimensions, arrows represent eigenvectors)

The approach does not distinguish between nodes and foci critical points, c. f. Figure 8. This would be a point for improvement as foci have a different flow behaviour than nodes. Figure 20 shows two final reconstructions of separation lines in two data sets which were used for evaluation in chapter 7.



(a) Data set 5CP

(b) Data set Perlin

Figure 20: Reconstructed separation lines, critical points with their cells. Orientation of the separation lines is from red to blue. Images made with JFlowVisFW.

5.6 DIFFUSION OF VECTOR INFORMATION

This last step is mostly inspired by the Heat Equation ([17]). At this point, there are still some cells which are not reconstructed, i. e. vectors at some vertices are missing. Mainly because no topological information was available in the previous steps. That said, expectations to the correctness of the following reconstruction should not be too high.

The diffusion in the Heat Equation describes the distribution of heat in a region with regard to the time. In the scenario of known vector information, I use a similar approach to diffuse these to locations where still no vector information is present. As the domain is in Cartesian coordinates, the vector Laplacian reduces to a scalar Laplacian applied to each component of the vectors, see equation 5.2 where $\mathbf{V}$ is the vector field and $\mathbf{V}_x, \ldots, \mathbf{V}_z$ .denote its three components.

$$\nabla^2\mathbf{V} = (\nabla^2\mathbf{V}_x, \nabla^2\mathbf{V}_y, \nabla^2\mathbf{V}_z) = (\frac{\partial^2\mathbf{V}_x}{\partial x^2}, \frac{\partial^2\mathbf{V}_y}{\partial y^2}, \frac{\partial^2\mathbf{V}_z}{\partial z^2}) \tag{5.2}$$

In this case, I need a discretization of the scalar Laplacian because the values are given and searched for at grid vertices. This is done with the help of finite differences, see equation 5.3 which is applied for every of the three components.

$$\begin{aligned}\Delta f(x,y,z) \approx & \frac{f(x-h,y,z)+f(x+h,y,z)-2f(x,y,z)}{h^2} \\ & + \frac{f(x,y-h,z)+f(x,y+h,z)-2f(x,y,z)}{h^2} \\ & + \frac{f(x,y,z-h)+f(x,y,z+h)-2f(x,y,z)}{h^2}\end{aligned} \tag{5.3}$$

For the implementation I have chosen the value h to be variable. At most, it should be the extent of one cell. But preferably, it is a half or a fourth of the cell extent. This will work in subvoxel space and allowing a better and more subtle approximation. If the value for h is chosen to be less than one, the values within cells are interpolated trilinear.

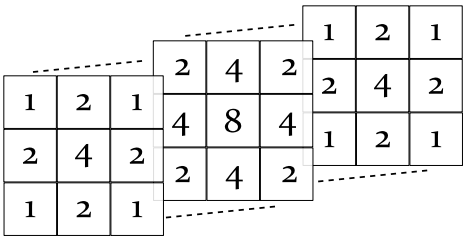
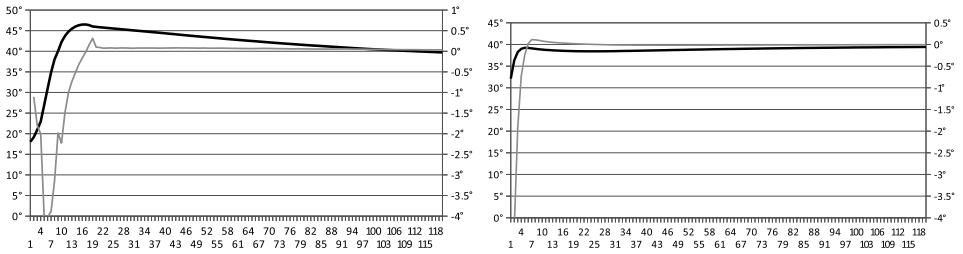


Figure 21: The three-dimensional Gaussian filter kernel (unweighted)

The previous equation (5.3) takes for a value at (x, y, z) only the six nearest neighbor values into account which make up a so-called *star*. It is also possible to use a bigger kernel which uses not only the six but the 26 nearest neighbors of a value. This gives a smoother result but is computational

more expensive. At locations where no vector information is present a value of zero is assumed. The diffusion is applied n times. In every iteration the already known values are restored at their location because they should not be manipulated by the diffusion. For the implementation I have chosen the filter kernel shown in Figure 21 which is a three-dimensional Gaussian kernel. The Gauss filter performs a diffusion, too, and is wide spread in image processing ([4]). Using the central differences gives only a rough approximation of the second derivatives. In three dimensions this can be achieved by a convolution with a Gaussian kernel. If this convolution is used at a subvoxel scope, the approximation should be sufficient.

The process will let the vectors converge to a state of equilibration which is considered when the vectors do not change for one iteration more than a given threshold. One problem is that the process may converge to a state which is worse than the states before. This cannot be detected without the help of the original vector field. The iteration must last at least until at every location is a vector information generated.



(a) 5CP data set (b) Perlin data set

Figure 22: Convergence of the diffusion, black shows the change of the difference of the original to the reconstructed vector field over the iterations (left axis), gray is the change of that difference between two iterations (right axis). The diffusion was not used at subvoxel scope.

The figures in 22 show two examples of the convergence. The left Figure is for the 5CP data set and the right for the perlin data set, see chapter 7 for information to both data sets. The two charts show how the difference of two vector fields changes over the iterations of the diffusion, measured in degree. For the 5CP data set this value raises fast in the first iterations. This is because the diffusion starts with only 0.1375% known data which is diffused, i. e. from 64,000 locations only at 88 of them the vector is known. This results in many uncertainties. After 16 iterations the difference decreases again to converge against a value of about 40°. The other data set starts with 4.9203% known data. Again the difference between original and reconstructed vector field increases to decreases shortly after the sixth iteration. But than the value for the difference seems again to slowly increase, converging against 40°. It was not reviewed if the approach always converges against that value and if the approach has other results for a subvoxel scope.

The result of this diffusion is to have vectors computed at every location of the input grid.

5.7 CONCLUSION

This chapter demonstrated a possibility of reconstructing a vector field when given only its first-order critical points. The reconstruction generated a local vector field which is in each cell trilinear interpolated. This can be seen as an extension to the approach presented in 2.2 where the space was subdivided into tetrahedra and the constructed vector field was linear interpolated in each tetrahedron.

Figure 23 concludes the three steps of this local reconstruction. The first step involves finding the cells with critical points of the grid and reconstructing the vectors at all vertices of these cells. The second step is to reconstruct the separation lines and to use the resulting curves to reconstruct the vectors at the passing cells if not already known. All information of the two first steps will be diffused in the last step, resulting in a complete reconstructed local vector field.

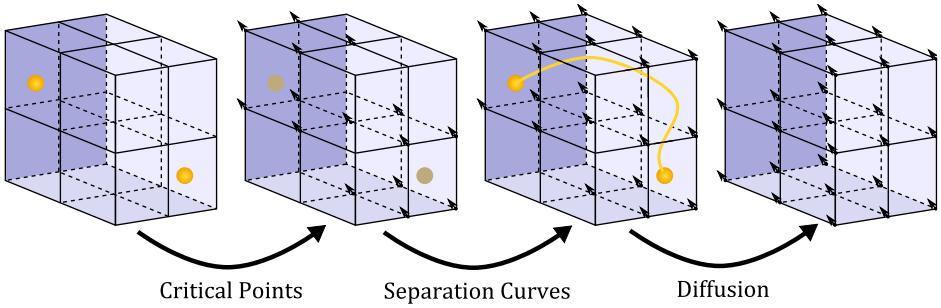


Figure 23: The three steps of local reconstruction presented in this chapter.

The results of this chapter are evaluated in chapter 7.

6

IMPLEMENTATION OF A PROTOTYPE

This chapter will give a review for the practical part of the thesis, the implementation of the previous shown, theoretical insights.

6.1 USED SOFTWARE

For the development, I have chosen Java as programming language. It gathers multiple advantages like being widely known and -used, mature and it gathers a huge amount of available frameworks and libraries. It is also important that it is well-known to the author. On the other side Java has the repute for not being very fast in comparison to more machine-orientated languages like C++. But as this should be a prototypical implementation, Java suffices and gives fast results. Due to the relationship between C++ and Java, it is possible to port the code to C++ afterwards.

Additionally the following libraries and packages for Java were used. All of these are open source software and can be downloaded freely with their sources.

OPENGL To visualise vector fields and corresponding mathematical structures I have chosen OpenGL. There exists a Java binding called JOGL which mimicries standard OpenGL code from C which in turn allows a straightforward translation of JOGL code to OpenGL in C(++).

APACHE COMMONS MATH This library is mostly used for analytical geometry and linear algebra purposes which disburdens to code existing algorithms again.

