TITEL

Image Synthesis Methods For Texture-Based Visualization of Vector Fields

A DISSERTATION SUBMITTED TOWARDS THE DEGREE DOCTOR OF ENGINEERING OF THE FACULTY OF MATHEMATICS AND COMPUTER SCIENCE OF SAARLAND UNIVERSITY

BY
VICTOR MATVIENKO

Saarbrücken
September 2016

Day of Colloquium:	June 7th, 2017
Dean of the Faculty:	Univ.-Prof. Dr. Frank-Olaf Schreyer
Chair of the Committee:	Prof. Dr. Dietrich Klakow
Reporters:	Prof. Dr. Jens Krüger Prof. Dr. Philipp Slusallek
Academic Assistant:	Dr. Daria Stepanova

SAARLAND UNIVERSITY

Abstract

Faculty of Mathematics and Computer Science
Graduate School of Computer Science

Doctor of Engineering

Image Synthesis Methods For Texture-Based Visualization of Vector Fields

by Victor MATVIENKO

This thesis presents several novel models and techniques for texture-based vector field visualization. The methods of discrete image analysis are consistently applied to popular visualization techniques such as Line Integral Convolution resulting in new insights in the structure of these methods and their possible developement.

Starting with formulation of a computation model for evaluation of visualization texture quality score, important attributes of an efficient visualization are identified, including contrast and specific spatial frequency structure. As a result, several visualization techniques with increasing applicability are designed aiming to optimize these quality features. Finally, a discrete probabilistic framework is suggested as a generalization of the state-of-the-art as well as presented here texture-based flow visualization methods. Based on this theoretical foundation, a novel multi-scale three-dimensional flow visualization approach is presented.

The visualization results are demonstrated on a large variety of examples delivered by several software programs developed for the purposes of this work. They are evaluated using formal quality metrics combined with formal statistical methods and an expert survey.

The scientific contribution of the thesis is the foundation for several journal and conference publications based on the presented here materials.

SAARLAND UNIVERSITY

Abstract

Faculty of Mathematics and Computer Science
Graduate School of Computer Science

Doctor of Engineering

Image Synthesis Methods For Texture-Based Visualization of Vector Fields

by Victor MATVIENKO

Diese Arbeit stellt mehrere neue Modelle und Techniken für die texturbasierte Vektorfeld-Visualisierung dar. Das Verfahren der diskreten Bildanalyse wird auf mehrere populäre Visualisierungstechniken wie Line Integral Convolution angewandt. Dieser Ansatz führt zu den neuen Einsichten in die Struktur dieser Methoden und deren möglichen Entwicklung.

Beginnend mit der Formulierung eines Modells zur Bewertung der Visualisierungsqualitätsmetrik, wichtige Merkmale eines effizienten Visualisierung sind identifiziert worden, einschließlich Kontrast und Ortsfrequenzstruktur. Als Ergebnis werden verschiedene Visualisierungstechniken mit zunehmender Anwendbarkeit entworfen, um diese Qualitätsmerkmale zu optimieren. Schließlich wird eine diskrete Wahrscheinlichkeits-Framework als Verallgemeinerung des State-of-the-art vorgeschlagen und die texturbasierte Strömungsvisualisierungsmethoden werden vorgelegt. Auf dieser theoretischen Grundlage wird ein neuartiger multi-scale dreidimensionaler Strömungsvisualisierungsansatz vorgestellt.

Die Visualisierungs Ergebnisse werden demonstriert auf zahlreichen Beispielen von mehreren Softwarekomponenten entwickelt für die Zwecke dieser Arbeit. Die gelieferte Daten wurden mit formalen Qualitätsmetriken und statistischen Methoden ausgewertet und zusätzlich mittels einer Expertenbefragung evaluiert. Die Ergebnisse der Arbeit wurden auf Konferenzen vorgestellt und als wissenschaftliche Artikeln in Zeitschriften publiziert.

Acknowledgements

The author is deeply grateful to Dr. Prof. Jens Krüger for creating the atmosphere of encouragement and friendly collaboration. Without his long-term guidance and support the ideas presented in this thesis would never be brought to life. The author also thanks Dr. Tino Weinkauf for the fruitful discussion of the proposed here results and valuable inputs.

Contents

List of Figures

List of Tables

Dedicated to my father

Chapter 1

Introduction

1.1 Outline

This work represents a summary of several research projects conducted by the author within Interactive Visualization and Data Analysis group at Saarland University under the supervision of Prof. Dr. Jens Krueger between 2011 and 2015. Within this period we tried to explore several intriguing aspects of Texture-Based (or Dense) Flow Visualization (DFV), ultimately aiming to develop new models and analysis methods within this area, advancing the state-of-the-art. Aiming for the above goal we required a powerful formal basis at the very core of our approach. For this purpose we rely on the solid foundation of the theoretical concepts and methods used in discrete image processing. Consistently applying this framework to texture synthesis methods commonly used for flow visualization allows us to get a novel perspective on some of the long-standing problems in the domain.

The structure of the thesis to some extent follows a development of the authors own knowledge on the subject, expanding from the very fundamental questions (What is DFV actually? What are the criteria of good DFV?) to a broader and more subtle problems and more sophisticated and generic findings. Thus, the flow of ideas does not necessarily coincide with the chronological order of journal and conference publications, underlying the chapters, but rather represents a coherent story of exploration and discoveries. First of all, it is necessary to put the work in the context of the existing research. In Section 1.6 we look at the related methods and models and the contribution of this work to the domain of texture-based flow visualization is discussed in the following chapters.

A key point for starting any endeavor is to define a measure of success. We address this question in detail in Chapter 2, investigating different ways to evaluate the quality of DFV images. The two ways to measure quality of DFV – user studies and automated metrics – both have their advantages and drawbacks. We mainly focus on the second one as it is automated and thus can be used in massive experiments. First, we propose an integral metric, based on automatic image processing and validate it using a user study. Our finding is the alignment of expert evaluations with our metric allows us to state that the measured visualization attributes correlate with perceived quality. In particular we recognize the role of contrast and spatial frequency control the visualization images with respect to the flow direction. This observation allows us to develop image processing methods optimizing for these criteria in the further chapters. We then employ our automated metric evaluation of the results combining with other evaluation approaches and putting it in the context of information theoretic framework, developed in the recent years for the visualization quality assessment. For the common scientific visualization

problems, we formulate the methods to explicitly control the following attributes of the target visualization motivated by our findings of the quality study:

- contrast;
- the spatial frequency;
- coherency along the flow.

We describe our first visualization model in Chapter 3 for the visualization of gradient vector fields with the means of isocontours. In this specific case of a vector field, the existence of a scalar potential allows us to introduce a scalar transfer function and to implement the requirements to the target visualization within the constructive design of this function. We were able to provide a function with adaptive spatial frequency management, which is being a valuable feature of the visualization technique, also determines the level of represented detail of the data. The constructed wave transfer function combines several harmonics to achieve smooth line pattern, and the underlying scalar field is used to guide the image synthesis process.

The approach using wave functions with varying frequencies to control the frequency of generated images for gradient flow visualization can be further exploited for an arbitrary flow. In Chapter 4, we suggest a novel model based on wave interference. We implicitly construct the scalar field, which is naturally available in the previous setting, with the means of a heuristically constructed distance function. Moreover we formulate the optimal visualization as a minimizer of a discrete quadratic form representing the constraints on the resulting image, resulting in a problem of numeric eigenvector computation for a large sparse matrix.

The wave-functions technique is significantly refined and improved in Chapter 5, where the heuristic distance is not a part of the model. Instead, the application of iterated Gabor filters, motivated by perceptional studies allows us to achieve similar looking images with higher accuracy of flow representation and as better frequency control compared to the previous approach.

The idea of discrete formulation of the constraints on the target visualization image and further computing it as an optimum point of some objective, first considered in Chapter 4, is further developed into an instrumental framework in Chapter 6. We suggest a discrete probabilistic representation of dense flow visualization, and demonstrate that several popular dense flow visualization methods can be represented within this framework, allowing for analysis of some of their non-trivial features. Then, we suggest a novel visualization method based on probabilistic particle mixture model, suitable for extracting complex flow structures. One of the distinguishing features of the developed DFV method is its applicability to volumetric flows.

Finally, the models and techniques presented in this work are summarized in the concluding chapter.

1.2 Why Flow Visualization

The first successful attempts to study flows date back to the early days, ages before the appearance of computer graphics or even the invention of computers. It was already

FIGURE 1.1: The air flow from the wing of an agricultural plane made visible by a technique that uses colored smoke rising from the ground. [75]

at that time when scientists realized that making flow patterns visible could provide a valuable insight into the underlying physical processes. This observation led to a variety of experimental visualization approaches for instance those, that highlight the flow lines injecting a foreign material or energy into it in certain positions (see Figure 1.1 for example). For a detailed overview of experimental techniques consider the work by Post et al. [85]

Numerous visualization techniques, being used nowadays, are empowered by the fascinating capabilities of the modern computer graphics. The summary of these approaches related to this work is given in Section 1.6. Computer graphics flow visualization has proved to be of a great usefulness in a wide spectrum of scientific and engineering disciplines, ranging from a daily weather forecast to complicated research aids in computational fluid dynamics and aircraft design.

Apart from the vast number of applications what probably makes this area of scientific visualization of particular significance is an especially high complexity of the problems requiring flow visualization. Up to the present time lots of problems in flow mechanics bear the reputation of being among the most challenging in physics. Turbulence is one good example for that, which can be illustrated by the famous quote, attributed to Werner Heisenberg [40] "When I meet God, I am going to ask him two questions: Why relativity? And why turbulence? I really believe he will have an answer for the first."

1.3 The Visualization Pipeline

Before considering the flow visualization methods, let us take a look at the general framework, commonly used to visualize scientific data, known as the visualization pipeline. The following description follows the schema in Figure 1.2.

The initial step of the pipeline is production of the numerical data by the means of measurement (acquiring from sensors) or numerical simulation. Then, the obtained data is enriched and enhanced, reducing its amount and/or improving the content quality. This step would include, for example, interpolation, sampling, and noise filtering. In the visualization mapping stage the abstract physical quantities are cast into certain visual primitives and attributes such as shape, light and color. The purpose of this step is to

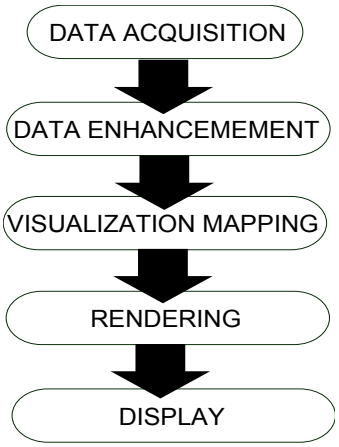

FIGURE 1.2: The visualization pipeline as presented by Post et al. [85]

derive entities suitable for the direct visualization from the raw data. In order to produce displayable images from the visualization data various rendering techniques are applied. The view transformation and lighting are handled at this step. Finally, the rendered data has to be displayed. This can mean showing the direct output of the rendering process or involve more complicated activities like color map manipulations and animations.

1.4 Overview of Flow Visualization Methods

According to Post et al. [86], vector field visualization methods can be classified into four categories: direct, texture-based, geometric, and feature-based methods. As illustrated by Figure 1.3 the four approaches vary in the amount of computation and the number of data processing steps required to produce the final image. In this sense they are differently integrated into the visualization pipeline regarding the data enhancement, visualization mapping and rendering steps.

1.4.1 Direct Flow Visualization

The methods in this category provide an immediate representation of the flow with a straightforward mapping of the data onto geometric primitives (like glyphs or arrows) or using color coding. Despite their simplicity, the solutions of this kind might be very useful, especially if one is interested in the short-term behaviour of the flow, rather than in the long-term behaviour.

1.4.2 Dense (Texture-Based) Flow Visualization (DFV)

The result of dense flow visualization is usually a dense fuzzy image where the flow is represented by the means of the intensity or color distribution. This is usually achieved by smearing a noise texture into the direction of the flow, thus introducing correlation along flow lines. A more detailed review of these methods can be found in Section 1.6.1.

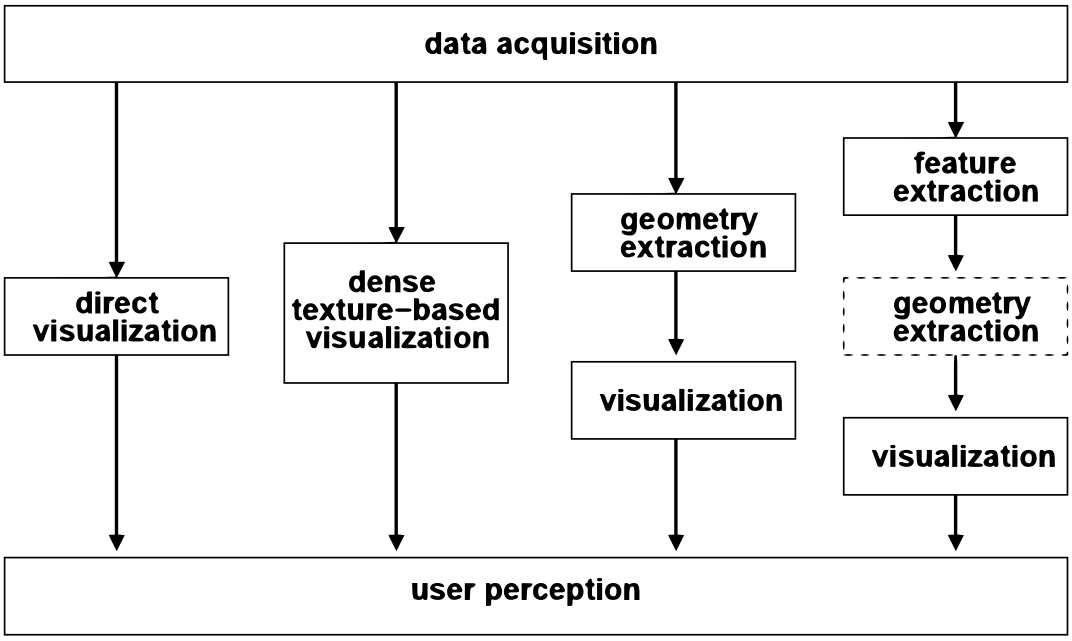

FIGURE 1.3: Classification of flow visualization techniques [61]: (left) direct, (middle-left) texture-based, (middle-right) based on geometric objects, and (right) feature-based.

1.4.3 Geometric Flow Visualization

This type of techniques first integrate the flow data to produce primitives, such as curves, surfaces or points, that have the geometry reflecting some of the aspects of the flow behavior. The examples of integral curves are streamlines (curves, that are tangential to the flow everywhere), pathlines (trajectories that individual fluid particles follow) and streaklines (the locus of points of all the fluid particles that have passed continuously through a particular spatial point in the past). We will explore the advantages of this approach and numerous variations of streamline visualization in Section 1.6.3.

1.4.4 Feature-Based Flow Visualization

This approach implies an intense processing and filtering of the original data, resulting in a compact representation of the most interesting for the researcher features such as important physical phenomena or topological information. Typical examples are critical points (vortices, saddle points) and separatricies (lines that separate regions of similar behaviour). Such sparse flow visualization can be thought of as the visualization of derived data. This derived topological data is often used as well for the enhancement of the other types of methods (consider for example the feature-based streamline placement, discussed in 1.6.3) A good overview of research on feature-based flow visualization is provided by Post et al. [86].

1.5 Mathematical Basis for Flow Visualization

This section briefly describes the basic mathematical concepts commonly used in flow visualization, mentioning specific choice of methods explored and extended within this work. Here we only discuss the basics of flow analysis, leaving the explanation of details of image processing methods, and the corresponding apparatus of linear algebra and probability theory which are at the core of our work to be introduced later in the corresponding chapters.

Generally the flow $v \in R^n$, within and n-dimensional domain $\Omega \subseteq R^n$ is described by the derivatives of the position $p \in \Omega$ with respect to time $t \in R$ as following:

$$v = \frac{dp}{dt}; \tag{1.1}$$

Throughout this work two-dimensional vector fields defined in R^2 are considered.

For most practical applications, the data v is not known in its analytical form, but obtained as a numerical solution of a differential equations system or as discretized output of flow measurement tools. For instance, flows used for the example images in Chapter 2 and Chapter 4 were produced with the Navier-Stokes Equation solver, presented in the work by Krüger and Westermann [57]. The flow data is thus usually given as a set of samples v_i defined on the vertices of a grid, but is supposed to be continuously reconstructible with some filter h as $v(p) = \sum_i h(p - p_i)v_i$. The ideas and methods presented in this work are independent of the sampling and interpolation schemes, in the sense that to benefit from them any reasonable reconstruction of the vector field is sufficient. In the accompanying implementation, rectilinear (Cartesian) grids were used along with bilinear interpolation kernel.

To obtain an intuition of the trajectory of long-term movement in the flow many DFV visualization methods require integration over time to compute a path $p(t)$:

$$p(t) = p_0 + \int_0^t v(p(\tau), \tau) d\tau, \tag{1.2}$$

which can be viewed as a solution of the initial value problem for Equation 1.1 with starting point p_0. Even if the flow data v is not given in the discrete form, still the equation above often can not be solved analytically for $p(t)$. As a consequence, it commonly is solved with iterative numerical methods. The most basic representative of those, which was used for the examples in this work is the Euler method:

$$p_E(t + \Delta t) = p(t) + \Delta t \; v(p(t), t) \tag{1.3}$$

Given a time step Δt it computes the new position $p_E(t + \Delta t)$ moving from the old position $p(t)$ in the direction of the flow. Other commonly used schemes belong to the family of higher-order Runge-Kutta methods. They allow to obtain more accurate approximation of the integral curve in exchange for higher computational overhead.

1.6 Related Work on Flow Visualization

This chapter consists of an overview of contemporary research related to the evaluation, analysis and improvement of existing flow visualization methods which is the primary concern of this work, as discussed in Section 1.1. In particular we will consider a number of wide-spread dense flow visualization methods and focus on the studies on the quality of the resulting images. We will as well look at the subset of geometric based approaches formed by the set of streamline visualization techniques, as there is an evident analogy between them and the technique presented in Chapter 4.

1.6.1 Dense Flow Visualization Methods

The roots of texture-based methods date back to the early nineties, with the pioneering work on spot noise by Van Wijk [131], and the invention of Line Integral Convolution (LIC) method by Cabral and Leedom [9]. Since then, a vast number of improvements has been proposed including improvements in performance (e.g. many-core/GPU acceleration, methodical enhancements), extensions to novel domains, and quality improvements (e.g. 3D DFV, flow on surfaces, time dependent/animated visualizations). The following overview of the most widely used techniques briefly summarizes the basic principles of the key flow visualization methods. It is loosely based on the excellent state of the art report by Laramee et al. [61], which is referred to if the reader is interested in a more detailed exploration of the variety of modern dense vector field visualization approaches.

Spot Noise. The essence of this method, closely related to experimental flow visualization, is the generation of a set of intensity spots covering the image domain. Each spot can be thought of as a particle, warped over a small time step. The resulting texture consists of blended streaks in the direction of the flow and can be computed as in Equation 1.4:

$$f(x) = \sum a_i h(x - x_i, v(x_i)) \tag{1.4}$$

with v being the orignal flow field, h a spot function, having unit intensity in a small region around a random position x_i vanishing everywhere else and a_i is a scaling factor.

Multiple extensions of this technique existing nowadays usually deal with the shape of the spot function, optimal placing of spots and efficient parallel implementation. Variation of the spot shape allows spot noise to show not only vector field direction but the magnitude as well. However, it is worth mentioning that the direction of the flow and the critical points which are indispensable for flow analysis are much better revealed by the Line Integral Convolution method (see below).

Texture Advection. This approach forms a wide area of research, considering the animation of the flow with moving texture elements or small polygons. Due to considerable amount of computation required, it is widely combined with efficient parallel implementations utilizing the power of modern graphics processing units. For details consider for example the book by Weiskopf [127]. The mathematical grounds of such methods is an advection scheme, usually based on the backward coordinate integration. This means that in order to compute a frame for each particle its position in the previous

time step is computed:

$$x_{-h}(i,j) = x_0(i,j) + \int_{\tau=0}^{h} v_{-\tau}(x_{-\tau}(i,j))d\tau \tag{1.5}$$

where $x_0(i,j)$ is a starting position and x_{-h} is a previous position given the time step h.

This scheme is then exploited to visualize the flow of a colored medium in the vector field. Compared to the rarely used forward coordinate integration, it allows to avoid holes in the resulting frame. Texture advection is particularly useful for exploring unsteady flows through animation. However, time-dependent flows as well as time-varying visualization(animation) are out of scope of this work.

Line Integral Convolution. LIC with all its numerous modifications is arguably the most influential and widely researched and used texture-based flow visualization method at present day. This technique is used in this work as a solid standard of dense vector field visualization to test the developed methods for quality measurement and improvement against it as well as for comparison to the developed visualization techniques. Original method, published by Cabral and Leedom [9] operates on a 2D vector field on a Cartesian grid and a white noise texture $n(x)$.

The foundation of LIC lies in the notion of integral curves or streamlines, which capture essential information about the flow. Given a vector field $\mathbf{v}$ defined by a map $\mathbf{v} : \mathbb{R}^2 \to \mathbb{R}^2$, its integral curves can be defined using the arc-length parameterization by Equation 1.6:

$$\frac{d}{ds}\sigma(s) = \frac{\mathbf{v}}{|\mathbf{v}|} \tag{1.6}$$

in the regions where $|\mathbf{v}| \neq 0$.

Here we assume that for each $\mathbf{x}_p \in \mathbb{R}^2$ of the image domain, an integral curve σ_p can be computed analytically or numerically satisfying the initial condition $\sigma_p(0) = \mathbf{x}_p$. The output gray value image u of the Line Integral Convolution method is then defined as in Equation 1.7:

$$u(p) = \int_{-L}^{L} n(\sigma_p(t))k(t)dt \tag{1.7}$$

where $n(\mathbf{x})$ is the input noise and $k(t)$ is a weighting kernel such as a Gaussian or a box filter.

Under mild smoothness assumptions (local Lipschitz condition [105]) about the left hand side of Equation 1.6, the computed curves are unique, given the initial conditions. A local violation of this property (e.g., in the immediate vicinity of the critical points) is allowed by the LIC method.

The application of this filter results in a high correlation of the gray value along streamlines and a high gradient orthogonal to them. The resulting image is very much affected by the parameter choice. For example the smaller kernel length captures better small details in the flow whereas the larger kernel produces smoother images. The type of noise and post processing filters, such as histogram equalization can make a great perceptional difference as well. These effects are discussed in more detail in Section 2.3.

The ongoing research on LIC includes a vast number of derived approaches extending it to various types of grids, higher dimensions and striving for efficient implementations. Some of the most notable examples are listed below.

- The necessity to visualize flow on the surfaces, arising in many engineering applications, was addressed by Forssell [28] who suggested a LIC-based technique for surfaces represented by curvilinear grids and was latter extended by Battke, Stalling and Hege [3] for arbitrary grids in 3D tessellated with triangles.
- Volume LIC is an extension of LIC for 3D flows, developed by Interrante and Grosch [41]. One of the main perceptional challenges for dense visualization in 3D is occlusion and they address it by introducing some notion of sparseness, thus making a trade-off between the domain coverage and visual complexity.
- Fast LIC, introduced by Stalling and Hege [105], is the origin of most effective LIC implementation approaches. It eliminates redundant streamline and integral computations that are present in the original method, achieving an order of magnitude speedup.
- Another source of performance increase is the utilization of modern graphics hardware. Work that focuses specifically on GPU-based methods can be found in the book by Weiskopf [127]

1.6.2 Research On Dense Visualization Quality

A number of approaches to achieve better visual quality of the dense flow visualizations exists and there is still room for significant research in this area, for example the concept of double or manifold LIC computations might be of interest. Introduced by Okada and Lane [78], it has been revised and refined by Weiskopf [128] and Hlawatsch et al. [38].

The basic idea is to apply the LIC algorithm multiple times, using the filtered output of the previous LIC pass as input "noise" for the next pass. Contrast normalization and high-pass filtering is applied as well to enhance image details. Often multi-pass LIC is used to improve performance but it also tends to reduce aliasing problems in the image. These approaches can be coupled with advances in other fields, such as the introduction of wavelet noise [20] which provides better input noise fields, resulting in improved LIC images.

As a consequence of the ever-growing number of algorithms in DFV and vector field visualization techniques in general, the demand for the comparison and analysis of existing methods grows. To study the quality of different vector field visualization methods, Laidlaw et al. [60, 59] carried out extensive user studies with experts and non-experts for 2D vector field visualization techniques. The studies were designed to compare fundamentally different visualization techniques (such as dense LIC images and sparse glyph-based techniques). In their studies LIC turned out to be the least effective method. Pineo and Ware [82] took a different approach by using a numerical model of the primary visual cortex of the brain (Visual Area 1) to understand how humans follow the flow, and conducted a user study primarily to verify that simulation. Despite using similar visualization models to Laidlaw et al., they concluded that LIC and equally spaced stream lines are the most effective visualization techniques for this domain. Later,

Forsberg et al. [27] conducted a similar study, however, in that study dense methods were not considered. While all the previously mentioned studies were interested in a comparison of visualizations, they were aimed at providing an accurate answer to the question, which visualization method is better for a given set of tasks. Advances in any of the involved methods would require a new study.

In contrast to these user-centric studies, other researchers have developed models to numerically evaluate the quality of a given visualization. Stalling and Hege [104], as well as Stalling in his thesis [105], did a thorough analysis of the influence of various LIC parameters on signal-processing level. Cui et al. [22] considered the abstraction in multi-resolution information visualization methods, while Chen [11] focused on the abstraction level in network visualizations. Van Wijk [132] proposed an economic benefits and costs model to compute the value of a visualization. In contrast, Filonik and Baur [26] considered the aesthetics of a visualization as a quality metric. Recently, Jänicke and Chen [42] proposed a salience-based metric to evaluate visualizations in general, but also used flow visualizations including a LIC image as a specific example. They evaluated the quality of an image by comparing a salience map with a relevance map that is either user-defined or computed from the visualization data with a salience map. Most closely related to the idea presented in the Chapter 2 is the recently published work by Jänicke et al. [43]. In their work the idea of using the original vector field to evaluate the visualizations is exploited as well but preceding by visualization-specific reconstruction of the field from the image.

1.6.3 Streamline-Related Methods

The major perceptional challenge, arising from the use of common dense visualization techniques discussed above, is the low contrast of the output an the presence of the underlying noise. It motivated the introduction of double or manifold LIC computation by Okada and Lane [78]. The introduction of multiple passes increases the importance of a careful parameter adjustment, including the choice of the initial noise texture. Although there exist several thorough works on mathematical analysis of the influence of various LIC parameters (e.g., [104], [105]), generation of a LIC-based image with specified visual properties is still far from being a trivial task.

Geometric approaches, and streamline placement in particular, on the contrary allow full control over the contrast while rendering. However, the major challenge of streamline visualization inherent to all sparse methods is a trade-off between visual cluttering and the coverage of the field. Consequently, the quality of these methods is connected to the problem of optimal streamline placement, which has been of a great practical interest. Two common ways two deal with it can be identified as the feature-based and the density-based approaches. The purpose of the first one proposed by Verma et al. [121] is to draw attention to certain regions of interest, usually critical points. To achieve a good visualization in those areas they provide templates for seeding streamlines around different types of critical points, which previously have to be localized and classified. Generally this necessity for the advance segmentation of the field into regions of interest and the rest is application dependent and might limit the applicability of feature-based methods as pointed out by Chen et al. [14].

Density-based approaches seed streamlines managing the placement density throughout the whole image. The highest quality results are achieved with the method of Turk and Banks [119], who proposed to use low-pass filtered image as a measure of density and random optimization process to achieve on average the predefined density value. Less computationally expensive techniques include [45], which separate streamlines by estimating the placement density as Euclidian distance between them, and the work of Chen et al. [14], who defined a streamline similarity distance, that is in fact related to the thickness function, defined in Section 4.1.3. They showed as well, that their method gives better visual quality compared to [45], resulting in a smaller angular error of the flow reconstructed from visualization. The most advanced and complex approaches in the topology-based set of methods can be found in recent works by Liu et al. [63] and Wu et al. [135].

In Chapter 4 a hybrid technique is proposed, merging to some extent streamlines and dense visualization. There are hardly any attempts, exploring the possibilities such a fusion with an exception of [122], that tries to relate LIC to streamlines . It is mostly devoted to computing LIC-like images mapping a 1D texture on a line, but there is as well an observation that streamlines can be computed in a texture-based fashion iterating of a LIC kernel over a point output. Also—though not directly related to streamlines—the work of Taponecco and Alexa [111] on visualization with Markov random fields produces slightly similar in style images to the proposed technique.

Chapter 2

Quality Attributes of Texture-Based Flow Visualization

The ultimate objective of this chapter is to explore he proper ingredients for a *really good* texture-based flow visualization. Such problem statement inevitably leads to the question: what flow visualization is considered good? That might be a simple question for a trained human, when he is presented with a pair of images, but surprisingly, it turns out the universally accepted quality criteria for dense flow visualization doesn't exist in literature as follows from the literature survey in Chapter 1. One reason for it may be that flow visualization with the means of computer graphics is still rather young area. The lack of any means to measure the quality of visualization, except for a qualified specialists survey, motivated the main contribution of this chapter. The idea was to come up with some mathematical apparatus to quantify the useful properties of flow images. To generate the test pictures, the Line Integral Convolution technique (see Section 1.6.1 for description) was used as it is arguably the most popular method in the field. It has the reputation to be simple in implementation and yet—being a dense method—it is great for revealing important flow details, which could be missed when sparse methods such as streamlines are used.

2.1 Quality Metric of Texture-Based Visualization

The formulation of texture-based flow visualization quality metric introduced in this Chapter exploits the assumption that high contrast across flow lines is a desirable property for a flow visualization image. Although based on mostly common sense observations, the metric has showed a good correlation with the researches opinions about the images in the IVDA group. To verify this result we conducted the largest to the best of our knowledge web-based expert survey on the flow visualization quality estimation. It confirmed the initial hypothesis that the proposed metric can be used to predict the expert estimation of the image quality in up to 99% cases. Among the several experiments with the metric, described in Section 2.3, the one involving the application of the metric for the automated search for the best image by varying LIC inputs is of particular interest. It represents the attempt to make a good flow visualization based on LIC, scanning the parameter space of the algorithm. The resulting images were indeed recognized among the best LIC visualizations ever published, but the following fundamental challenges that one faces with this approach are nevertheless unavoidable.

- LIC by itself, as shown in Figure 2.4, produces very low contrast images, and requires histogram equalization and high-pass filtering to produce high quality results. Although, these additional steps negligibly complicate the computational process, they severely limit the predictability of the properties of the resulting image.

- The spatial frequency of the resulting images is an important quality parameter of the visualization images, for which LIC offers no straightforward way to control it. The issue of comparing the image quality of images with different frequencies is addressed in Section 2.3.5.

- While the influence of the underlying noise on the end result can not be exaggerated, yet it is the kind of a parameter that is difficult to optimize with a brute force approach due to its tremendous range of values. Even the multi-pass LIC concept (see Section 2.3.3), which restricts the input texture space to the output of a sequence of LIC processes, combined with contrast enhancement and high-pass filtering, leads to an exponential growth of input parameter range with the increase of the filter sequence length.

- The noise in the resulting images can not be eliminated completely and the whole strong dependency of the result on a random image parameter might look somewhat redundant once the optimization criteria is known. (In the experiments in Section 2.3 the quality metric was used as a such criterion to select the best output of the method).