APACHE LOG4J Log4j is a quasi standard for handling logging in Java programs. It allows a finer configuration and a more sophisticated use than simple *System.out.println()* calls. The main task is to give a first level of debugging information.

GUAVA Although Java comes with a huge class library for handling collections and input/output, Google's Guava provides more sophisticated methods for handling data structures. Important uses concentrate on generating hash codes for objects in order to find fast a certain object in a huge collection.

JUNIT This is a library for unit testing Java program code. To ensure the program works correctly the most important classes have so called

unit tests. They can be executed to tell if the classes work the way they were intended. In fact, often the tests were written firstly followed by implementing the classes until the tests assured that they work correctly.

All the practical work resulted in a software library I named *JFlowVisFW* which offers the possibilities to read, write, visualise and reconstruct vector field and their corresponding features.

6.2 IMPLEMENTED FEATURES

This section will give an overview for the most important parts of the framework JFlowVisFW. It separates in the three parts Algorithms and Reconstruction, Visualisation and Input and Output. The first does the main work which is the reconstruction of vector fields. The second is used to show and explore the results of the reconstruction. Input and Output as third part acts as connection for importing and exporting outer data.

6.2.1 *Algorithms and Reconstruction*

The JFlowVisFW implements the interpolation algorithms mentioned in section 3.4 in a way the reconstruction can use it. The vector diffusion from section 5.6 is implemented to use multiple parameters like iterations, used diffusion kernels and sub voxel scope. The inverse distance weighting interpolation can be extended to support other approaches (like nearest neighbor).

All entities which this work deals in respect with vector fields, are modeled by object-oriented classes, like *CriticalPoints* and *PlainVectorField* as concrete, and *AbstractVectorField* as abstract base classes. The steps of the global and local reconstruction handle with these objects and are logically split in separate classes offering their functions. For example, the reconstruction of the separation lines from section 5.5 is divided into finding the pairs of critical points which are thought to be connected and modeling the separation curves.

Both reconstructions are implemented as systems of combining different algorithms which allows to omit some steps of the reconstruction. On the other hand, separate steps can be individually tested visualised to study their performance.

6.2.2 *Visualisation*

There are two important topological features of a vector field which needs to be visualised: Critical Points and Boundary Switch Curves. Furthermore it is important to show vector fields, too. The following list gives an insight in the visualisation techniques. Mathematical details can be found in 3.2.1 (critical points) and 3.2.2 (boundary switch curves).

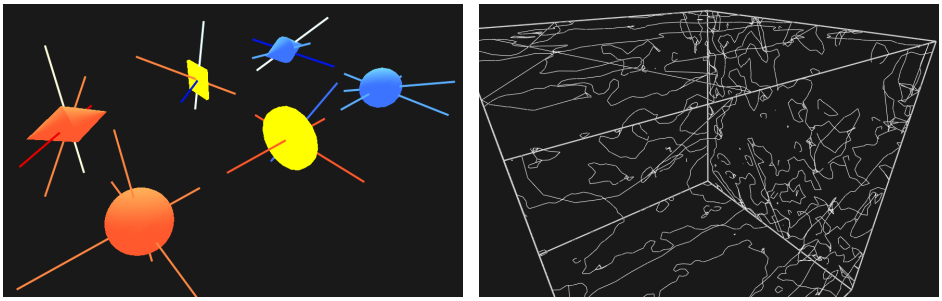
CRITICAL POINTS A visualisation sample of six different critical points is shown in Figure 24a. I was inspired for the visualisation of critical points by [14]: a critical point is depicted by an icon, a sphere for a node, a double cone for a focus (sinks and sources). The colour for a sink is blue and for a source red. For node saddles a flat disc and for focus saddles a flat cuboid is used. Both types of saddles' icons are yellow. An addition to the model in [14] is the drawing of the eigenvector lines. Each critical point has three eigenvectors whose eigenvalues determine if they are kind of attracting ($\mathfrak{R}(\lambda) > 0$) or repelling ($\mathfrak{R}(\lambda) < 0$). The former kinds are drawn blue and the latter red, likewise that a sink as fully attracting critical point is drawn blue. With the drawing of the eigenvectors one can distinguish also the saddles into (mostly) attracting and (mostly) repelling just by counting if there are more red or more blue drawn eigenvectors. The information of focus critical points if the imaginary part of a eigenvalue is less (or greater) than zero is encoded by a lighter (or darker) colour.

BOUNDARY SWITCH CURVES The boundary switch curves of a vector field are stored as a point list and this makes the drawing of them to a straight forward exercise in OpenGL: drawing stripes of lines. Figure 24b shows an example of a dataset. The thinner lines are the boundary switch curves. The thicker ones are just the edges of the domain and shown as orientation.

VECTOR FIELDS Under the assumption that a vector field is given in a grid based structure, i.e. vectors are mapped to all cell vertices, a visualisation of all these vectors at once will produce a visual clutter. I decided to use a sliced version of that idea showing only vectors at points which lay in the same plane. These slices can be moved to get an impression of the vector field. Figure 25a gives an example. There are shown three slices of a vector field at once and also the edges of the boundary domain for orientation. The lines only show direction not length of the vectors, all vectors have the same length. For this purpose the longer vectors are drawn in red while the shorter one are in blue. In between they become white.

These are the basic visualisation techniques. On top of them I made some exploring tools. As showing all information of a three dimensional vector field at once produces a visual clutter, I developed the idea of the slices, mentioned in the above list. This is inspired by tomography and also allows interactive exploration of the data.

LENGTH OF VECTORS Figure 25b gives an example of a slice showing the length of the vectors of a field. One can also see the critical points of the corresponding data set. Now, the clue is that a red-yellow-green colour scale is used. The longest vectors are represented by a green data point, the shorter a vector the colour changes over yellow to red where pure red is zero length vectors. There are nine critical points



(a) Critical points (with eigenvectors) (b) Boundary switch curves

Figure 24: Two topological structures of a vector field rendered in JFlowVisFW

in the figure that are just one unit away from that slice. One can see that the colour goes to red where there is a critical point.

DIFFERENCE OF TWO VECTOR FIELDS Figure 25c shows an example of visualising the difference between two vector fields. Difference means here, take the two vectors of each field at a point and compute the angle between them. The values of this difference range from 0°to 180°It does not take into account the length of the vectors. Again this difference values are mapped against a colour scale. Vectors which have the same direction produce a green data point. The colour changes over yellow, orange, red, violet to white for the greatest difference. This scale is also exchangeable.

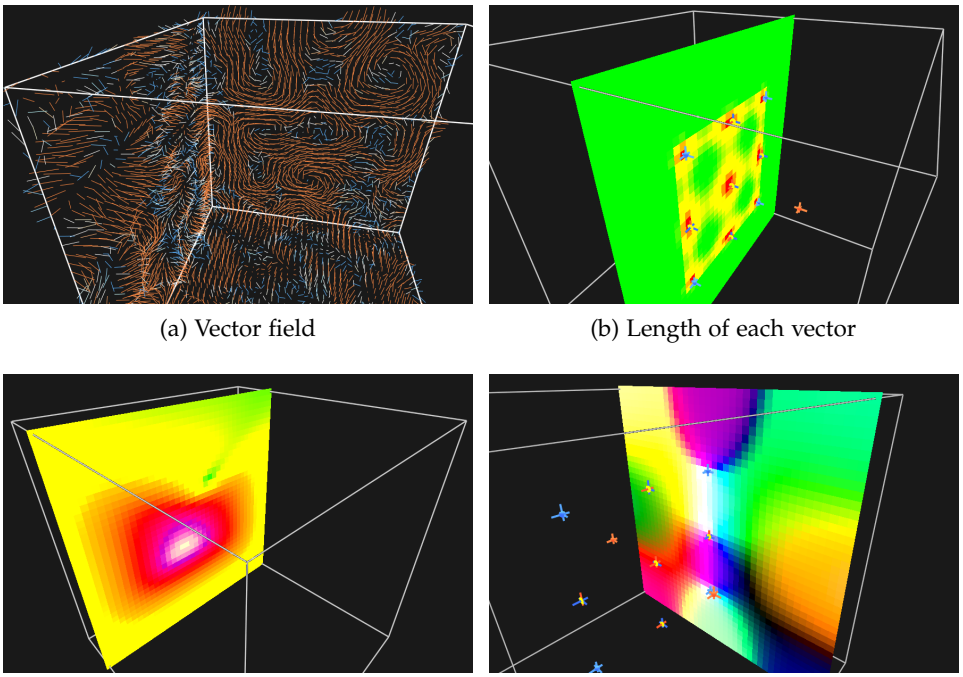
ORIENTATION OF THE VECTORS Figure 25d depicts a sample of a slice which draws the orientation of the vectors. Orientation means here that the x, y, z components of the vector are translated into r, g, b components of the RGB color space.