Thus, the proposed metric allows to **evaluate** the quality of the resulting images, but not as much to **control** it. The next step towards an excellent flow visualization is to design visualization techniques based on the gained insights, instead of using LIC as a black box and evaluating the output. Consequently, an idea to incorporate the quality metric assumptions directly into an image synthesis process evolved into the contribution of the subsequent chapters of this work.

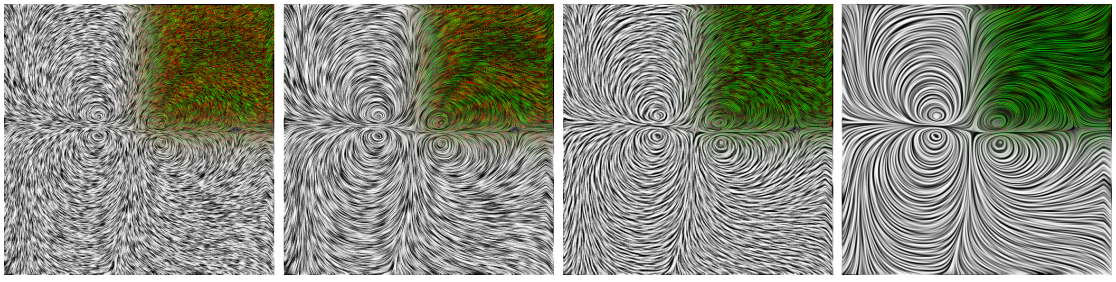

FIGURE 2.1: LIC images of different quality overlaid with a local quality metric proposed in this chapter. Green and red channels encode positive and negative components of the metric.

The remainder of the chapter is structured as follows: in the next section the basic idea of the metric is explained as well as the simplifications that make it useful in practice. In Section 2.3, a number of experiments are presented that we carried out with the metric in the IVDA group, different approaches are compared and new extensions to the LIC

method are proposed. In the conclusion a discussion of the results and a number of ideas for future research is proposed.

2.2 Constructing the Metric

In this section, a very general description of of a quality metric is proposed and then this formulation is refined and simplified to make it useful in practice. As the metric is supposed to express the scientific quality of an image for a human observer, a user study was conducted to train and verify the proposed formulation.

2.2.1 Basic Idea

The idea of the metric is very straightforward and based on the motivation of LIC images. What the LIC method (as well as other texture-based dense methods) tries to achieve is a low contrast along the path of a integral curve, while at the same time maximizing the contrast perpendicular to that curve[1]. This property of LIC images can be formulated as the local inequality in Equation 2.1; desirable to hold in each point of the image.

$$\left|\left\langle \nabla f, v^{\perp} \right\rangle\right| > \left|\left\langle \nabla f, v \right\rangle\right| \tag{2.1}$$

where v is the vector field, f is the intensity of the image, and the term $\langle \nabla f, v \rangle$ denotes the gradient of the image projected into the vector field. This means that we require the image-gradient magnitude in the direction of the flow to be lower than the image-gradient perpendicular (see Figure 2.2) to the flow. The best visualization under this assumption will be a black and white image depicting continuous lines perfectly aligned with the field and having a width of one pixel and distance of one pixel between them. Such a visualization is impossible to achieve with existing stream line placement methods, except for the trivial cases (e.g. the constant field), but it would allow users to recognize and follow the contours, which is an important process for understanding the flow, as detailed by Pineo and Ware's work [82].

As Equation **??** demonstrates he metric $\tilde{M}$ represents the difference between terms from two sides of the inequality in Equation 2.1 averaged over the entire image to get a single value per picture. Thus, large positive differences would correspond to the desired high metric value. Negative differences, arising in the points where the inequality in Equation 2.1 does not hold, would result in a low metric value. It is worth stressing that the computed quantity $\tilde{M}$ should not be interpreted as some inherent attribute of an image. What makes it interesting for the applications is the ordering it produces between several dense visualizations of the same vector field.

[1] Please note that for now we assume that the images are not downsampled for display, and users are capable of viewing the image in its full result, i.e. they have sufficient eyesight. The question of how to deal with downsampled images is discussed later in Section 2.3.

$$\tilde{M} = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} F\left(\left| \left\langle \nabla f_\omega, \frac{v_\omega^\perp}{|v_\omega|} \right\rangle \right| \right) - G\left(\left| \left\langle \nabla f_\omega, \frac{v_\omega}{|v_\omega|} \right\rangle \right| \right) \tag{2.2}$$

where Ω is the image domain in which $|v| \neq 0$. Functions F and G are two weighting functions that can be chosen arbitrarily. In the implementation the gradients are approximated using finite differences and a sum over all pixels in the image is computed and normalized by the number of pixels.

2.2.2 Simplification

The major issues with the metric introduced above are the two unknown functions F and G. This problem can be approached y the principle of *Occam's Razor* [134]. In this case this means to start with the most simple formulation of Equation 2.2 and gradually increase the complexity of the model until the results of the metric match the quality rated by a human observer. Following this train of thought, let's reduce the functions F and G to a simple weighting term λ, resulting in the following simplified metric M:

$$M = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \lambda \cdot \left| \left\langle \nabla f_\omega, \frac{v_\omega^\perp}{|v_\omega|} \right\rangle \right| - (1-\lambda) \cdot \left| \left\langle \nabla f_\omega, \frac{v_\omega}{|v_\omega|} \right\rangle \right| \tag{2.3}$$

This simple version of the metric was expected to give somewhat useful results, but as many of the perceptional processes in the human visual system are highly non-linear, one may assume that F and G should be some arbitrary functions that would probably be highly specific to each person. Surprisingly, the results computed from this simple version of the metric resulted in perfectly rated images in the IVDA group, with $\lambda = 0.5$ effectively eliminating the weighting term and thus making the formulation even simpler. This means, that given sets of images that were computed by randomly varying a number of parameters (e.g. integration length, choice of noise, histogram equalization) the ordering by metric-values matched the ordering that IVDA team had chosen.

2.2.3 Training and Verification of the Metric

As the experiments of the informal evaluation within the IVDA group of five (probably biased) researchers were not sufficient to consider the the metric applicable for a larger population, we decided to extend the experiment to a much larger group of independent subjects. For this purpose, we designed a browser-based survey platform.

Survey Setup. To conduct the survey, over 2300 DVF visualizations of 11 flow fields were selected from the previously computed database containing over half a million images. Most of those flow fields were also used for the illustrations in this chapter. The images were selected to cover a wide range of methods and settings. The survey database was then accessed by a web service developed for exclusively for this work. To acquire participants, invitation emails were sent to a number of experts in the field of flow visualization. The users were not provided with any compensation for performing the study.

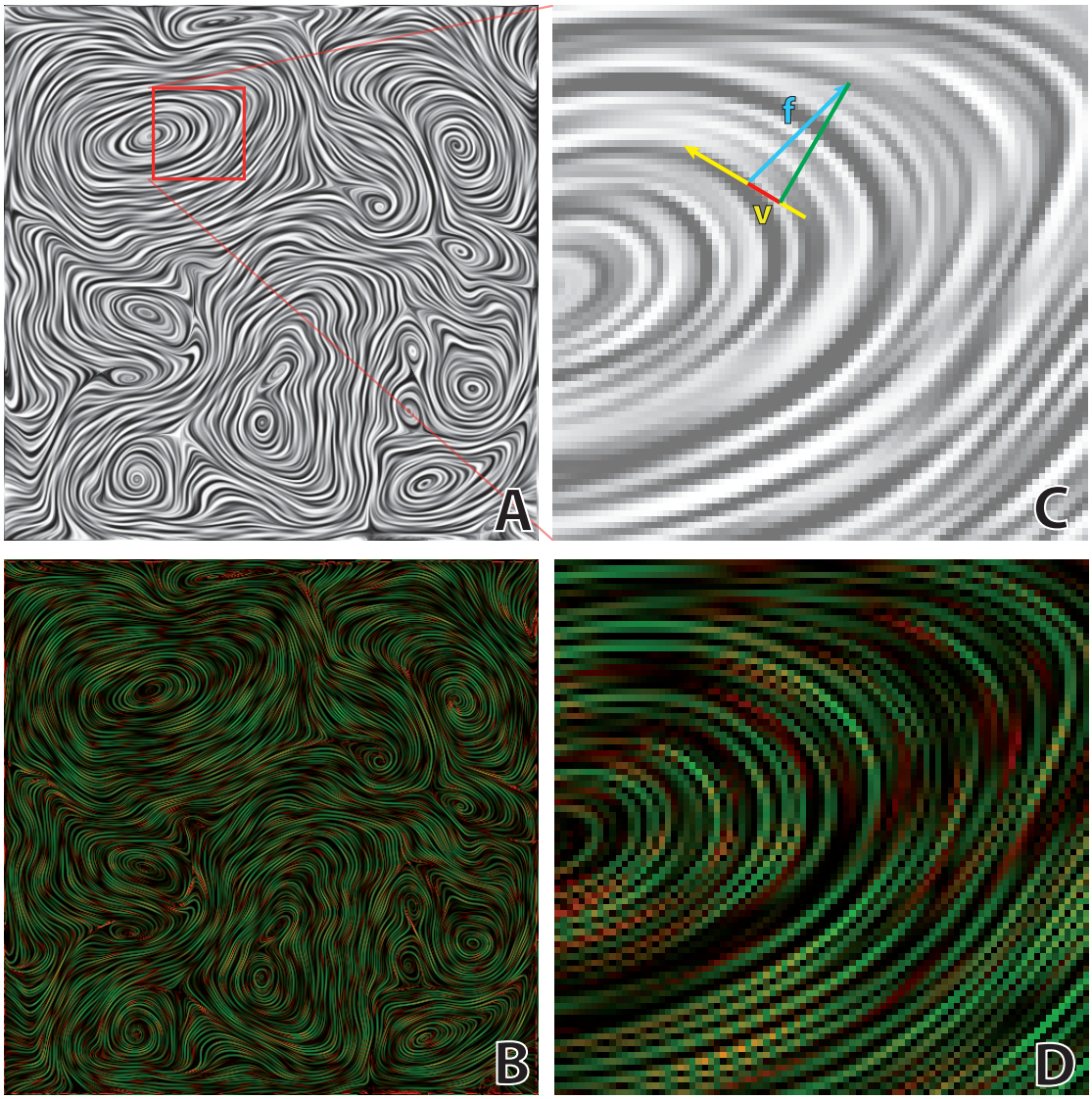

FIGURE 2.2: Image A depicts a DVF visualization with a small section magnified in image C. In this magnified image for a single pixel, the vector field direction (yellow) and the image gradient (blue) are shown. The red and green lines in this image indicate the decomposition of the gradient in the flow direction and perpendicular to the flow. Image D shows these two components for the magnified image and B for the entire domain.

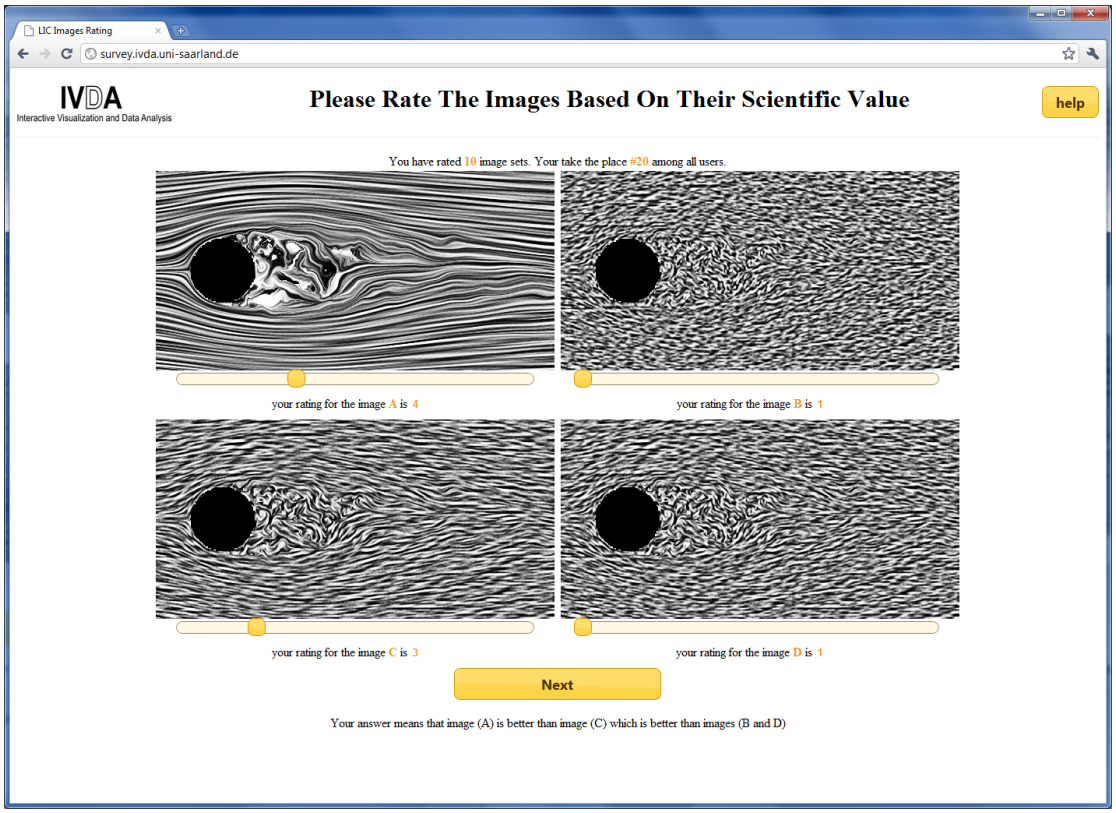

FIGURE 2.3: A screen-shot of the web survey. The user is asked to rate 4 images from 1 to 10.

The survey web-page was designed as follows (see Figure 2.3): After a brief introduction to the survey itself, each of the subjects was asked once if they considered themselves experts in the field of flow visualization and wanted to participate in the scientific visualization study, or if they preferred to judge the images purely by their artistic merit. This question was added because of the anonymous nature of the survey, since it was impossible to prevent people from forwarding the survey invitations to others.

In the actual survey, the subjects were presented with a set of four randomly chosen DVF visualizations of the same flow field. The task was to rate the images from 1 to 10 on either scientific or artistic merit. The subjects could evaluate as many sets of images as they liked. The hypothesis was that the metric, perhaps with some minor adjustment of parameter λ, would produce the ordering of images close to the one provided by the subjects based on their expertise in the domain of scientific visualization.

Survey Results. By the time of the evaluation, a total of 596 votes were recieved from 53 distinct users. To ensure that those are likely to be distinct persons the IP address and a tracking cookie were used. From these users, 37 designated themselves as experts in the field, and 16 chose to evaluate the images by their artistic merit. These answers, from each set of four images resulted in $\binom{4}{2} = 6$ orderings or, total of 3576 orderings. Along with these, the differences in rating were stored as well as a measurement of the user's confidence of this ordering.

To perform the training and verification of the metric, the stratified holdout method was used with a $\frac{2}{3}$ to $\frac{1}{3}$ training to test-set split. From the training set, a $\lambda = 0.26$ was learned for the metric, which resulted in 91% matches in the test. Since this was a relatively large test set we can assume that this score generalizes well. With a confidence level of 95%, a confidence interval of $[88\%, 93\%]$ was achieved, applying the scheme as proposed by Kohavi [54].

In the next step, considering all the images pairs where the metric ordering disagreed with the user's choice, two categories of such images were identified:

1. Images with metric values different by 5% or less. Those were mostly low-quality images, hardly recognizable as flow visualizations at all. In this case, the confidence value from the user was also low (rating difference of only one) and one can assume that the participants in most cases just saw these images as *bad images* and the classification was relatively arbitrary.

2. In disagreement with the metric's evaluation, low-contrast images were sometimes rated better than contrast-enhanced images with less coherence along the flow by some participants (see Figure 2.4). Upon further investigation, it was discovered that for these image pairs, about half of the participants' votes preferred the low-contrast images and one half that prefers high-contrast images.

   Consequently, it was impossible to adjust the parameter λ to get a perfect match for all votes. One reasonable explanation of this fact is that there are two types of people, one half that prefers the low-contrast images and the others. Therefore, next we split the votes into those two types. As a regularization, the voters that never encountered a set with low-contrast images were added to both groups. Within each group, the same training/test splits were used as detailed above. Then the parameter λ was optimized independently for both groups, resulting in a

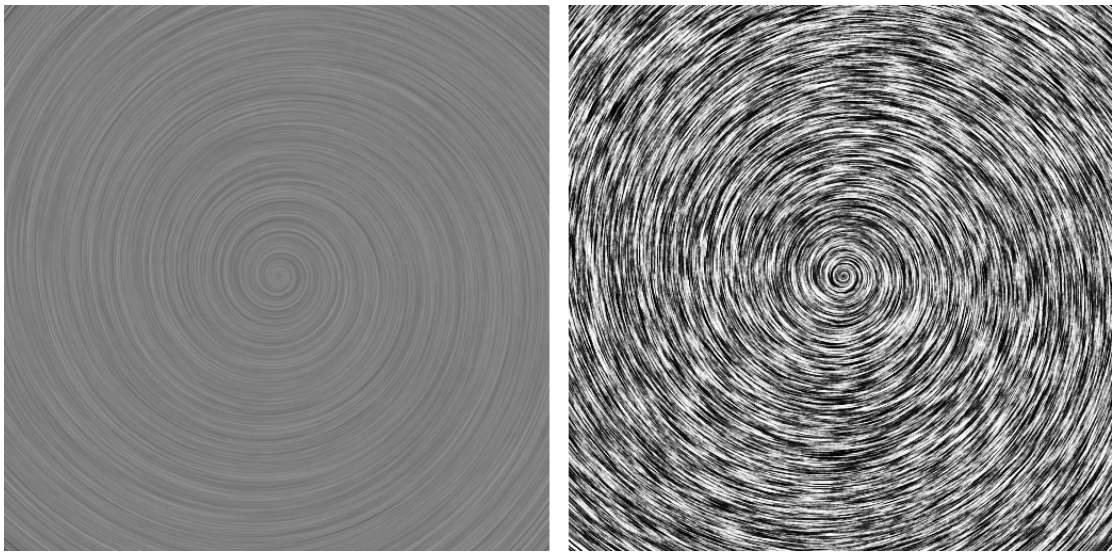

FIGURE 2.4: An example of the ambiguous low vs. high contrast case. If you prefer the left image (ordinary LIC) over the right (LIC with histogram equalization), then you should use a $\lambda = 0.1$ otherwise a $\lambda = 0.4$

$\lambda = 0.1$ for the "low-contrast group", which then matched 93% of those votes in the "low contrast and regularization group". For the "high contrast" group, a $\lambda = 0.4$ setting resulted in 98.9%. To solve this ambiguity in practice, one can use a small training set of one or two images pairs (e.g. Figure 2.4) to determine in which category a user falls. An application could then adjust λ accordingly.

This second observation has not been described in the literature before. As the survey was carried out anonymously, it cannot be concluded whether this difference results from cultural background or past experience, or if it is rooted in differences in eyesight, psychological reasons, or any of a multitude of causes.

2.2.4 A Different View on the Metric

In the first Equation 2.1 and the derived metric Equation 2.3, we considered two local quantities of the DVF image combined: the direction of the vector field and the gradient of the image. Now, let us formulate a metric, using the image gradient magnitude and the angle between the gradient and the vector field.

Assuming that the direction perceived from the image is orthogonal to the image gradient it makes sense to measure the angular error between these two directions. The image gradient magnitude, can be thought of as the confidence of the direction. If the gradient is low we will hardly be able to perceive the direction in the image but if it is high we would want it to indicate the right direction. This observation allows us to use the gradient magnitude in a weighting factor for the angular error, resulting in a following generic formulation of the metric.

$$M = -\frac{1}{|\Omega|} \sum_{\omega \in \Omega} W\left(|\nabla f_\omega|\right) \cdot K\left(\alpha_\omega\right) \tag{2.4}$$

where α_ω is the angle between $\nabla f_\omega^\perp$ and v_ω, K is the generic error factor, measuring the angular error and W is the generic confidence factor, depending on the image gradient. The minus sign in front of the sum denotes that the quality of the image is higher for the lower error.

The function W and K can be chosen with respect to the specific task. If one is interested in the overall contrast level which is clearly dependent on the gradient magnitude the simplest form for the W factor is $W(x) = x$. It also might be useful to normalize the weights over the whole image domain if one is not interested in distinguishing low and high contrast images. For the angular error factor K the simplest natural choice is $K(x) = |x|$. With these two definitions for W and K the metric in the Equation 2.4 becomes a weighted L_1 norm of the angular error in Equation 2.5 :

$$M = -\frac{1}{|\Omega|} \sum_{\omega \in \Omega} |\nabla f_\omega| \cdot |\alpha_\omega| \tag{2.5}$$

As the user study in Section 2.2.3 shows there might not be a unique metric suitable for all users. Consequently a parameter is required to adjust the metric to user preferences. For the metric formulation in Equation 2.4 the natural choice would be to use a function space of L_p norms with parameter p as in Equation 2.6:

$$M = -\frac{1}{|\Omega|} \sum_{\omega \in \Omega} |\nabla f_\omega| \cdot |\alpha_\omega|^p \tag{2.6}$$

Now, let us take a second look at the original simplified version of the metric with parameter λ in Equation 2.3 and bring it to the generic form, of equation Equation 2.4. With a few simple steps, the gradient magnitude and the angular error can be separated. To achieve that let's rewrite the dot products between the gradient vector and the vector field direction using cosines of the angle between the vector field and the gradient, obtaining the Equation 2.7:

$$M = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} (\lambda \cdot |\cos(\alpha_\omega)| - (1-\lambda) \cdot |\sin(\alpha_\omega)|) \cdot |\nabla f_\omega| \tag{2.7}$$

where α_ω is the angle between $\nabla f_\omega^\perp$ and v_ω.

In the Figure 2.5 the shape of some frequently used L_p norms is illustrated as well as the angle factor in the original formulation (Equation 2.7) with different λ. The important observation is that varying λ produces similar curves as commonly used norms, this similarity is also reflected in the quality of the metric, i.e. how well it can predict the users choices after the learning phase (in this case tweaking the exponent). Experiments indicate no significant improvements of the prediction quality when using Equation 2.6 as metric over the initial choice Equation 2.7.

Since there is no evidence that one particular type of K is better than the other it makes sense to take the computational efficiency into account. For performance reasons it is generally a good idea to avoid the computation of a integral over the whole image to obtain the metric value for every new K. The original formulation in Equation 2.3 allows the pre-computation of the sums over the image separately. This makes the metric M a linear function of the parameter λ, that can be easily computed for different λ.

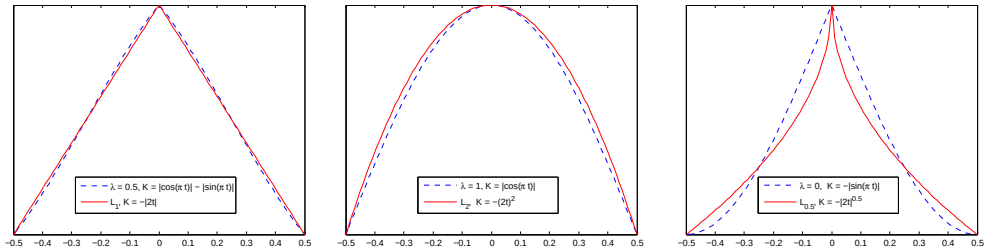

FIGURE 2.5: The shape of K factor in Equation 2.4 for functions producing common norms and for different values of λ in Equation 2.7.Note that the functions are biased and rescaled appropriately for display in this image.

2.3 Experiments and Application Scenarios

The metric, as trained by the user study, can then be used to compare two images of the same data set computed with different DVF methods or parameter sets. Experimental data suggest that absolute values of the metric can also be used to make a general statement about the image quality. Scaled the metric value with the factor 200 one can assume that images with a metric above 70 were generally considered high-quality. The scaling factor of 200 was chosen to get "nicer" values and is motivated by the observation that the best images that were produced in experiments often ranged around an unscaled metric value of roughly 0.5; therefore, the values are upscaled to get numbers that one can think of as a percentage, though values larger then 100% are possible.

This scale is used for all the graphs in this chapter. For the computation of all the graphs in this section we set λ to 0.26 but note that although the shape of the graphs changes, the conclusions are the same for the specific low and high contrast groups.

In the remainder of this section, a number of scenarios is presented where the metric is used not only to compare two images, but to look at more general issues. All of these experiments share the same idea that an automated system "looks" at a large set of DVF images and extracts images with interesting statistical properties (such as the best or worst) or considers the statistics itself.

2.3.1 The Per Pixel Metric Distribution

In this experiment, we are going to take a closer look at the distribution of the local metric values. In most other scenarios, only the average over the entire image is considered in order to compare different techniques. To further analyze a specific technique for a given data set, however, it may be interesting to visualize where it achieves a high score and where it fails. This could be used as a local confidence in the flow visualization or as a clue for how to improve a given visualization algorithm. Various figures in this section show a number of DVF visualizations side by side to the corresponding metric images. In these meta-visualizations in Figure 2.6a-d it can be seen how the metric captures problems in the LIC image. Visualizations in Figure 2.6a and 2.6b suffer from aliasing, appearing as red dot pattern in the metric image. This problem can be avoided with a two-fold LIC or a usage of band-limited noise (e.g. wavelet noise (Figure 2.6c and d)). The locality of the metric is an especially valuable feature in this

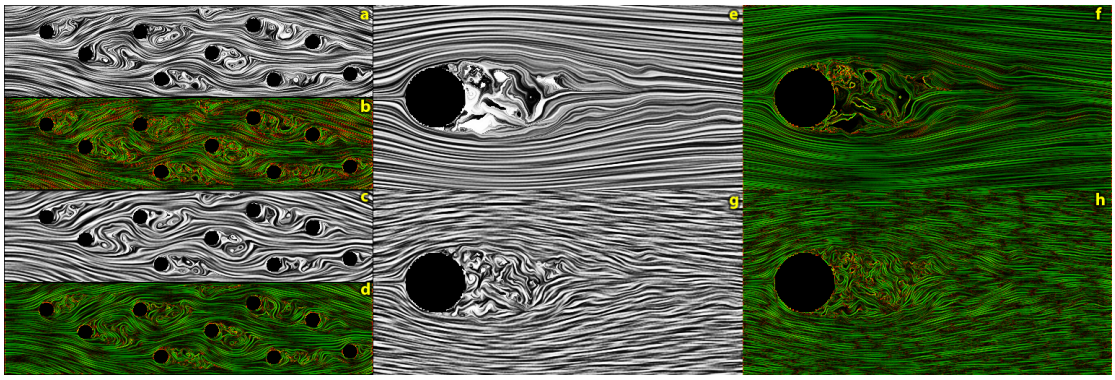

FIGURE 2.6: The images a-d on the left demonstrate how the metric nicely captures aliasing artifacts (in red). The four images e-h, in the center and on the right, show that a constant step-count LIC-method either captures laminar regions accurately (with a large kernel) or turbulent regions (with a smaller kernel)

case, since the aliasing doesn't have a great influence on the integral metric value if the percentage of aliased pixels in the image is small. On the other hand those pixels can be easily recognized, since they results in regions in the metric image having extremely low value. In Figure 2.6e-h, the influence of the number of integration steps on laminar and turbulent regions can be seen. In Figure 2.6e/f, 200 steps were used, compared to 60 in Figure 2.6g/h. Longer traces improve the metric in laminar regions, while shorter traces improve the quality in turbulent regions. This observation can be used to implement an optimized LIC algorithm, as described below.

2.3.2 Influence of the Random Seed

Most of the DVF visualization techniques rely on a noise field somewhere in the algorithm. While the types of noise vary from method to method, one input to the algorithms is a (pseudo) random number generator. Naturally, these generators produce different outputs depending on the random seed, which may be controllable by the algorithm or not. Either way, a different random seed will produce a different image and at a very fine scale this variation will influence the quality of the image. Considering this random seed influence a background noise in the visualization, one can quantify its impact on the image quality. To test this, the following experiment was conducted: 100 distinct white-noise textures were generated, and for each of them, a number of line integral convolution visualizations was computed with a varying kernel length l from 0 to 200.

Each l resulted in several sets of metric values depending on the noise texture. From these sets, the mean and variance of the metric value were computed before and after histogram equalization. The results for one flow field are shown in Figure 2.8. The behavior of the LIC images without equalization is shown in Figure 2.8a. We can see that starting from noise (LIC with zero steps) with increasing step count we observe a rapid improvement in metric values followed by a quick decline. This happens because the images converge to uniform grey. In the second graph in Figure 2.8b, we can see the influence of the random seed in the results. It is shown by the means of the variance and the difference between the lowest and highest values of the metric with varying

random seed for different kernel sizes. It is relatively small, even for small kernels, and decreases even more with increasing kernel length. This can be explained as the image converges to the mean noise value, which is a property of the noise itself that should be independent of the random seed.

In contrast to these results, the histogram-equalized LIC behaves differently. First, for this field the mean metric value does not have a maximum inside the tested range but seems to converge to a relatively high metric value (see Figure 2.8c). Continuing the tests with even larger kernels shows that for very large kernels this curve will eventually drop too, though much later. The more important observation, however, is that the metric variance is about an order of magnitude larger then before, and it increases (seemingly linearly) with the kernel size (see Figure 2.8d). This can be explained by the fact that the histogram equalization step, with increasing LIC kernel length, has to stretch smaller and smaller variations onto the full range of grey scales. Consequently, a growing influence of the random seed can be observed. This can lead to a difference between the best and worst image of up to 15%, which results in visible quality differences (shown in Figure 2.7).

To summarize it all, in most situations the random seed has little influence on the image quality, as expected, but if intermediate steps of the computation process are significantly upscaled by operations such as histogram equalization, then random seed can have a significant influence on the image quality. It is worth noting that while the mean-value graphs behaved differently for other vector fields, the variance and maximum variation characteristics stay the same for all tested vector fields.

2.3.3 Manifold LIC

One important aspect of LIC is the enhancements proposed by Okada and Lane [78]. While discussing a number of other LIC improvements, they proposed using a combination of LIC (*l*), high-pass filtering (*h*), and histogram equalization (*e*) to improve LIC image generation speed and image quality. They further suggest that the sequence *llhe* (meaning: first start with a LIC image, then use that image as noise input to a second LIC computation, next perform high-pass filtering, and finally do a histogram equalization) produces the best results. Next, they mention that they "found no appreciable difference in executing the LIC algorithm more than twice". With the metric the process of evaluating a large number of LIC and filter parameters and combination of these settings for double, triple, or even higher iterations can be automated.

Extensive parameter exploration by evaluating tens of thousands of combinations confirms that the *llhe* in fact results in the best metric value when only two LIC passes are performed. Moreover, it is possible to improve the quality by applying more operations. Tests of combinations of up to four LIC applications with and without interleaved high-pass filtering and/or histogram equalization, showed that, for example, a triple LIC sequence *lelhele* can produce visualizations with an about 15% higher metric value (see Figure 2.9).