The so-called slice drawers can be set to different colour scales giving the control on which difference value for example matches to which colour. There exists already some scales, see Figure 26 for common used scale for the difference between two vector fields. 0°is the smallest difference, therefore having green assigned. While getting larger in the difference, the scale gets over yellow to red and violet.

For example, the user can use the difference visualising slices to explore where the reconstruction generates the greatest failure. This can be combined with the presence of the critical points giving possibly the insight that the reconstruction is best in the surrounding of the critical structures.

6.2.3 *Input and Output*

The JFlowVisFW software handles different input and output file formats. It can read plain ASCII file formats for critical points, boundary switch curves and vector field. In addition, vector fields and boundary switch



(a) Vector field

(b) Length of each vector

(c) Difference of two vector fields

(d) Orientation of the vectors

Figure 25: Visualisations of vector fields and corresponding features like length and orientation of vectors or differences between vector fields. All pictures showing the edges of the domain.

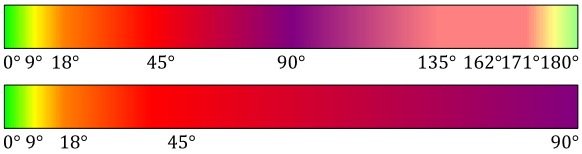


Figure 26: Example for two scales which are used in JFlowVisFW

curves in the Amira[1] ASCII format can be read, too. The following list gives a short overview.

PLAIN VECTOR FIELD Each line contains the three coordinates of the position and the three components of the vector, separated by white space

PLAIN BOUNDARY SWITCH CURVES Each line contains the coordinates of a BSC as line loop, different BSC are separated by empty lines

PLAIN SKELETON Enumeration of critical points with their location and corresponding Jacobian matrices

AMIRA VECTOR FIELD Contains an implicit description of the grid followed by the one vector per line in a certain order

AMIRA BOUNDARY SWITCH CURVES Contains different data section. An indexing of points is followed by linking the indices to a BSC

The input is realised for all formats, the output only for the vector field formats. Due to its architecture, it is easy to extend JFlowVisFW the read and write of other formats if needed.

6.3 PROTOTYPE

This section will describe the result of this diploma thesis: a prototype of an application for reconstruction of vector fields. The graphical user interface (GUI) of that prototype is shown in Figure 27. It offers controls for input, output and visualisation. In the upper panel, named *Input*, the user can choose which topological structures should be read. This includes at least the critical points and can be further extended by the boundary switch curves. In addition, it is possible to choose the original vector field which can be later visualised with the reconstructed one. The supported input files can be in plain and Amira formats. The panel in the middle, named *Output*, offers options whether writing the result of the reconstruction to which file. Here the output format must be explicitly chosen. An important option is the scope: the user must set if the reconstruction should be in a global or in a local sense. The resolution of the grid can be edited. If the original vector field is given, the grid will be deduced from it. The grid is defined by the three values *min*, *max* and *spacing* in three dimensions, resulting in a regular grid.

The interactive part of the prototype is the visualising of the result. The two lower panel, named *Visualisation* and *Tune Visualisation*, give the options to show various visualisations mentioned in section 6.2.2. Also, if the original vector field was given, the user can explore the differences between it and the reconstructed vector field by enabling the *Show Difference* option. The slider named *Slice* allows to move the active visualisation, see 6.2.2 for details to the sliced visualisation.

1 A visualisation and data-analysis platform, see `http://amira.zib.de/`

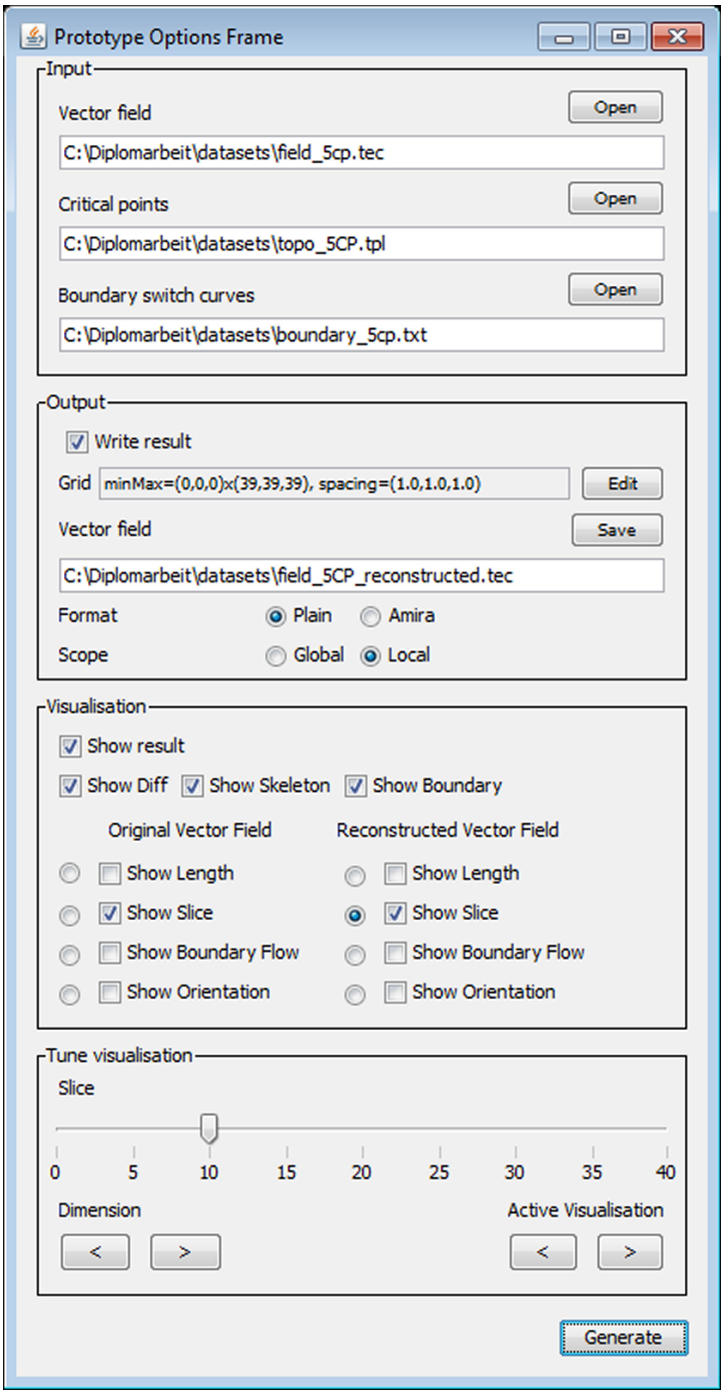


Figure 27: Graphical User Interface of the prototype for reconstructing vector fields

If every input parameter is tuned and the user hits the *Generate* button, the reconstruction will start. If the visualisation was enabled, in a second window the results will be shown.

6.4 RESULTS

This chapter gave insights in the programming tasks and their results during this diploma thesis. Vector fields cannot only be reconstructed, the results can be visualised in context to the original input structures. This gave direct feedback of the performance of the reconstruction.

7

EVALUATION OF THE RESULTS

After shown the theoretical and practical results of the thesis, these have to be evaluated. This will include experiments which measures the quality of the approaches. This chapter is divided for the local and the global approaches prefaced by an introduction section. During the evaluation, a computer with 8 gigabyte of RAM and 4 processors with 2.53GHz was used.

7.1 INTRODUCTION AND EXPECTATIONS

What this chapter should evaluate, is how good the approaches in chapters 4 and 5 reconstruct the original vector fields. This performance should be measurable by numbers. The output of both approaches are vectors. I decided to measure the angle between given vectors of the original vector field and the ones of the output. In order to do so, the approaches must reconstruct at all location where the original vector fields have vectors. The comparison is done with the help of histograms showing the amount of angles in respect to intervals and to the total number of vectors. This is done with nine intervals from 0° to 90° and 18 intervals from 0° to 180°.

The angle between the vectors of the original and the reconstructed vector field can be considered as a local performance number at every location. To get one global indicator all angles can be summed up and divided by their amount, giving simply the average of the angles. Figure 28 shows an example of an original vector $\mathbf{o}$ and two reconstructed vectors $\mathbf{r}_1$ and $\mathbf{r}_2$. Two cases are considered: angles between 0° and 180° and on the other hand 0° and 90°. The latter case is similar to the first but all angles greater than 90° are subtracted from 180°. Note that angles between three-dimensional vectors are within 0° and 180°, inclusively. This gives a measurement which ranks vectors worse the more they are tending to 90°. This case means in other words that angles are not considered by their orientation, i. e. parallel vectors with different orientation are then considered to be *equal*. In Figure 28, vector $\mathbf{r}_{3a}$ has an angle of more than 90° to vector $\mathbf{o}$. It is then mirrored at the imaginary dashed line which is perpendicular to the original vector $\mathbf{o}$. This is the same as the "subtraction of 180°". The three colours in the figure are inspired by the visualisation in section 6.2.2.