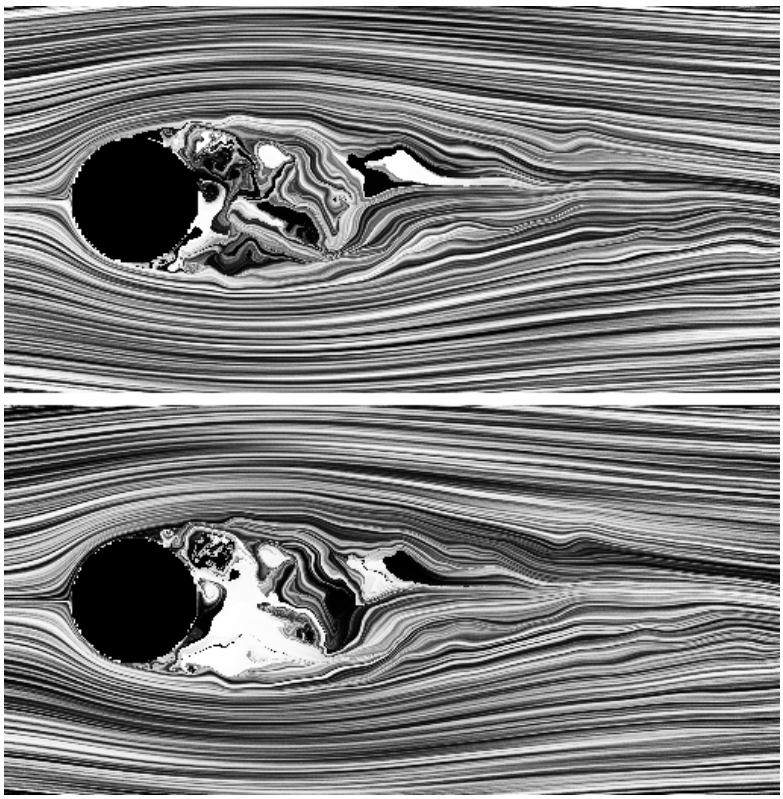

FIGURE 2.7: Two images with the same LIC parameters but different random seeds. Notice the quality differences especially in the turbulent region.

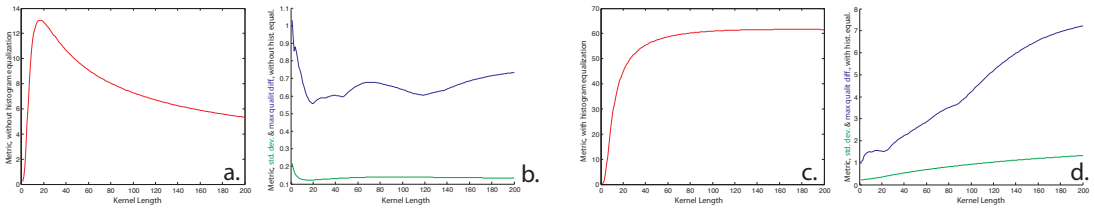

FIGURE 2.8: Graph a. shows the behavior of the metric value for the vector field shown in Figure 2.7 when only LIC without histogram equalization afterwards is performed. In graph b. the maximum quality difference in metric, as well as the standard deviation between different random for seeds in relation to the LIC kernel size, can be observed. The graphs on the right show the same information for histogram-equalized images.

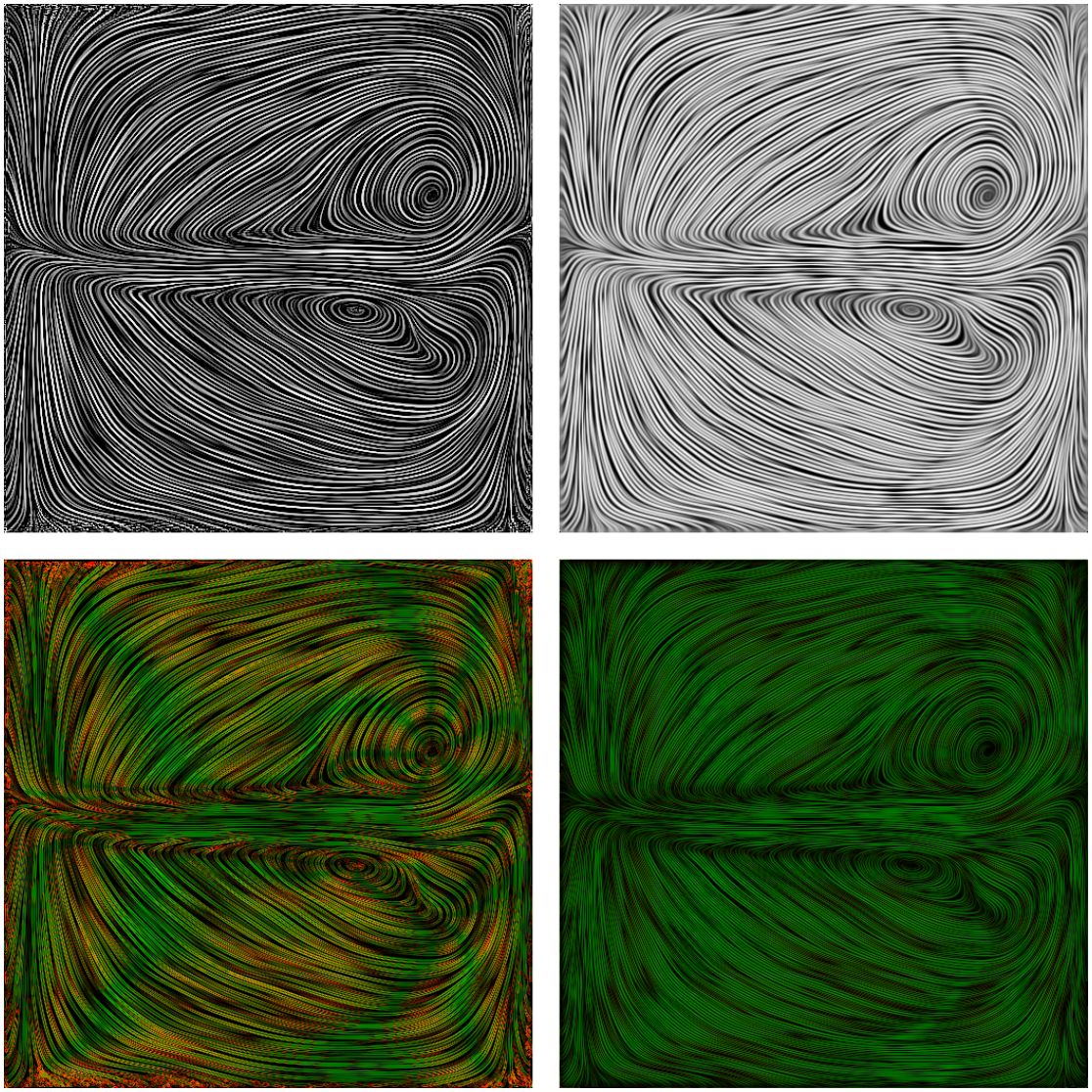

FIGURE 2.9: The image on the left was computed with the *llhe* sequence proposed by Okada and Lane [78]. The right image was computed using the sequence *lelhele*

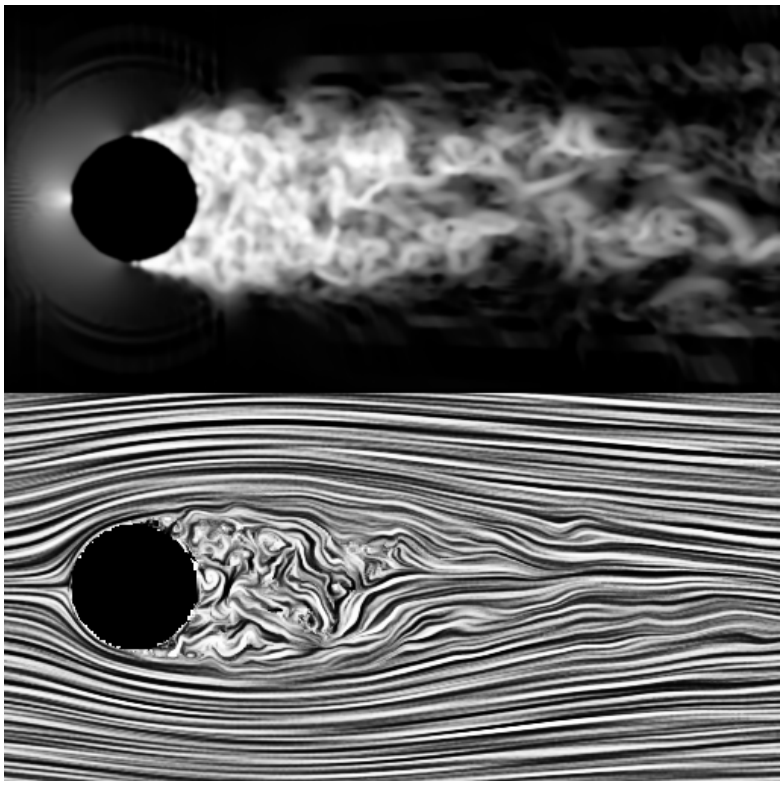

FIGURE 2.10: The top image represents the variance of the flow direction. In the bottom image, this weight texture is used to scale down the step-count and achieve a better overall visualization. Compare this image to the results in Figure 2.6.

2.3.4 An (Adaptive) Metric as Part of a DVF-Visualization Pipeline

An obvious use of the metric, apart from simply comparing two images, is to integrate it into a visualization pipeline, instead of choosing a fixed method and parameter set for all images, as it is usually done. An integrated metric can optimize these parameters for each image. Therefore, the visualization system would need to support a number of different DVF techniques and using parameter exploration would search for the optimal image. Intermediate best results could be presented to the user as the system optimizes. As an extension to this fully-automated system, a semi-automatic steering approach could further improve the image quality. In this extended version the user is given the ability to use a brush-like metaphor to indicate interesting regions in the intermediate result. Within the metric computation one can use this *focus image* as a scale for the metric. As a consequence, the metric will automatically steer the computation of the DVF visualization towards parameters that produce better results in those specific regions. While the basic concept of a user-defined or vector-field based scalar parameter has been proposed before [42, 51, 97, 129], the connection with an automated metric and a progressive interactive optimization is a novel concept.

If this focus and context visualization is not desirable, the findings of the optimization process can also be used to improve the DVF itself. From Figure 2.6, one can conclude that LIC produces better images if shorter kernels are used in turbulent regions and longer kernels are used in laminar parts of the domain. To verify this, the following

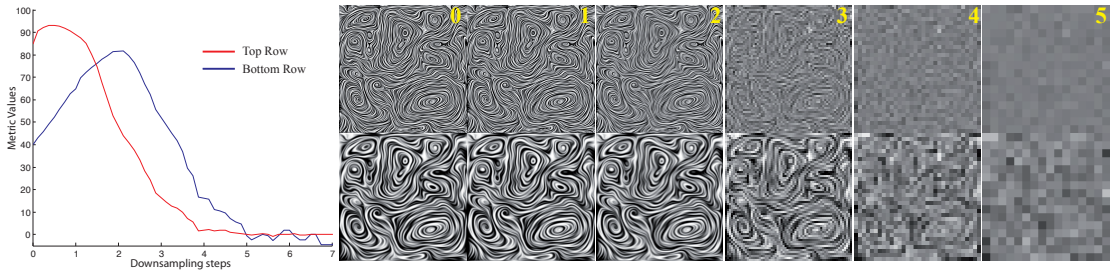

FIGURE 2.11: In the top row, a fine detail LIC image is down-sampled. The resulting sequence suffers from aliasing and a loss of contrast, consequently the metric value (red) degrades quickly. In the bottom row, a LIC image with coarser structures was generated. Its metric value (blue) first increases—the same information is conveyed with less space—until it also degrades, but still achieves better quality than the first image.

experiment was conducted: the local variance of the flow direction was conducted as an approximation of the turbulence and used to scale the LIC kernel. This results in a unique method that achieves a better image than with a uniform kernel (see Figure 2.10).

2.3.5 Image Robustness

In the final experiment, let us look at the changes in the metric if the DVF-visualization image undergoes image transformations. In this example, we do not transform the domain and then re-run the visualization algorithm, but compute the image once and then see what happens when it is transformed. Here, we specifically focus on image rescaling, but other operations such as quantization fall into the same category. For practical applications, it may be interesting to generate an image that is robust to such transformations, specifically if you do not have control over them. An example is the generation of a LIC image for usage on a web page or a newspaper. When asked to generate such a visualization, it may not be clear at what resolution this image will be integrated into the web page or at what resolution/quantization it is printed. For this example considering different LIC algorithms let us explore how their metric values behave under sub-sampling. For instance, as mentioned in [78], twofold LIC produces smooth solid stream lines representing the flow field on a coarse scale. The resulting visualization (Figure 2.11 bottom row) is thus robust against downsampling. While ordinary LIC images are much more sensitive to minification (Figure 2.11 top row), both these observations agree with the behavior of the metric.

2.4 Implementation

2.4.1 Requirements and Design Considerations

The experiments, described in the previous section, as well as the web-survey would be impossible to carry out without a versatile software infrastructure that fulfilled the research needs. The understanding of this fact lead to the development of several interconnected software components, communicating through the database back-end.

The research needs, forming the basis for the requirements to the developed software can be classified as functional and non-functional. Among the functional requirements is the ability to make computational experiments with existing flow visualization algorithms and to evaluate the resulting images. The natural way to specify an experiment is a short program or a script, which defines an execution scenario of the chosen technique. Moreover, the system should provide the ability to handle large inputs and outputs (tens of thousands of images at once). Consider for example Figure 2.8, which required to process 20000 images for one plot. While the execution time of the experiment was not an issue, the process had to be automated to save researcher's time .

In fact, the time constraint, dictated by the publication submission deadline, was the main non-functional requirement, so rapid programming and research were the primary objectives.

The economy of time of the researcher, conducting the experiments, and the developer, building the experimental framework, might be contradictory tasks, unless these are the two roles of one person (which was actually the case). This balance between the complexity of software and the time spent on experiments was shifted towards making a more capable software. Yet, the goal of the project was not to come up with highly reusable general-purpose software product, but to facilitate the achievement of the presented scientific results. This consideration affected the preference towards simplest solutions in the choice of implementation technologies.

Fault-tolerance was as well one of the guiding principles of the software design, as it is almost impossible to get everything right from the very start in the research project like this. It implies reproducibility of the experiments when errors in algorithm are found and fixed. Since the system ended up with half a million images, this could by no means be done without automation and recording of all the results and the metadata associated with them.

The software performance was not considered to be a significant factor, since the developed system was meant as a throw-away research prototype.

These requirements lead to the following design considerations. All the experiments were recorded in the relational database along with all information necessary to reproduce them. The computational modules for flow visualization and metric computations were implemented as independent from the rest of framework OpenGL Shaders. Experiments were implemented as independent scripts, run from console, and could be rerun with new version of computational module if necessary. Experiment reports were automatically generated from templates. The system services were stateless to simplify the resource management and increase scalability.

2.4.2 Scripting Environment

Now, let us look at the implementation details. The discussed framework evolved from a set of Perl scripts. This language is rarely used for large projects requiring long-term maintenance because of readability issues, but in small and medium scripting projects it—especially when used on advanced level—is indefensible for formulation of high-level tasks in a very laconic way. As a consequence, the whole project fits in 5000 lines of code, not including third-party packages. More importantly Perl has the oldest package repository [19] among commonly used scripting languages (Python, Ruby) with a huge

number of modules facilitating the solution of for virtually any problem ranging from natural language parsing to image processing.

The data was stored in an SQLite [102] database. SQLite is an embedded database management system (i.e, a library providing functions to access the database file) without client-server overhead but with all the essential features of relational database system, including SQL queries, indexes, triggers, primary and foreign keys. Requiring no installation, it allows quick deployment and replication over network with simple file copying. For the database persistent layer standard Perl database interface (DBI) was used along with Class::DBI [18] object-relational mapping module. The latter is highly flexible and adds no configuration overhead, since it is able to generate domain objects schema in runtime, based on the schema of the currently attached database, which is convenient under the circumstances of constantly evolving database schema.

Most of the vector fields and noise textures were created in third-party software and imported from files into the database. Several independent converters from different file formats were written to the unified BLOB records in the database. The images, were not stored in the database, but as files in PNG format in a file system folder. As a consequence, the database file was kept small and the images could be directly referenced from the web interface. To keep track of the generated images they were assigned unique names, formed from the MD5 hashes of their content. These hashed were stored in the database and associated with all the computational steps used to generate the image, including algorithm parameters and a shader version.

The idea to record the experiments in the database and make them repeatable gave the opportunity for easy recovery in case when the previous results had been compromised due to an error in flow visualization algorithm implementation. For instance, in one the late stages of research the bug was found in the metric computation, affecting the resulting images with aspect ratio not equal to one. The invalid results were immediately eliminated from the database with a trivial SQL query and excluded from the statistics computation. Later, the experiments were recomputed using exactly the same scenarios as before.

A great deal of effort has been made to make the system stateless and loosely coupled. These design decisions serve the purpose of simplicity, tolerance to error and fast development.

Every image computation was implemented as a unique script, running an OpenGL shader or/and using ImageMagick library for image processing. Plots were generated with Matlab scripts. The typical experiment scenario involved the launch of a number of such computational processes from the command line with certain parameters, using the database as a storage for input and output. This simple modular scheme allowed to avoid efforts for careful control of resources that would be necessary in a complicated monolithic concurrent system, thus inevitably increasing the design and debugging time. It of course can cause a decrease of performance compared to the more obvious solution where all the computations for the experiments are done in one process, mainly because there is no memory caching of the large inputs. This drawback was to a large extent compensated by locating the database on a Solid State Drive, thus increasing the speed of memory access without the loss of modularity.

In this stateless scheme, execution of simultaneous experiments on several machines over network was a trivial task. Different database files, used on different machines,

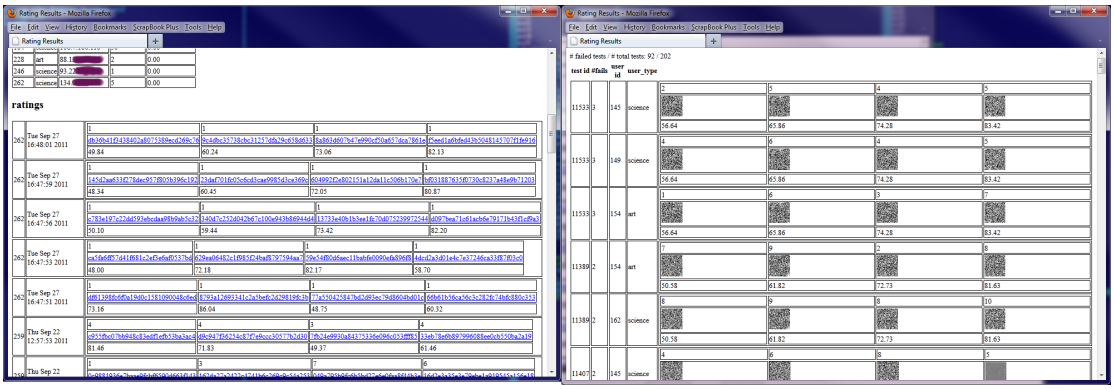

FIGURE 2.12: The administrative interface to the expert survey, featuring real-time statistics monitoring.

were then combined with a simple script for table merging. To enable interruption of the time-consuming experiment and its continuation from the previous step, the progress of current running jobs was tracked in the database. Such persistence of tasks made possible to schedule several experiments to be run overnight.

The current progress in the research was documented in numerous human-readable reports in portable document format (PDF). Report generation was automated with Perl scripts, which allowed to reuse report templates for similar data or to regenerate them if the experimental data appeared to be incorrect. Templates were described with Template Toolkit. [114] This utility features a mark-up language for text files and a template processor for Perl and Python languages. The templates were filled with the data from database and references to plots, resulting in TeX files, that were than the source of PDF reports. To facilitate image manipulation, several SQL extension functions, written in Perl, were hooked into the database access driver. This allowed for example to select images into the report directory with a SQL statement embedded into the template, or export a query result Matlab-readable format for plot generation. This template approach was used in the web-interface as well.

2.4.3 Web Interface

The decision to set up the database from the very start made it possible to handle a large number of images, tracking the visualization parameters and gathering statistics. The next logical step was to set up a quick-and-dirty web interface to the image catalog and a tiny, but carefully designed user interface of the expert survey.

The web pages were generated with a Perl script executed in the Apache web-server environment and delivered through traditional Common Gateway Interface (CGI). Session mechanism along with IP address tracking was used to identify users in the expert survey with an administrative interface (see Figure 2.12) available for the review of results.

Unlike more popular PHP language, where server-side code can be embedded into web page and mixed with HTML, scripting languages like Perl, Python and Ruby enforce a better design approach of separating logics and presentation. The pages were generated from templates filled by the scripts with the database content. The implementation of the web front-end consists of several HTML page templates with intense use of CSS,

and JavaScript. The user interface (UI) was based on JQuery framework [49], which allowed to add effortlessly the following features to the web-page, otherwise non-trivial to express with standard HTML means: slider controls, modal dialogs, window fading effects. The web-services were tested and known to run identically in the latest versions of Google Chrome, Mozilla Firefox, Internet Explorer and Safari browsers (including mobile devices).

Summarizing the features of the developed environment, it is worth mentioning the most important:

- numerous modules of flow visualization;
- quality metric computation;
- interface for image evaluation in the expert survey;
- database persistence;
- basic image processing;
- console interface for experiment launches;
- batch image computation jobs planning and execution;
- parsing and import of vector fields;
- automated report generation from templates;
- plot generation and analysis of statistics;
- interface for the real-time survey statistics monitoring;
- a hierarchical catalogue of computed images ;
- search of images matching certain criteria.

2.5 Conclusion

In this chapter a novel metric to evaluate the quality of DVF visualizations is proposed and the results of a web-based user study are presented. The survey was conducted to verify the applicability of the metric and to fine-tune the metric parameter. In the course of this optimization, two distinct groups of users were discovered. To be able to correctly rate the images for both groups, two parameter settings are proposed as well as a simple—one image pair—test to determine which group a person belongs to. With this distinction, the metric agrees with up to 99% of all user evaluations, and it allows to use it in an automated process and draw conclusions in a number of experiments in which existing methods are compared, previous findings are validated, and visualizations are optimized.

For the survey and experiments, a number of different DVF techniques was implemented, and in the course of those implementations, naturally a significant amount

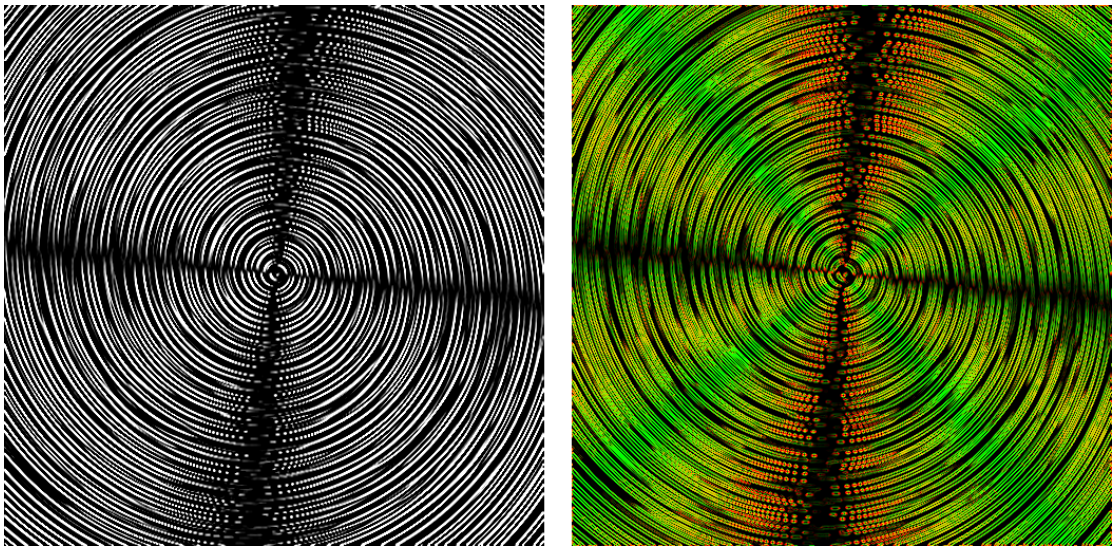

FIGURE 2.13: The left image shows a visualization of a spiral flow generated with the manifold LIC method and uses a poorly implemented high-pass filter in the process. As can be seen on the right, while the metric picks up some of the distortions, the zero-metric areas in black do not significantly contribute to the average value to degrade this image's value.

of time was spent debugging. Thereby, the metric proved to be also an excellent verification/debugging tool as in most cases issues with a new method this resulted in significantly lower metric values then for previous methods. So far, we encountered only one exception, which was an issue related to a high-pass filter that we applied during a multi-pass LIC computation; this example demonstrates a limitation of the method. For the visualization shown in Figure 2.13, we computed a high metric value. Further investigation revealed that the problem in this image was that the majority of pixels achieved high metric values and black regions do not significantly reduce the average. In fact, if these regions would be distributed differently, we would probably rightly consider this image of good quality. The problem is a global phenomenon—namely, that the black cells cluster in an arbitrary pattern and cause disturbing structures. To tackle this issue, a hierarchical application of the metric is considered, similar to the idea of Jänicke and Chen [42] for the salience-based metric.

Chapter 3

Tangent Vector Fields: Texture-Based Visualization Using Isocontours

In this chapter a framework is proposed for visualization of tangent and gradient fields induced by a scalar potential. Rather than following the traditional approaches to vector field visualizations we utilize the isocontours of the underlying scalar field, that are also the integral curves of the tangent vector field. That is, given a vector field $\bar{u}^{\perp}$: orthogonal to a gradient field $\bar{u}$ of a scalar field f, i.e., $\bar{u}(x) = \nabla f(x)$, we visualize the level sets S_c, defined as $S_c = \{x : f(x) = c\}$.

Visual representations of isocontours, the connected components of the level set, have been successfully applied for years as a useful means for visualization and analysis of scalar fields. Our objective is to give the viewer an overview of the entire range of isocontours, and thus the integral curves of $\bar{u}^{\perp}$ similar to dense flow visualization approaches such as the Line Integral Convolution (LIC). In order to achieve this, we introduce a novel technique that is capable of displaying of as many isocontours as possible at the given view scale without introducing any aliasing. The cornerstone of the discussed system is a view dependent periodic transfer function with the period *depending* on the gradient magnitude of the underlying scalar function such as to create a dense visualization *independent* of the gradient magnitude. We demonstrate that the presented approach is easy to implement, computationally efficient, and suitable for the fields that have reasonably structured level sets, i.e. are sufficiently smooth.

3.1 Isocontours For Dense Vector Field Visualization

Level set representations commonly referred to as *isosurfaces* for 3D- and *isolines* for 2D-fields have a long standing history in visualization. They are mostly useful in the applications, where few scalar values, corresponding to static isosurfaces, are of interest, such as medical CT images that have well defined skin and bone surfaces. In many other scenarios it is required to track the evolution of the scalar field through its isosurface, many simulation for instance datasets exhibit this property In this case, a manual exploration of the values is necessary, which might become a tedious task. An alternative approach would be first generate an overview image of all of the isocontours similar to the way the dense methods work for flow visualization and next allow the user to selectively show detailed information of specific isosurfaces. To the best of our knowledge, such an approach of interactively presenting a large number of isocontours simultaneously has not been presented in the literature. Therefore, we demonstrate a

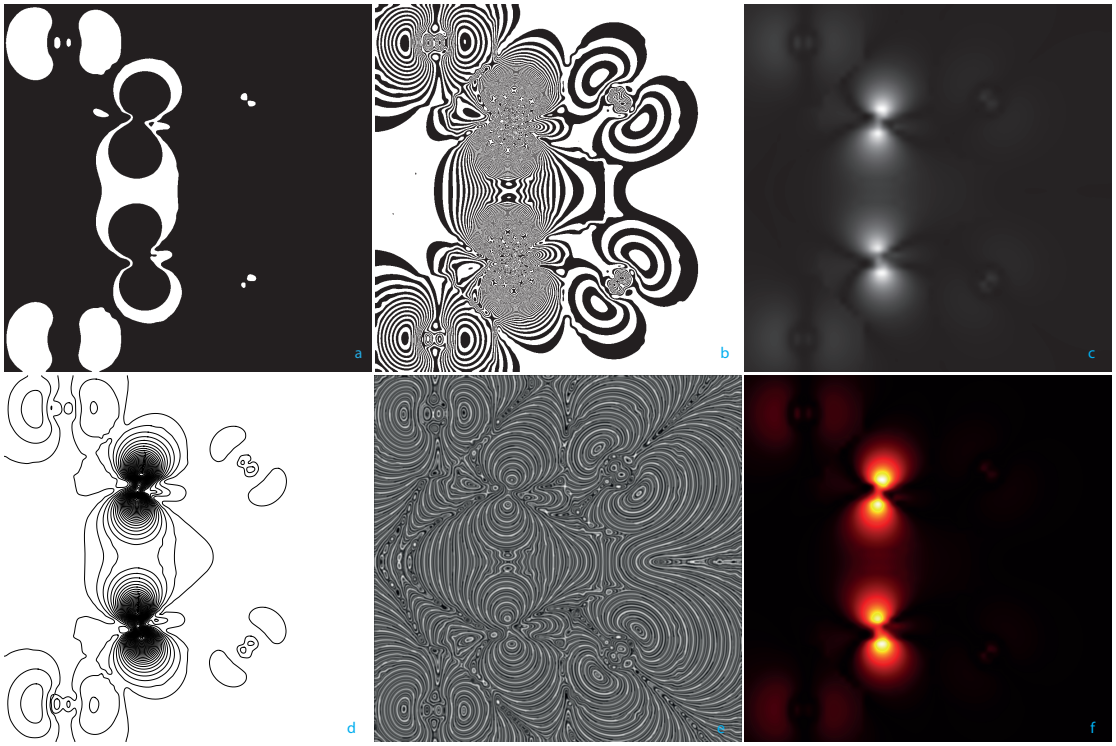

FIGURE 3.1: Different transfer functions and marching cubes algorithm (d) applied to a slice of the hipip dataset. a) box function, b) sum of box functions, c) identity function, d) geometry (vector graphics), extracted with marching cubes for isovalues from 0 to 1 with step 0.03, e) our transfer function f) identity function maped to a heat color map

novel technique that provides such a visualization concept based on ideas of dense flow visualization transferred to the visualization of scalar data.

Dense (texture-based) flow visualization has proven itself in many applications to be an efficient way to provide a big picture of the underlying data. It is the technique of choice whenever one intends to make the most use of the available screen space. Moreover it allows to avoid missing important details, as little input data filtering is involved and it is highly suitable for parallel computation. However, this approach represents obvious visual cluttering challenges in 3D, is not applicable directly, so we suggest to use dense visualization of sections of the original volume with some clip geometry (e.g., clip planes). The motivating idea is to provide an overview of the isocontour pattern on top of the rendered volume and to allow direct interaction with it using isocontour slices as a means of the isosurface selection which is a desirable mode of operation in many applications. In our 3D application we demonstrate how it is beneficial to equip traditional isosurface extraction and direct volume rendering methods with a convenient interface providing the overview of the isosurfaces for the whole dataset and allowing to choose an interesting value for detailed examination. To allow interactive exploration by relocating the sections, the method needs to be computationally efficient, which is actually the case, since it is based on basic transfer function computation. That is, striving to make it possible to explore multiple isocontours at once, not just one static isosurface, we propose a novel visualization technique based on a periodic transfer

function. It brings the advantages of dense visualization into the 2D and 3D isocontour visualization domain, combing the best of both worlds.

Specifically, the contribution of this chapter can be summarized as follows:

- we combine dense visualization of flow fields and scalar fields;
- we present a simple to use yet effective technique based on a specially-designed transfer function;
- we demonstrate the applications of the presented approach to 3D and 2D tangential fields.

The remainder of the chapter is structured as follows: we first review the state of the art isocontours visualization and dense flow visualization methods, considering how these traditional techniques can be applied to our problem. Then, we discuss the benefits and challenges of a transfer function approach. In Section 3.5 we give a detailed description of the method and demonstrate it in two applications we have implemented. We conclude with a discussion of the results.