Not considered is the relation of the lengths. The two approaches do not try to reconstruct the length of the vectors but the attention lies only on

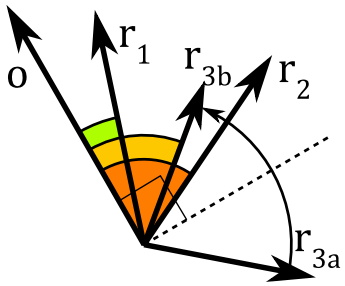


Figure 28: Principal difference between two vectors $\mathbf{o}$ and $\mathbf{r}_i$. If a reconstructed vector has an angle greater than 90°, like $\mathbf{r}_{3a}$ it is subtracted from 90°, to result in vector $\mathbf{r}_{3b}$

their direction. Especially the result in the global approach can contain very short vectors, in dimensions of 10^{-8} and less, depending on the parameters.

All charts in this chapter show the result of the reconstruction as the difference between the reconstructed and the original vector field. They map the relative frequency of a vector difference to a interval of angle which is shown by bars. Additionally an accumulated relative frequency for differences less or equal to the current interval is depicted with a line.

7.2 GLOBAL RECONSTRUCTION

The global reconstruction of a vector field is described in chapter 4. This section will evaluate the performance of the approach when using only critical points and when additionally the boundary switch curves are taken into account. The amount of samples per critical point is 1000 and the distance ε to a critical point of the samples is 10^{-1}. The distribution strategy for the samples is to use 10 layers of samples, as described in the chapter.

7.2.1 *Test data*

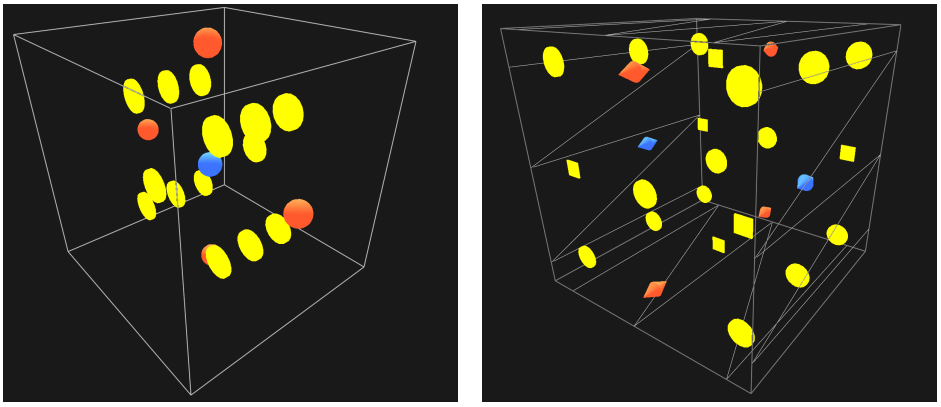
The test data consists of two sets of vector fields. They are both depicted in Figure 29. The equation in (7.1) gives an analytic description of the vector field *VRTD*. For the other data set *GCS* no analytic description is given. The names are short forms of *Global Case Shepard* and *VR Test Data*, respectively.

$$v(x,y,z) = \begin{pmatrix} \sin(0.15)\cdot\sin x - \cos(0.15)\cdot\sin(y) \\ -\cos(0.15)\cdot\sin x - \sin(0.15)\cdot\sin(y) \\ -\sin z \end{pmatrix} \tag{7.1}$$

The *GCS* data set contains four repelling and one attracting nodes (sources and sink, blue and red spheres, see Figure 29a) and 14 node saddles (yellow discs). No data for the boundary switch curves is given. The original vector field is rastered in a $32\cdot32\cdot32 = 32768$ values grid, going from $(-3,-3,-3)$ to $(-2.8125,-2.8125,-2.8125)$.

The *VRTD* set contains four repelling and two attracting foci (blue and red double cones), ten repelling and five attracting node saddles, and two

repelling and four attracting focus saddles, resulting in 29 critical points. Additionally the boundary switch curves were given. The original vector field is sampled in a $64 \cdot 64 \cdot 64 = 262144$ values, regular grid, going from $(-4, -4, -4)$ to $(3.875, 3.875, 3.875)$. In relation to the *GCS* data set, this one has eight times more sampled locations. The minimal distance of the all critical points is for the *GCS* data set about 0.9788 and for the *VRTD* data set 3.1416 (ca. π).



(a) The GCS data set: 19 critical points

(b) The VRTD data set: 27 critical points and additionally boundary switch curves

Figure 29: Two test data set for the global reconstruction illustrated by JFlowVisFW with domain bounding box in white

Because of the absence of the boundary switch curves in the *GCS* data set, the global reconstruction described in section 4.2 will be used for evaluation. The extended approach with boundary switch curves (c. f. 4.3) will be used to evaluate data set *VRTD*.

7.2.2 *Results for the GCS data set*

The approach reconstructs a great amount of vectors that are perpendicular to the original vectors, see Figures 31a and 31b. Almost 37.3% of the vectors are between 80° and 100°. Albeit, the third most frequent interval is already $[0°, 10°)$ with 7.3%. This distribution of vectors seems to be no coincidence, however knowing that a vector is almost perpendicular to the original does not imply in which direction to rotate it to become a better reconstruction for the original. This is due to the three-dimensional space. At least for this data set, the approach reveals some structure of the original vector field by its reconstruction.

7.2.3 *Results for the VRTD data set*

The *VRTD* data set was reconstructed twice: without and with involving the given boundary switch curves. Figures 32a and 32b show the result for the first case, 33a and 33b for the second case. The main observation is that the approach with the boundary switch curve is not significantly

better than the one without. In fact, without the reconstruction produces an average of angle difference of circa 47° while the reconstruction with boundary switch curves generates an average of 50.5°.

A second observation is that the reconstruction at all seems to be only a guessing of vectors. Taking the range between 0°and 90°, all intervals have an average frequency of about 12%, except the first which has significantly less. This means more than 90% of the vectors are far from reconstructed correct.

In contrast to the previous data set, it should be analysed why the approach fails here so much.

7.2.4 *Parameters and Runtime*

The following was tested with the *GCS* data set. Results of the reconstruction are within 0°and 90°.

The two parameters have different influence on the result. Firstly, the sample amount is set to ten different values. Figure 34a shows the result and, in direct relation, the amount of time it took to compute the result for the reconstruction. The result of the reconstruction degrades from amount 25 to 100 but than slightly decrease to converge to a value of circa 59.5°. The only effect of increasing the amount of samples beyond this point is to elongate the time of the computation. Reasons for the stagnation of the result at the value of 100 samples and more, may be that even more samples do not contribute more topological information for the reconstruction. It is remarkable that the reconstruction is slightly better for 25 than for 100 samples. This fact needs some more attention. Probably, it is not enough to generate equally distributed samples around the critical point. It should be checked if less samples result in a better reconstruction if they are arranged in regard to the Jacobian and their eigenvectors. All test runs used a value of circa 0.49 for epsilon which is the half of the minimal distance of all critical points.

Secondly, the distance between the samples and the critical point is set ten different values. Figure 34b shows the relation of these values to the result of the reconstruction. Values lower than 0.01 do not change the result significantly. If the value gets too small, the result should become unpredictable, as numerical instabilities occur. All test runs used a value of 100 for the sample amount per critical points.

At least the first parameter is left for further optimisations.

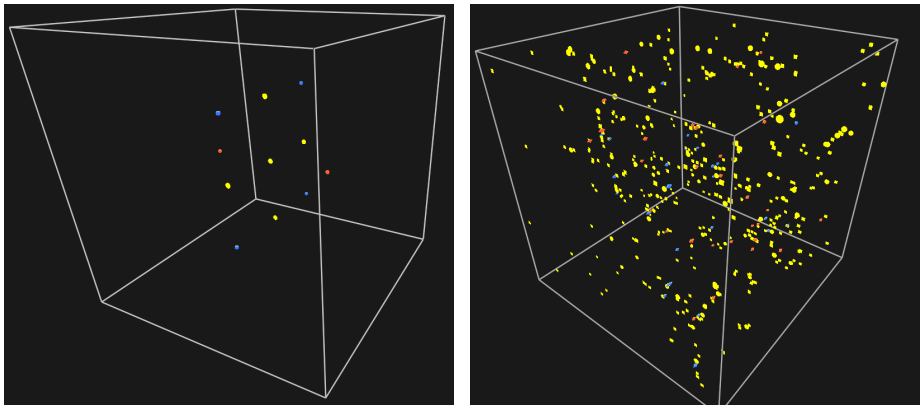
7.3 LOCAL RECONSTRUCTION

The local reconstruction of a vector field is described in chapter 5. This section will evaluate the performance of the approach as a whole and parts of it. These are reconstructing cells from critical points and from separation lines, and the diffusion of the vector information as the final step which makes three configurations: consider only cells with critical points, consider these cells again but additionally diffuse the result in the whole domain,

and finally the intermediate step of reconstructing vector information from separation lines. In general, only the diffusion generates vectors at all vertices of the grid. For all reconstructions, an iteration amount of 15 steps and a subvoxel scope of 0.25 was chosen for the diffusion.