3.2 General Methods of Isocontour Visualization

There exists a vast amount of literature on the isosurface extraction and rendering methods, indicating the high interest of the scientific community to this problem. Here we briefly review selected fundamental approaches in the area.

The grounds of isosurface extraction were laid by the well-known marching cubes algorithm [65], which processes the volume data to form a mesh corresponding to an isosurface. It is probably still the most widely applied method, despite several alternative and improved techniques such as marching tetrahedra [117] and particle-based surface extraction [21] being out there for decades.

Almost at the same time as geometry extraction, the approach to isosurface visualization via direct volume rendering with the aid of ray tracing and transfer functions were explored by Levoy [62]. In this area most of the work addresses the challenge of the transfer function design and extraction of meaningful isosurfaces discussed below, with notable exceptions including the scale-invariant volume rendering approach of Kraus [55], which achieves the effect of uniform opacity by computing the volume rendering integral in the data space instead of the object space.

The vast majority of the available publications are devoted to extension of the traditional techniques for performance improvement. There is much progress in increasing the run-time efficiency by employing data structures such as octrees [99] and interval trees [17]. Aiming to avoid speed-accuracy trade-off, numerous view-dependent rendering methods and adaptive tessellation techniques have been proposed [2] [32] [64]. A lot of effort has been invested to develop efficient implementations [15], using parallel processing [35], [29], [138] and modern GPU programming [8], [112], [6].

At the same time there has been a great deal of research on the selection of the representative isosurfaces, having the goal to cope with visual cluttering in the 3D visualizations. There are generally two approaches: automatic selection of the interesting surfaces and the user-operated control.

The first one consists mainly of value selection and transfer function design for highlighting the surfaces and connected components, representative according to some specific criteria.

Pekar et al. suggested [81] to use regions of the steepest gradient as an indicator of meaningful surface boundaries. Tenginakai et al. [115] used histograms of gradient to detect salient isosurfaces. Topology has been actively used as a tool to decompose the volume into simple structures, allowing to emphasize the variation of isosurface shape with color and opacity. Contour tree computation algorithm introduced by Van Kreveld et al. [56] found its extensions and applications in the works of Takahashi et al. [109] [110] and Zhou et al. [139]. The importance of critical values on the topology of isosurfaces was reconginzed and utilized by Weber et al. [124].

There is a much smaller variety of methods proposed with a user-centric approach to isosurface selection. The notion of contour spectrum, introduced by of Bajaj et al. [1], was developed to guide the user in the isovalue selection, presenting aggregated characteristics such as surface area and enclosed volume of the isosurface. Additionally the implementation displayed the contour tree, which was turned into fully functional widget for isosurface selection by Carr et al. [10] in his work, introducing flexible isosurfaces. Alternative approach was taken by Kniss et al. [52], who introduced the means for interactive design of a multidimensional transfer function, and specifically the transfer function depending on the value and the gradient magnitude of the volume.

We are considering a new type of interactive visualization technique, allowing the user to pick the isosurface of interest while using the geometry of a representative subset of existing isosurfaces as a guidance. Having such visual clues of the big picture of the isosurface evolution, if it doesn't lead to visual cluttering, might be beneficial in providing an intuitive way to explore the surfaces of interest. To the best of our knowledge no methods of this kind are represented in literature.

3.3 Vector Field Visualization on Surfaces For Tangential Flow

One viable approach to imaging isolinies on a clip surface is to apply methods of dense flow visualization on a surface. Considering multiple isoline extraction as a flow visualization task for the vector field tangential to the isolines, we can utilize a variety of techniques, available for visualization of flow on surfaces, including but not limited to curvilinear LIC [28], dye advection [100] or image based flow visualization for surfaces [133]. However, the downside of this approach is that discarding the original scalar value results in the necessity to integrate the tangent field, while traditional isosurface extraction approaches, give us way more efficient methods to trace an isosurface or isoline. Fortunately, as we show below, a simpler, more visually pleasing and much more computationally efficient solution is possible for scalar fields.

Another option for the visualization of multiple isocontours is to involve the geometric methods of flow visualization. The idea here is to combine the speed of isosurface and isoline extraction algorithms (compared to streamline integration) and the simplicity of streamline visualization. However, the problem of optimal dense streamline placement, being actively research in flow visualization community, turns out to be non-trivial one.

Two common approaches to it that try to find a trade-off between visual cluttering and the coverage of the field, are the feature-based and the density-based solutions.

The first one, proposed by Verma et al. [121], highlights certain regions of interest, usually selected by topological considerations. In their work, they provide templates for seeding streamlines around different types of critical points, which have to be localized and classified beforehand. More advanced and complex approaches in the topology-based set of methods can be found in recent works by Liu et al. [63] and Wu et al. [135]. It is important to note that the segmentation of the field into regions of interest in advance, which is application-dependent as pointed out by Chen et al. [14], limits the applicability of this approach to our problem, since we try to avoid intelligent line selection.

Density-based approaches seed streamlines by managing the placement density throughout the whole image. High-quality results are achieved with the method of Turk and Banks [119], who proposed to use a low-pass filtered image as a measure of density and a randomized optimization process to achieve on average the predefined density value. Less computationally expensive techniques include the work of Jobard et al. [45], which separates streamlines by estimating the placement density as Euclidian distance between them and Chen et al. [14] who define a streamline similarity distance to place lines.

Most of the discussed optimal line seeding algorithms mostly governed by the following principle: new lines are inserted in the sparse regions and one of two lines should be terminate at the points where they are close to each other. This approach requires to process each line individually and to keep track of the distance between lines, while with the discussed method we are able to compute each pixel in parallel, using the underlying scalar field value to avoid measuring the distance between isolines.

3.4 Isocontouring Transfer Function

Applying a transfer function to the scalar data is the most intuitive way to map it to a visual representation, so it is natural to apply it to the isocontour visualization. Moreover, a spatially-varying transfer function can easily incorporate local settings, specific to scalar field behavior and view characteristics in a certain region. For instance, one important fact inherent to isocontour visualization is that it is generally only possible to approximate the true isocontours up to a certain resolution due to their fractal structure. See for example the work of Khoury and Wenger [50] for the thorough analysis of fractal the dimension of the isosurfaces. This resolution dependency becomes an even more important factor when perspective projection on the screen is used, which is usual for most 3D applications. In this case since different pixels on the screen correspond to different level-of-detail, some segments of the isolines should necessarily have coarser representation than the others. Consequently, the upper bound on the sampling frequency of the isolines in the screen space for the given view is space varying, and the transfer function should account for it. See Figure 3.2 for the illustration of variable level of detail.

While the highest representable frequency is limited as discussed above, visualization of the isolines with low frequency transfer function on the other hand is non-informative. One issue with traditional "intuitive" transfer functions is the need to compress of high

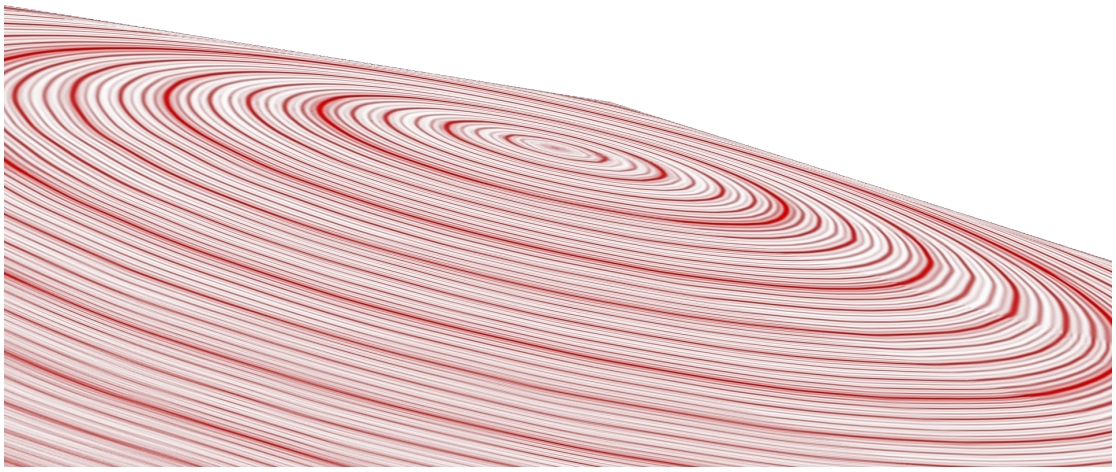

FIGURE 3.2: Red concentric circles represent isolines of the distance field to the center of the plane in 3D. Note that due to the perspective projection higher level of detail and thus the higher density of lines arises in the front.

contrast images for display on low dynamic range devices. It is closely related to one of the central problems of the extensively researched area of high dynamic range imaging (HDRI),.

In fact, while the HDRI techniques are commonly applied to real-world photo images which have natural restriction on a range of useful values and scales, the image sources in the scientific visualization domain, such as simulation and various acquisition techniques have a larger variety of characteristics, making the range compression issue more prominent. In the discussed technique we exploit a commonly recognized assumption [24] that the human visual system is much more sensitive to local intensity ratio changes, corresponding to high spatial frequencies, than to global intensity differences, which is the basis of many tone mapping algorithms. This observation, in contrast to upper frequency bound, discussed above, suggests that high-frequent sampling of the lines is preferable to perception. Also, unlike tone mapping our problem allows to add high frequent details to the resulting image in order to enhance the geometry of the visualized lines, even if they aren't present in the original data.

Our objective is, thus, to find for a view-dependent, frequency-bounded, locally adaptive transfer function with the following desirable properties

- a uniform filling of the screen space with isolines;
- as high as possible sampling rate of the isolines.

3.5 Method Design

3.5.1 The Basics

Now, we construct an appropriate transfer function, first presenting the general formal assumptions, our method is based on. We first focus on the 2D case, when the isocontours

are represented by lines (e.g. a section of the volume with a plane or surface), with a straightforward generalization to 3D (isosurfaces).

Consider a function $v(\bar{x})$ defined on a 2D domain Ω, which might be a section of a volume, assuming that the function is normalized such that $0 \leq v(\bar{x}) \leq 1$ and the gradient ∇v is bounded. The further restrictions on the function are introduced when necessary.

As discussed above, the crucial aspects of the visualization are the view resolution and the level of detail for each screen pixel. So we exploit the fact that is only useful to display the isoline up to a certain change threshold in our definition (Equation 3.1). We name a set of points $S_{v_0,\varepsilon}$ an isosurface (isoline) of the scalar field $v : \Omega \to \mathbb{R}$ for a value v_0 up to the error ε if

$$S_{v_0} = \{\bar{x} \in \Omega : |v(\bar{x}) - v_0| < \varepsilon(\bar{x})\} \tag{3.1}$$

where the domain $\Omega \subset \mathbb{R}^n$ and n is typically equal to 3, or 2. The error is generally specific to each point and depends on the gradient magnitude $|\nabla v|$ and the level of detail of the rendered line.

3.5.2 Isoline-Generating Transfer Candidate Functions

Given Definition 3.1, a natural way to design a transfer function $I_{v_0,\varepsilon}(v)$ for visualization of an isosurface $S_{v_0,\varepsilon}$ is to use a box function (see Figure 3.3)

$$I_{v_0,\varepsilon}(v) = \delta_\varepsilon(v - v_0) \tag{3.2}$$

where $\delta_\varepsilon(t) = 1$ for $|t| < \varepsilon$ and $\delta_\varepsilon(t) = 0$ otherwise.

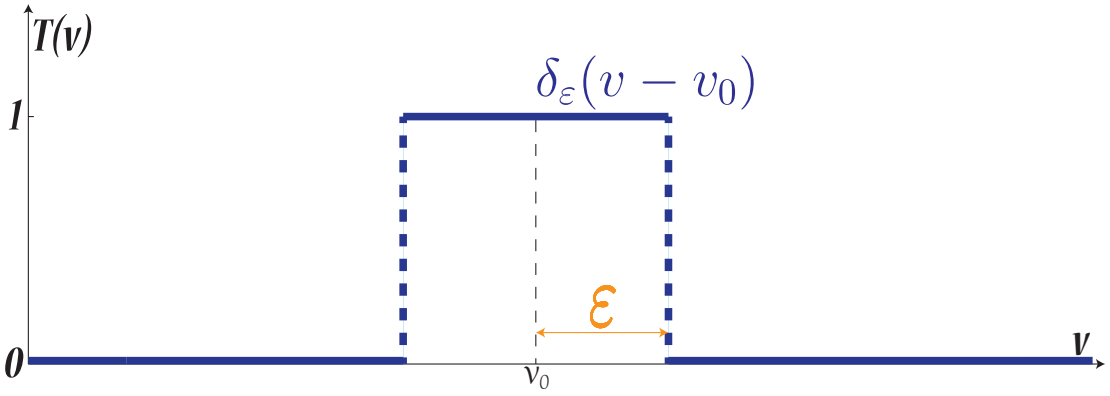

FIGURE 3.3: Box transfer function.

For visualization of several isosurfaces with uniform sampling distance h and constant level of detail ε, one can use the sum of functions, representing individual isolines as in Equation 3.3 and Figure 3.4 and Figure 3.1.a

$$SI_{v_0,\varepsilon,h}(v) = \sum_i I_{v_0+ih,\varepsilon}(v) \tag{3.3}$$

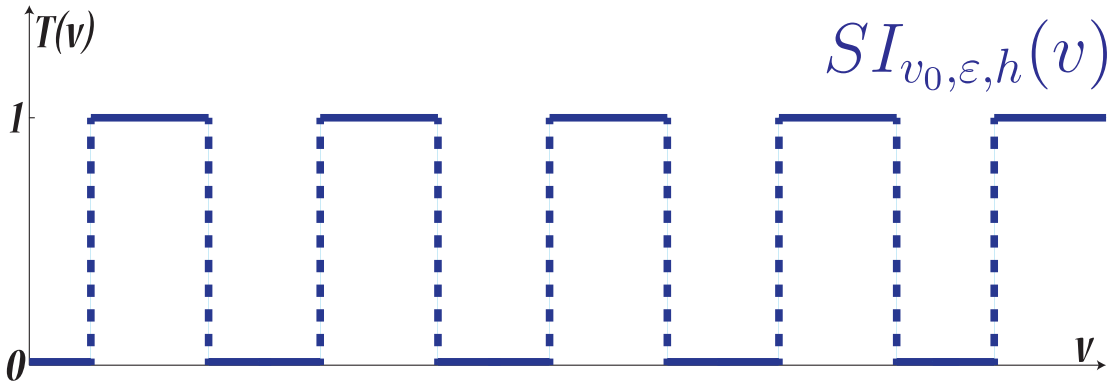

FIGURE 3.4: Sum of box transfer functions.

This function is periodic in v with period h and for our purposes can be replaced with a cosine as in Equation 3.4

$$T(v) = \cos(2\pi v) \tag{3.4}$$

corresponding to $\varepsilon = \frac{h}{2} = \pi$. Obviously, neither $I_{v_0,\varepsilon}(v)$ nor $SI_{v_0,\varepsilon,h}(v)$ provide a satisfactory isocontour visualization for most of the scalar fields. See for example Figure 3.1.a, Figure 3.1.b The approach we take, explained in the next sections consist in the generalization of definition in Equation 3.4 for the non-uniform sampling and varying level-of-detail. However, first, we simplify the problem further, reducing it to one dimension.

3.5.3 One-Dimensional Setting

For the proper visualization of the isolines essentially the perception of the direction of the normals to lines plays the crucial role. This information is transferred by the direction of gradient ∇T of the visualization image T, colinear to the gradient of the original data ∇v. This is confirmed for example by some flow visualization quality evaluation methods [69]. The orientation of ∇v and the magnitude $|\nabla v|$ are irrelevant and can be discarded. Since $\nabla T = T'(v)\nabla v$ virtually any function $T(v)$ is suitable for representation of isolines in the aforementioned sense. The only points where $T(v)$ fails to transfer the direction of the original gradient are those where $T'(v) = 0$. Additionally a special care should be taken of the magnitude $|\nabla T|$, which is related to the contrast of the resulting image, and the resulting spatial frequency.