7.3.1 *Test data*

There are two different data sets used in the evaluation: an highly symmetric (called *5CP*) and a more real life data set (called *Perlin*), Figure 30 gives an overview.



(a) The 5CP data set (b) The Perlin data set

Figure 30: Two test data sets for the local reconstruction illustrated by JFlowVisFW with domain bounding box in white

The *5CP* data set contains four sinks and two source, completed with five saddles, making up eleven critical points. All of which are nodes. One important remark: all eleven Jacobians are diagonal matrices and having pair wise perpendicular eigenvectors aligned to the coordinate axes. On the other hand, the *Perlin* data set contains 344 focus saddles, 83 node saddles, 26 focus sinks and 29 focus sources, summing up to 482 critical points. Both vector fields are rastered in a Cartesian grid between the points $p_{min} = (0, 0, 0)$ and $p_{max} = (39, 39, 39)$. The minimal distance between all critical points is in the *5CP* data set 9.7347 and in the *Perlin* data set 0.0113.

7.3.2 *Results for the 5CP data set*

Figure 35 shows the results for reconstructing the *5CP* data set locally. The grey bar at the interval of [0°,10°] stands for the reconstruction of cells with critical points. As this data set has eleven critical points, each in its own cell, the bar represents 88 almost correct reconstructed vectors. None of them has an angle of more than 0.5° to the original vector. This is due to the method described in section 5.4. In relation to all 64,000 vectors of the grid, this makes less than 0.2% of the reconstruction.

Applying the diffusion generates vector information at the remaining locations. This result is shown by the black bars. In both charts the three highest bars are for the three lowest intervals. This makes more than 50% of all vectors could be reconstructed with a difference of 30° or less. The average angle for all reconstructed vectors is for intervals up to 90° 32.9°.

Due to an implementation error, during the experiment only 6490, i. e. ca. 10% of all vectors were reconstructed. Nevertheless, this result can be taken representatively but further analysis should be made.

As seen in Figure 20a, reconstructing separation lines for this data set gives only straight lines. For now, the approach cannot use these single straight lines to reconstruct vectors of a cell. However, it would be possible if there were three straight lines within one cell which are not coplanar.

7.3.3 *Results for the Perlin data set*

Figure 36 shows the results for reconstructing the *Perlin* data set locally. Again, the white bars and lines represent the reconstructed vectors at cells with critical points. The grey bars and lines stand for the additional diffusion of the vector information. Finally, the black bars and lines represent the whole local reconstruction with taking the reconstruction of separation lines into account.

One can see that the highest white bar is for the best angle interval [0°,10°] and already 46% of the vectors are reconstructed with a difference to the original from not more than 30°. This step reconstructed 3117 or 4.9% of 64,000 vectors. The diffusion of the vector information moves the peak of the most reconstructed difference to the interval [30°,40°].

Figure 20b shows that many separation lines could be reconstructed for this data set. Despite this fact, using these for the reconstruction makes the result worse than without. The black bars and lines in Figure 36 depict that the majority of the vectors are reconstructed with a big difference to the original ones. Almost 50% have a difference of 60°or more.

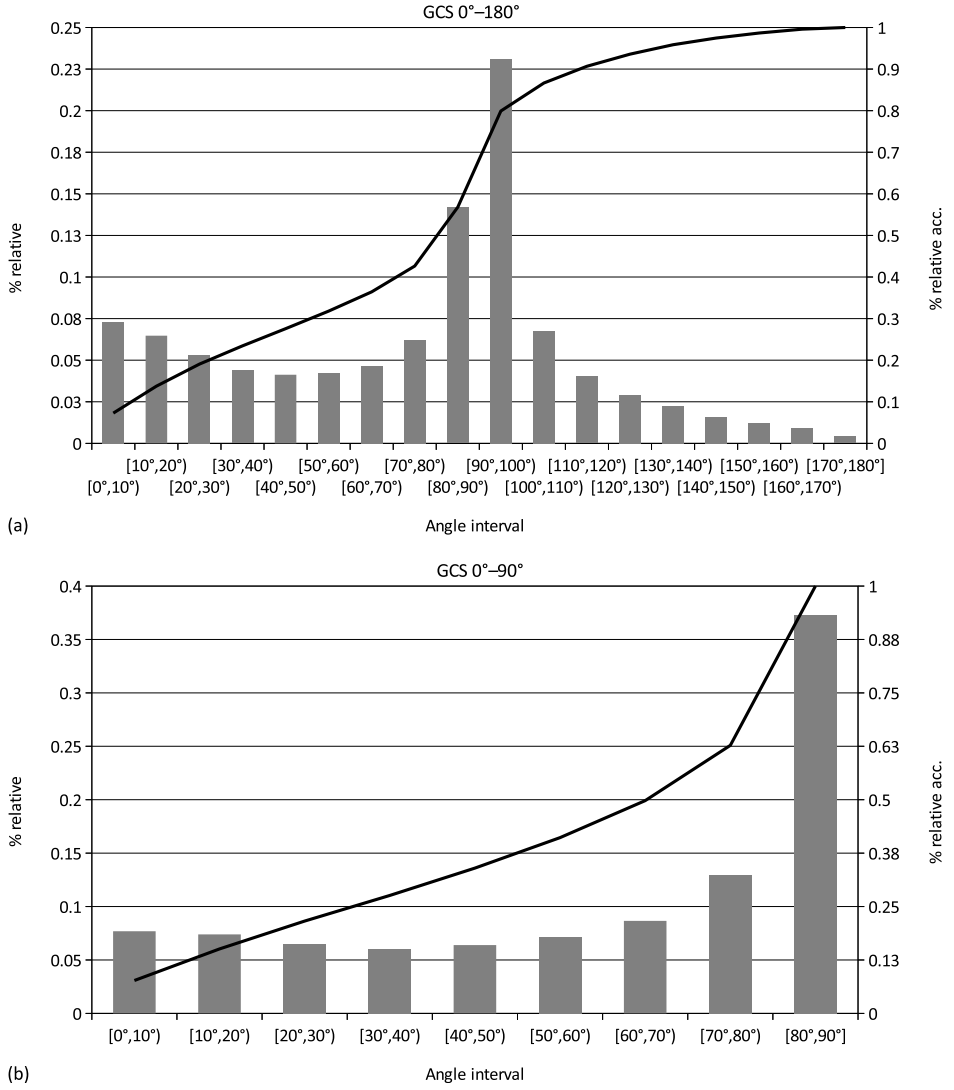


Figure 31: Histograms for the difference of the vectors from the original and reconstructed data set, for *GCS* and *VRTD*, no boundary switch curves were used in the reconstruction, grey bars show relative amount for the difference (left axis), black line shows accumulated amount (right axis)

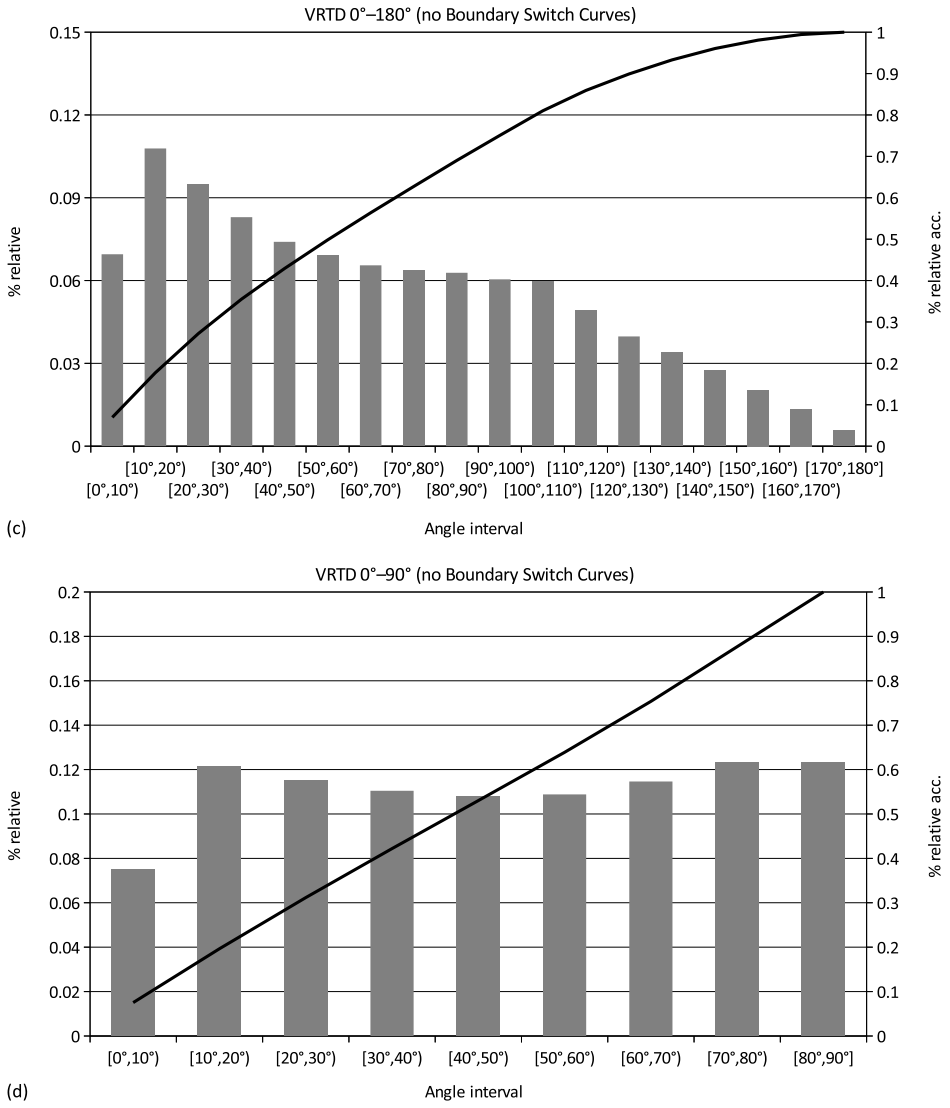


Figure 32: Histograms for the difference of the vectors from the original and reconstructed data set, for *GCS* and *VRTD*, no boundary switch curves were used in the reconstruction, grey bars show relative amount for the difference (left axis), black line shows accumulated amount (right axis)

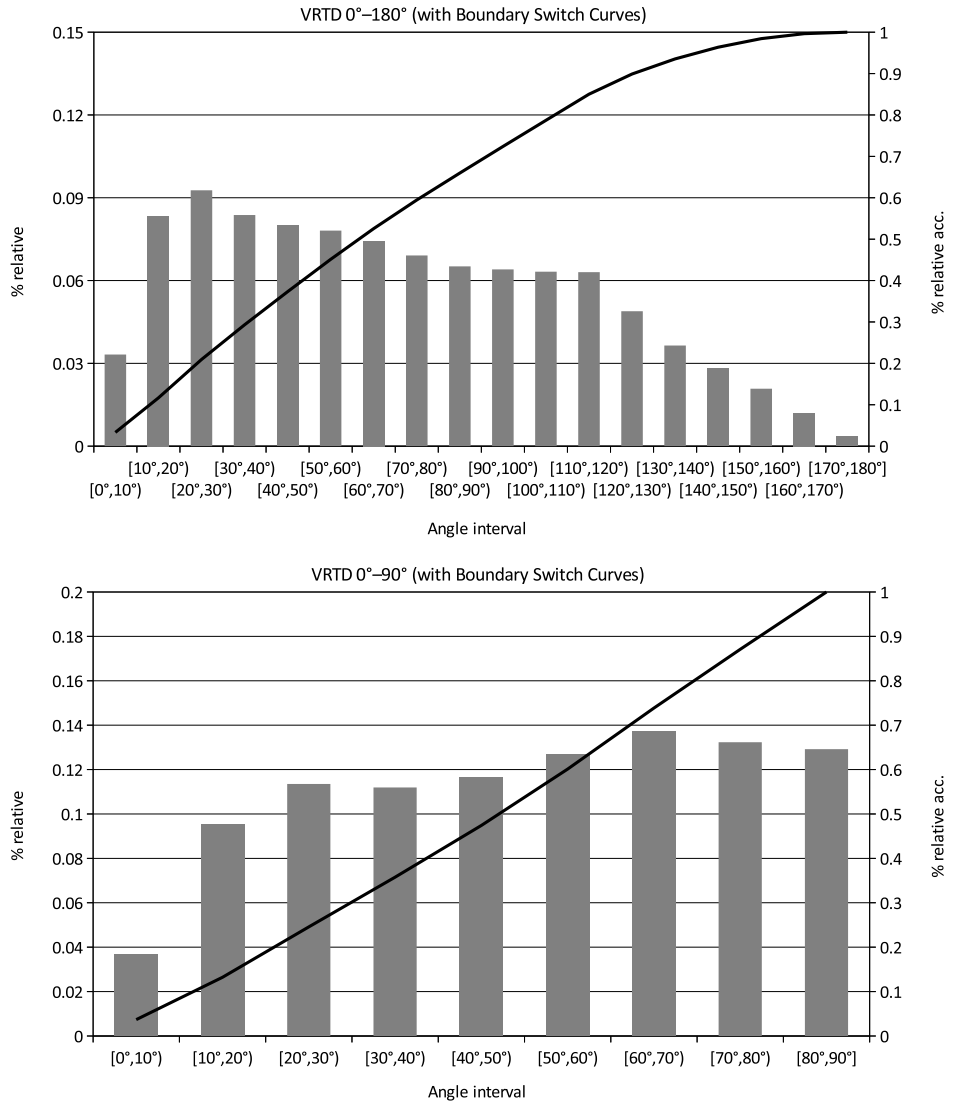
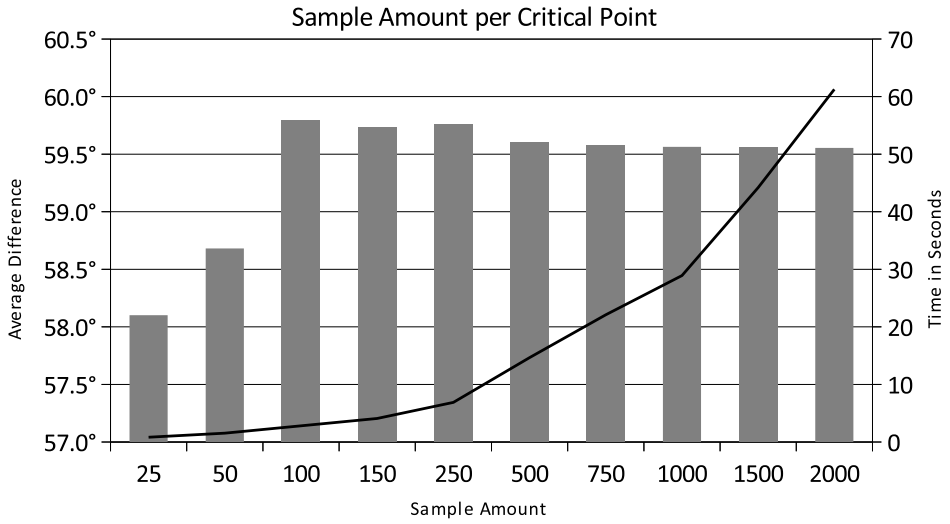
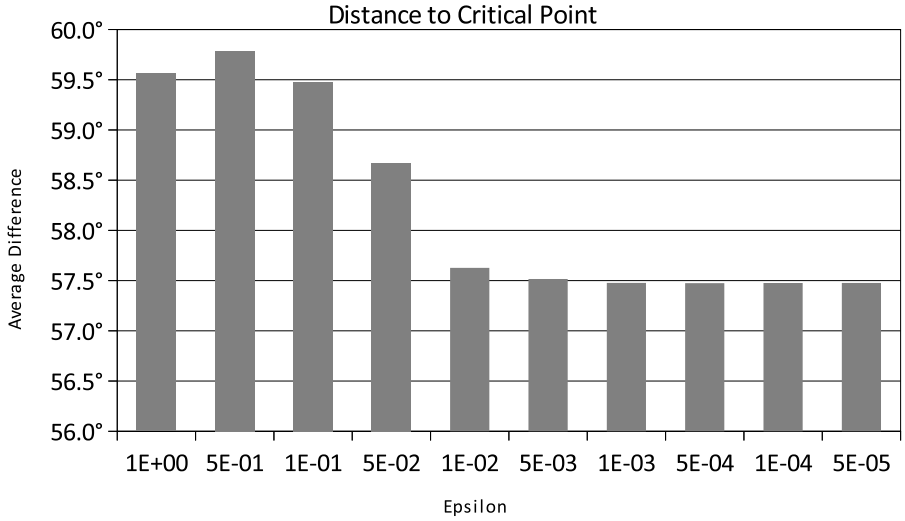


Figure 33: Histograms for the difference of the vectors from the original and reconstructed data set, for *VRTD*, boundary switch curves were used in the reconstruction, grey bars show relative amount for the difference (left axis), black line shows accumulated amount (right axis)



(a) Amount of samples per critical point increases exponentially (grey bars). The black line shows the time of computing the reconstruction.



(b) The distance of samples to the critical point decreases logarithmically (grey bars).

Figure 34: Histograms showing the changing of the average result of the reconstruction in relation to the two parameters, sample distance to critical points and sample amount per critical point.

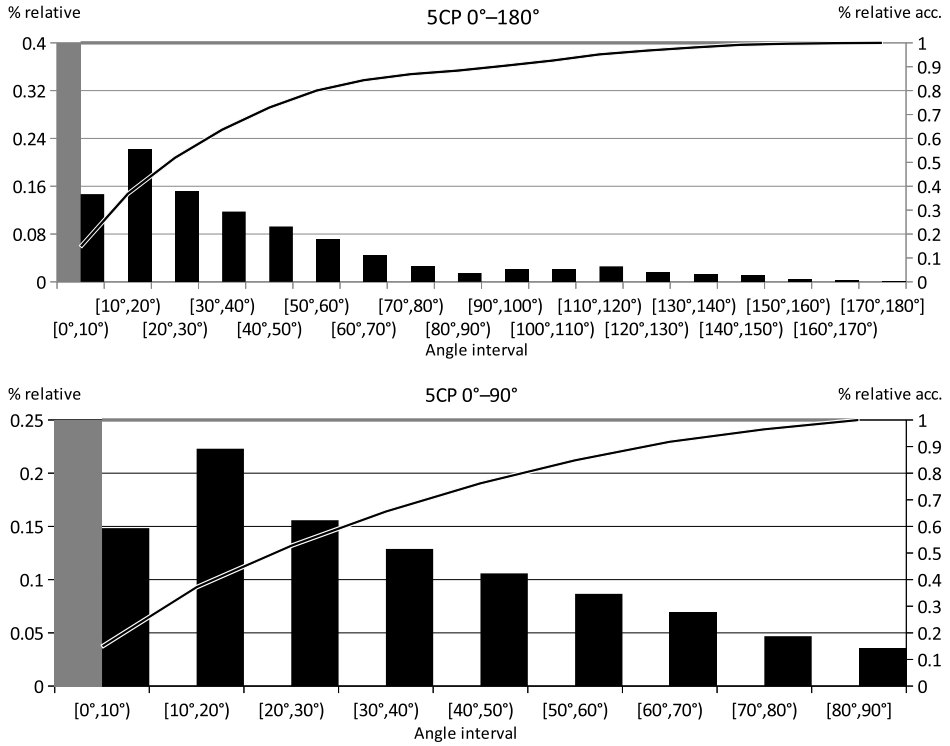


Figure 35: Histograms for the difference of the vectors from the original and reconstructed data set, grey bars and lines represent the reconstruction in cells with critical point, black ones also include the diffusion. The grey line is everywhere at 1.0 and the grey bar is 1.0 for [0°,10°].

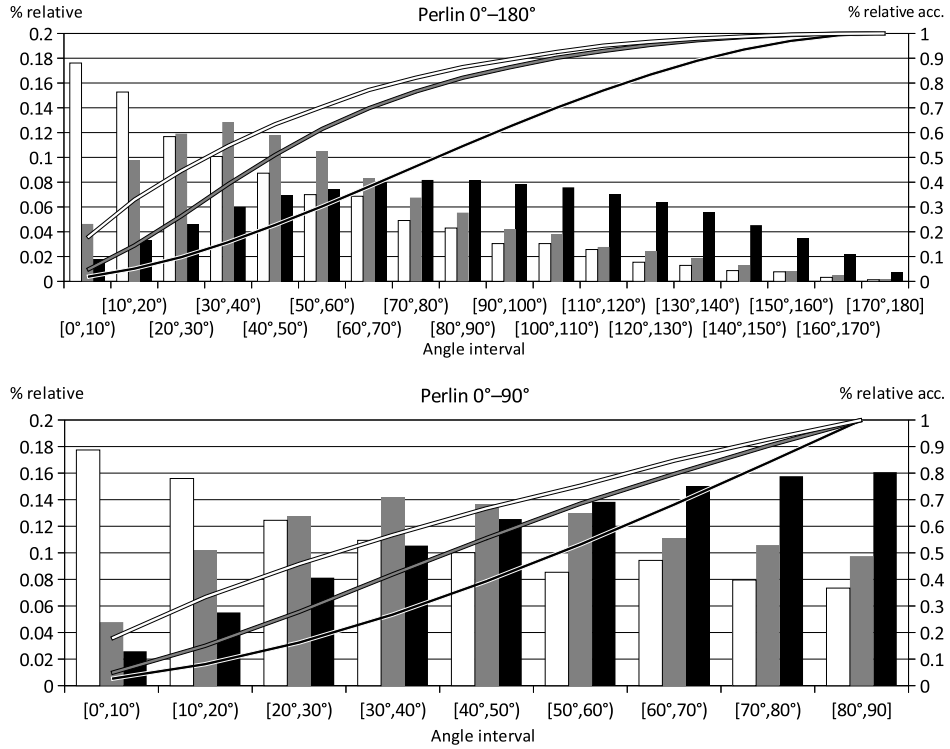


Figure 36: Histograms for the difference of the vectors from the original and reconstructed data set, white bars and lines represent the reconstruction in cells with critical points, grey ones also include the diffusion and black ones take additionally the reconstruction of separation lines into account.

8

CONCLUSION

To conclude this diploma thesis this chapter sums up and interprets the results and gives a perspective.

8.1 SUMMARY OF THE RESULTS

This diploma thesis has shown what possibilities exist to reconstruct vector fields from topological data, namely critical points and boundary switch curves. This reconstruction was considered in a global (Chapter 4) and a local sense (Chapter 5).

To practically use the insights of the reconstruction, a prototype was developed which can read different input formats (Chapter 8). With the input data a reconstruction can be done whose result can again saved to file. One notable aspect of the prototype is its ability to visualise the results of the reconstruction together with the input data.

8.2 INTERPRETATION AND COMPARISON

Chapter 7 gave an evaluation of both approaches. The global reconstruction revealed as an unsatisfying approach. It could not be used to reconstruct a majority of vectors right. Nevertheless, it seems that the approach needs some more attention. Probably further research will make it work better. As section 7.2 showed, at least for one data set an interesting distribution of the reconstructed vectors could be observed: the majority of the reconstructed vectors were orthogonal to the original ones. This leads to the assumption that the approach needs further tuning.

On the other hand, the local reconstruction was most successful when considering cells of the grid which contain critical points. As these points are the only information used in this approach, first improvements of the approach should start there. For certain critical points, a nearly perfect reconstruction of the vectors of the cell containing these points could be achieved. This should be expanded to all types of first-order critical points. The diffusion of the vector information showed that the reconstructed vectors could be successfully used to generate vector information at cells where no critical points reside. The reconstruction of separation lines between critical points is one try to generate extra topological information

at cells where no critical points are observed. As the evaluation showed, this did not work as expected.

8.3 LIMITATIONS AND PERSPECTIVES

There are some limitations for the presented reconstruction which should be addressed here. First, the approaches use only first-order critical points. The two first approaches presented in chapter 2 work with higher-order critical points. It should be analysed if the construction there can be combined with parts of the approaches in this thesis. Namely, this thesis could support higher-order critical points for the reconstruction and both construction approaches could be extended to use trilinear interpolation, i. e. not depending on a tetrahedron net but on a regular or Cartesian grid.

As this work in the local approach only uses trilinear interpolation, it should be researched if it would have benefits for supporting tricubic interpolation. This was not done here because of the assessed amount of work.

Another limitation is the sole support for parameter-independent vector fields. Further research is left for reconstruction of vector field which change over time, for example.

The very next step for the global reconstruction will test if it is better to interpolate the Jacobians of the critical points than first sampling vectors around them and interpolating these.

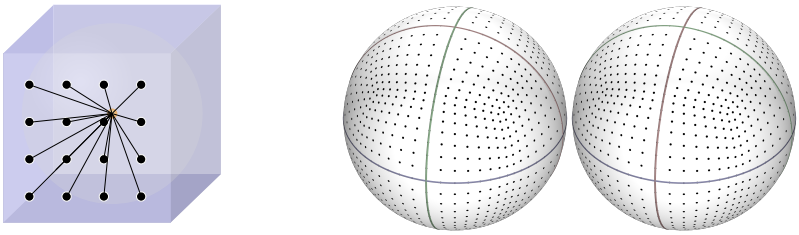
A

APPENDIX

A.1 DISTRIBUTE POINTS EQUALLY ON SPHERE SURFACE

There exist many approaches to distribute points most equally on the surface of a sphere. In fact, due to the nature of the sphere surface this task is by no means trivial. One can see an analogy in projecting the world map from a globe to a plane where there are also many projection styles. All of them will stretch and deform parts of the topological data of the map. It is easy to distribute points equally on a plane or a rectangle. Mapping this to the surface of a sphere changes the distribution.

A first idea is to take a regular cuboid and equally distribute on each of the six faces an amount of n^2 points. Mapping all these $6 \cdot n^2$ to the inscribed sphere of the cube gives a first approximation for the problem, see Figure 37. The black lines connect the centre of the inscribed sphere with points on the cube surface. Where the intersect with the sphere surface, a point for the solution is generated.



(a) Strategy of the approach

(b) Result of the approach for 1350 ($= 6 \cdot 15^2$) points

Figure 37: A simple approach for equally distributing points on a sphere

A nearly perfect algorithm is given by [8]. The JFlowVis Framework uses a modified version, mentioned in [12]. It uses a spiral that goes from one pole of the sphere to the other. The next listing gives the pseudo code.

Listing 1: Algorithm for distributing points nearly equally on a sphere's surface ([12])

```
public static double[][] distribute( int n, double eps ) {
  // eps is distance from centre
  // n is amount of points to be generated
  double[][] result = new double[n][3]; // holding the result
```

```
  double[] theta = new double[n];
  double[] phi = new double[n];
  for ( int k = 1; k <= n; k++ ) {
    double h = -1.0 + 2.0 * (k - 1.0) / (n - 1.0);
    theta[k - 1] = acos(h);
    if ( k == 1 || k == n ) {
      phi[k - 1] = 0.0;
    } else {
      phi[k - 1] = (phi[k - 2] + 3.6 / sqrt(n * (1.0 - pow(h, 2.0))))
          % (2.0 * PI);
    }
  }
  // convert spherical coordinates
  for ( int i = 0; i < phi.length; i++ ) {
    result[i][0] = Math.cos(phi[i]) * Math.sin(theta[i]) * eps;
    result[i][1] = Math.sin(phi[i]) * Math.sin(theta[i]) * eps;
    result[i][2] = Math.cos(theta[i]) * eps;
  }

  return result;
}
```

The complete Implementation can be found in DistributePointsOnSphere.java of the JFlowVisFW.

Equation A.1 gives a matrix orientated form of equation 5.1. x, y, z is the location of the critical point, $\mathbf{J}_x, \mathbf{J}_y, \mathbf{J}_z$ are the row vectors of its Jacobian $\mathbf{J}$ and $\mathbf{v}_{000}, \ldots, \mathbf{v}_{111}$ are the eight vectors at the vertices of the cell. Note that $\overline{x}, \overline{y}, \overline{z}$ are abbreviations for $(1-x), (1-y), (1-z)$, respectively.

$$\mathbf{Ax} = \mathbf{b}$$

$$\begin{pmatrix} \overline{x}\overline{y}\overline{z} & \overline{x}\overline{y}z & \overline{x}y\overline{z} & x\overline{y}\overline{z} & \overline{x}yz & x\overline{y}z & xy\overline{z} & xyz \\ -\overline{y}\overline{z} & -\overline{y}z & -y\overline{z} & \overline{y}\overline{z} & -yz & \overline{y}z & y\overline{z} & yz \\ -\overline{x}\overline{z} & -\overline{x}z & \overline{x}\overline{z} & -x\overline{z} & \overline{x}z & -xz & x\overline{z} & xz \\ -\overline{x}\overline{y} & \overline{x}\overline{y} & -\overline{x}y & -x\overline{y} & \overline{x}y & x\overline{y} & -xy & xy \end{pmatrix}_{4\times 8} \cdot \begin{pmatrix} \mathbf{v}_{000} \\ \mathbf{v}_{001} \\ \mathbf{v}_{010} \\ \mathbf{v}_{100} \\ \mathbf{v}_{011} \\ \mathbf{v}_{101} \\ \mathbf{v}_{110} \\ \mathbf{v}_{111} \end{pmatrix}_{8\times 1} = \begin{pmatrix} 0 \\ \mathbf{V}_x \\ \mathbf{V}_y \\ \mathbf{V}_z \end{pmatrix}_{4\times 1} \tag{A.1}$$

BIBLIOGRAPHY

[1] Robert Bringhurst. *The Elements of Typographic Style*. Version 2.5. Hartley & Marks, Publishers, Point Roberts, WA, USA, 2002.

[2] Paul Falstad. 3-d vector field simulation, 5 2011. URL http://www.falstad.com/vector3d/. last visited: March 04th, 2012.

[3] P.A. Firby and C.F. Gardiner. *Surface topology. 2nd ed.* Ellis Horwood Series in Mathematics and its Applications. New York etc.: Ellis Horwood. 200 p. , 1991.

[4] B. Jähne. *Digitale Bildverarbeitung*. Springer, 1993. ISBN 354056926x. URL http://books.google.de/books?id=5EShlIsf4TQC.

[5] Marcos Lage, Fabiano Petronetto, Afonso Paiva, Hélio Lopes, Thomas Lewiner, and Geovan Tavares. Vector field reconstruction from sparse samples with applications. In *Sibgrapi 2006 (XIX Brazilian Symposium on Computer Graphics and Image Processing)*, pages 297–304, Manaus, AM, october 2006. IEEE. doi: 10.1109/SIBGRAPI.2006.47. URL http://thomas.lewiner.org/pdfs/vector_field_sibgrapi.pdf.

[6] Marcos Lage, Rener Castro, Fabiano Petronetto, Alex Bordignon, Geovan Tavares, Thomas Lewiner, and Hélio Lopes. Support vectors learning for vector field reconstruction. In *Proceedings of the 2009 XXII Brazilian Symposium on Computer Graphics and Image Processing*, SIBGRAPI '09, pages 104–111, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3813-6. doi: http://dx.doi.org/10.1109/SIBGRAPI.2009.20. URL http://dx.doi.org/10.1109/SIBGRAPI.2009.20.

[7] Wolfgang Nolting. *Grundkurs Theoretische Physik 3 Elektrodynamik*. Springer, 2011. doi: 10.1007/978-3-642-13449-4.

[8] E. A. Rakhmanov, E. B. Saff, and Y. M. Zhou. Minimal discrete energy on the sphere. *Mathematical Research Letters*, 1:647–662, 1994.

[9] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, ACM '68, pages 517–524, New York, NY, USA, 1968. ACM. doi: 10.1145/800186.810616. URL http://doi.acm.org/10.1145/800186.810616.

[10] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Saddle connectors - an approach to visualizing the topological skeleton of complex 3d vector fields. In G. Turk, J. J. van Wijk, and R. Moorhead, editors, *Proc. IEEE Visualization 2003*, pages 225–232, Seattle, U.S.A., October 2003. URL http://tinoweinkauf.net/.

[11] Holger Theisel. Designing 2d vector fields of arbitrary topology. *Comput. Graph. Forum*, 21(3):1–10, 2002.

[12] Knud Thomsen. Generalized spiral points: further improvement, February 2005. URL http://sitemason.vanderbilt.edu/page/hmbADS#spiral. last visited March 17th, 2012.

[13] T. Weinkauf. Krümmungsvisualisierung für 3d-vektorfelder. Master's thesis, University of Rostock, Department of Computer Sciences, Institute of Computer Graphics, June 2000. URL http://tinoweinkauf.net/.

[14] T. Weinkauf. *Extraction of Topological Structures in 2D and 3D Vector Fields*. PhD thesis, University Magdeburg, 2008. URL http://tinoweinkauf.net/.

[15] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Topological construction and visualization of higher order 3D vector fields. *Computer Graphics Forum*, 23(3):469–478, September 2004. URL http://tinoweinkauf.net/. Eurographics 2004, Grenoble, France, August 30 - September 03.

[16] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Boundary switch connectors for topological visualization of complex 3d vector fields. *Data Visualization 2004*, 04:0, 2004.

[17] D. V. Widder. Chapter i introduction. In D.V. Widder, editor, *The Heat Equation*, volume 67 of *Pure and Applied Mathematics*, pages 1 – 16. Elsevier, 1975. doi: 10.1016/S0079-8169(08)62165-0. URL http://www.sciencedirect.com/science/article/pii/S0079816908621650.

COLOPHON

This thesis was typeset with LaTeX 2ε using Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL* were used). The listings are typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.)

The typographic style was inspired by Bringhurst's genius as presented in *The Elements of Typographic Style* [1]. It is available for LaTeX via CTAN as "`classicthesis`".

Final Version as of April 10, 2012 at 11:32.

SELBSTSTÄNDIGKEITSERKLÄRUNG

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur mit erlaubten Hilfsmitteln angefertigt habe.

Magdeburg, 10. April 2012

Alexander Schulze