We exploit the fact that the usefulness of the transfer function $T(v)$ to isoline visualization is mainly determined by the action along the gradient in our further analysis. For the sake of simplicity we will first consider the one-dimensional signal $u(t) = v(\bar{x}(t))$, where $\frac{d\bar{x}}{dt} = \frac{\nabla v}{|\nabla v|}$, i.e., the scalar field sliced along the gradient field. Here we assume that the critical points where $|\nabla v| = 0$ are excluded from the domain of the function, or handled separately. Trivial substitution shows that this setup results in $u'(t) = |\nabla v|$.

3.5.4 Piecewise Instantaneous Frequency Normalization

Motivated by definition 3.4 we are going to use the notion of instantaneous frequency to look for a transfer function $T(u,u')$ in the form $T(u,u') = \cos(2\pi u g(u'))$ where the choice of function $g(u)$ is to be clarified.

The instantaneous frequency $f(t)$ for the signal $w(t) = \cos(2\pi\phi(t))$ is defined (see Boashash [4] for an overview) as the derivative of the instantaneous phase $f(t) = \frac{d}{dt}\phi(t)$.

Our goal is now to design a band-limited transfer function

$$T(u,u') = \cos(2\pi u g(u')) \tag{3.5}$$

with instantaneous frequency:

$$\theta(t) = \frac{d}{dt}\left(u(t)g(u'(t))\right) = u'(t)g(u'(t)) + u(t)g'(u'(t))u''(t) \tag{3.6}$$

In addition to that the frequency $\theta(t)$ should be bounded from above by the Nyquist limit [33] defined by the current view resolution.

First, consider the sketch of the instantaneous frequency $u'(t)$ of a simple transfer function $T(u) = \cos(2\pi u)$ in time-frequency plane, shown in Figure 3.5.a We use a logarithm scale with base 2 for the vertical axis, thus multiplication and division by 2 corresponds to a translation by 1 unit, assuming for simplicity that 2 is the sampling frequency.

For each segment σ_i such that the instantaneous frequency changes by one unit on a log scale, we define a piecewise continuous transfer function $T_i(u)$ with Equation 3.7:

$$T_i(u) = \cos\left(\frac{2\pi}{\lambda}2^{-b_i}u\right) \tag{3.7}$$

where $b_i - 1 \leq \log_2(u') \leq b_i \ \ \forall t \in \sigma_i$ and $\frac{\lambda}{2}$ is a user-defined parameter, corresponding to the spacing between isolines.

As Figure 3.5 illustrates, this corresponds to cutting the plot into pieces of unit height, and shifting each piece to the desired frequency band (see Figure 3.5.b) The desired frequency band of the transfer function is bounded by the Nyquist frequency and the lowest acceptable frequency. Intuitively, we would like to keep it as narrow as possible, but at the same time one should keep in mind that the lengths of the segments σ_i should be larger than the wavelength λ to achieve full-period osculation at the given frequency, so we have to impose a restriction on the growth of function $\theta(t) = \log_2 u'(t)$. Our method is applicable to the functions, satisfying the Lipschitz condition of order 1. That is, there exists a constant M such that for any pair of points t and t' Equation 3.8 holds:

$$|\theta(t) - \theta(t')| \leq M|t - t'| \tag{3.8}$$

Given that $\max\limits_{t,t' \in \sigma_i} |\theta(t) - \theta(t')| = 1$, the upper bound for the choice of λ can be deduced easily: $|t - t'| \geq \frac{1}{M} \geq \lambda$. From below λ is only bounded by the sampling frequency.

We now consider the implications of our method in 2D, illustrated by Figure 3.7. The definition of the transfer function in Equation 3.9 is completely analogous to that in

Equation 3.7:

$$T_i(v) = \cos\left(\frac{2\pi}{\lambda} 2^{-b_i} v\right) \tag{3.9}$$

The regions σ_i in 2D can be defined as regions where $b_i = \lceil \log_2 |\nabla v| \rceil$ is constant. Their boundaries are prominent in the Figure 3.7.a. Note, that under assumption of continuity of the gradient ∇v for any adjacent regions σ_i and σ_j it holds that $|b_i - b_j| = 1$. The derivative along the gradient is just the derivative of the one-dimensional function $T_i(u)$, while the derivative in orthogonal direction is zero inside the regions σ_i. The problem

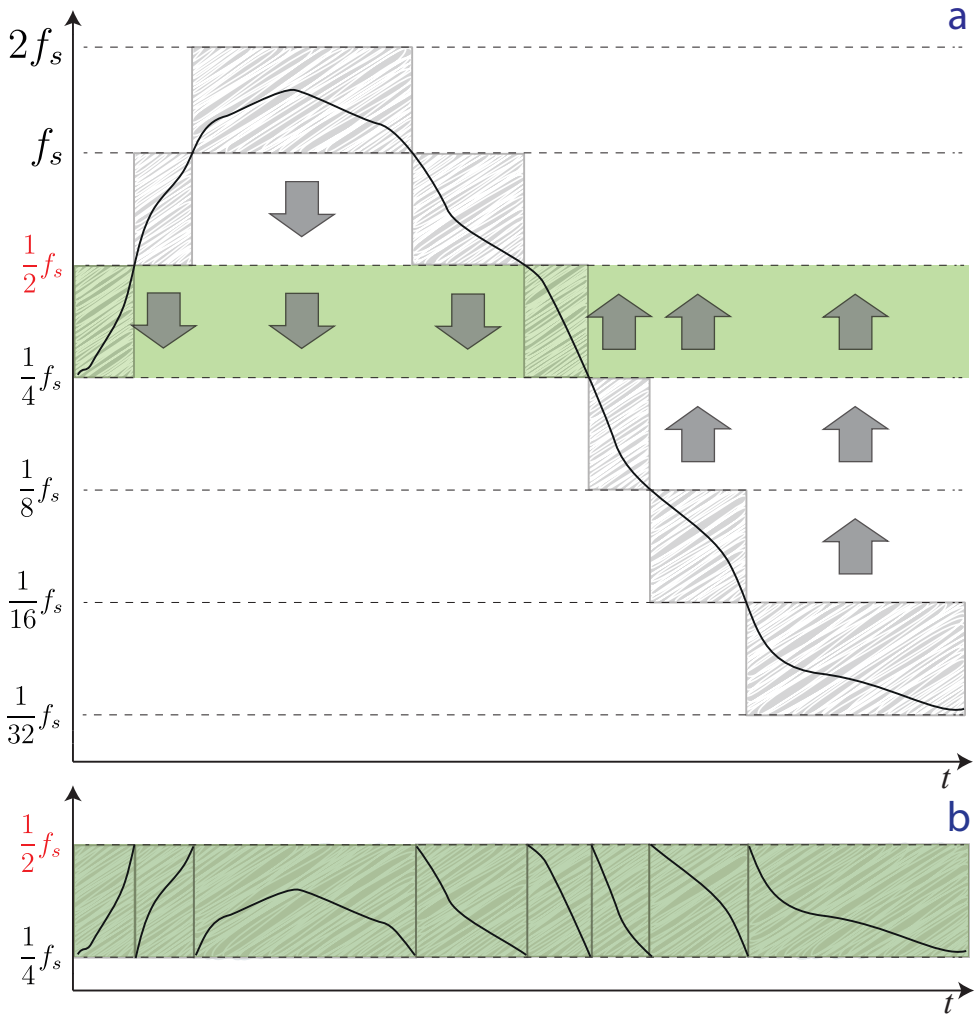

FIGURE 3.5: a) Sketch of the instantaneous frequency $u'(t)$ of a one-component function $T(u) = \cos(2\pi u)$ in the time-frequency plane. Vertical axis is in multiples of view sampling frequency f_s in logarithmic scale with Nyquist frequency marked red. b) The plot is signal is braked into time-frequency boxes (grey) with widths σ_i, which are then shifted in the target band (green).

is that transfer function is generally discontinuous at the regions boundary. However, the lines of maximum brightness (white lines) are continuous. Indeed, looking closer

at the behavior of the isolines at the boundary of adjacent regions σ_i and σ_j where $b_j = b_i + 1$ we immediately see that for $T_i(v)$ attains maximum brightness on a isoline where $v = 2^{b_i}$ since $T_i(v) = T(0) = 1$, while at the same time $T_j(2v) = 1$. In other words, every line from the segment σ_i is continued in the segment σ_j and additionally a new line is inserted in the middle between two existing lines. That is the line density is maintained automatically at a constant rate by introducing and terminating new lines (see Figure 3.7.b).

The final step we take to improve this result greatly, is combining several harmonics in the transfer function.

But first, we would like to mention briefly an interesting connection of the described operation to the Bernoulli map, well-studied in chaos theory, thus giving an interpretation of our method on at a completely different viewpoint, not going into much detail. The Bernoulli (or bit shift) iterated map is defined by the Equation 3.10:

$$f_{n+1} = 2f_n \mod 1 \tag{3.10}$$

where $x \mod 1$ stands for taking the fraction part of x. Essentially the Equation 3.7 is analogous to applying b_i iterations of the Bernoulli (or bit shift) map to the argument u, destroying highest b_i bits of information of the binary representation of u.

3.5.5 Multicomponent Transfer Function

The smoothness of the function can be further improved by adding lower-frequency harmonics with decreasing amplitude. We choose exponential weights with base 2 to sum up the harmonics. The resulting isosurface-generating transfer function is then given by a harmonic sum in Equation 3.11

$$T_i(v, v') = \sum_{n=0}^{r} 2^{-n} \cos\left(\frac{2\pi}{\lambda} 2^{-(b_i+n)} v\right) \tag{3.11}$$

where we take parameter r equal to 3 or 4. This is the suggested transfer function in its complete general form. In the one-dimensional setting we see an overlap of frequencies in the time-frequency plane, resulting in more uniform distribution of instantaneous frequency in the desired frequency band, gained at the cost of the increased bandwidth as illustrated by Figure 3.6.

In the 2D setting the overlap on the boundary produces smooth transitions between regions. Also the brightness of the line gets contribution from harmonics of different scales, so longer lines are brighter. To refine the result we apply a small amount of Line Integral Convolution filtering along the isolines as a post-processing step, eliminating subtle segmentation artifacts, if still present. Compare for example Figure 3.7.a, where one fixed frequency is used for the whole image to Figure 3.7.b, where several harmonics are combined.

3.6 Implementation and Results

3.6.1 Slicing Plane for Volume Rendering

To evaluate the applicability of the presented approach we realized a proof-of-concept implementation, incorporating our novel technique as well as traditional direct isosurface rendering methods.

Our interactive visualization system (presented in the Figure 3.9 and Figure 3.8) is a straightforward extension of the ImagVis3D open source project, which is capable of rendering large 3D volumes with a variety of widely adopted techniques.

Our hybrid visualization consists of two components: a semi-transparent current isosurface and the planar slice through a volume, imaging the discussed transfer function.

The purpose of the slicing plane is to provide a dense visualization of the isosurfaces section, that is to give the big picture of the shape of all isosurfaces, intersected by the plane, and to allow the user to focus on one isosurface.

The current rendered isosurface is determined by the isovalue selected by the mouse cursor on the slicing plane. As far as the interaction goes, the user has full control over the position and orientation of the slicing plane as well as over the viewpoint.

The core of the implementation is a basic OpenGL shader, computing the view-dependent transfer function value on the slicing plane. For the sake of the completeness the source code is provided in Section 3.9. As follows from the code, the base frequency of the function is taken depending on the gradient magnitude and the current level of detail, which is estimated by the QueryLod function.

3.6.2 Web Page Demo

In order to deliver a seamless demonstration of the isocontouring function and to show its computational efficiency, the author has built a demo web-service, exploiting the experimental WebGL support of the latest versions of modern browsers. The developed HTML5 and JavaScript application, running directly in browser, allows to explore the effect of our function applied as an image filter and in a magic lens fashion on a set of real-world images. The function is computed in real-time in the shader, resulting in a highly interactive user interface. The demo is available at `http://matvict.github.io/isocontouring/index.html`

To interact with the application it is recommended to use the latest versions of Google Chrome, as it provides the most complete and effective WebGL support. At the moment current versions of Mozilla Firefox and Apple Safari (with experimental features enabled) are also supported. In the future we are going to make the demo compatible with the upcoming versions of Internet Explorer.

Application of the transfer function to real-world images apart from the artistic effect is useful for exploring the smooth gradient behaviors. For instance, its isocontour-enhancing property allows to visualize compression artifacts, arising from the block structure of the JPEG images (see Figure 3.11), which are barely noticeable in the color images.

3.7 Limitations of the Suggested Approach

It is important to mention that the use of the isocontouring transfer function is only justified when there are some meaningful isosurfaces to extract, that is the isosurfaces significantly larger than the pixel size. For noisy scalar fields with unstable gradients such as CT scan images (see Figure 3.12), the noise regions do not provide any useful information. Our transfer function is designed to discard gradient magnitude as much as possible to highlight isocontours independently of the local contrast. So far, serving to the advantage of our method, this feature turns out to be undesirable for images with significant amount of noise, because our technique immediately makes even low-contrast high-frequent noise prominent in the visualization. That is, the smoothness of the input data is an important requirement for successful use of our transfer function. To overcome this problem for the real-world photo-images images in the 2D demo some Gaussian smoothing is applied to achieve continuous isoconotours.

We do not touch the question of the quality of the resulting visualization specifically for the tangent fields, instead addressing it for generic DFV methods in Chapter 2. Intensive research in this area is done, including development of verification methods [83] [25]and guiding heuristics [31]. The proposed technique being based on adaptive view-dependent refinement of the displayed isoline contours differs from the traditional approaches, but it may still benefit from the accuracy estimation. Moreover, as demonstrated by the implementation, the novel technique can be combined with the well-studied direct isosurface rendering.

3.8 Conclusion

In this chapter we have explored the problem of dense visualization of tangent vector fields, presenting a simple to use and effective technique based on a specially-designed transfer function. We demonstrated the utility of the approach in two practical aspects:

- for development of novel interfaces for isosurface selection in volume rendering software
- of exploration of gradient fields in photo images

The first of these applications features a dense visualization of isosurface sections with a plane, guiding the user in isovalue selection. The second application, is mostly a demonstration of the artistic and smooth gradient-revealing effect of our method on photo images. We exploit its high run-time efficiency implementing it entirely in a web browser. We believe that dense visualization of tangent vector fields using isocontour of the scalar potential can find its place as an instrumental technique in the toolkit of the visualization scientists among other types of dense visualization.

3.9 Example OpenGL Shader Implementing Isocontouring Transfer Function

```
const float pi = 3.14159265358979323846264;
// number of the harmonics in the transfer function
const int n_harmonics = 6;
// smallest allowed value for the gradient magnitude
const float gradient_magnitude_cutoff =  1e-6;
// ratio of the harmonic amplitudes of consecutive scales
const float amplitude_ratio =  0.75;
// log2 of the nyquist frequency
const float log_nyquist_freq = - 2;

vec4 MagicTransferFunction(in vec3 pos)
{
    vec3 gradient = Gradient(pos);
    float gradient_magnitude = length(gradient);
    if (gradient_magnitude < gradient_magnitude_cutoff)
        return vec4(0, 0, 0, 0);

    float lod = QueryLod(pos);
    int log_base_freq =
      int(floor(log2(gradient_magnitude) + lod));
    float base_freq =
      exp2(log_nyquist_freq - log_base_freq);

    float value = SampleVolume(pos);
    float arg =  2 * pi * value * base_freq;

    float amplitude = 1;
    float scale = 1;
    float sum_amplitude = 0;
    float result = 0;
    for (int i = 0; i < n_harmonics; i++)
    {
        result +=
          amplitude * 0.5 * (cos(arg * scale) + 1);
        scale /= 2;
        sum_amplitude += amplitude;
        amplitude *= amplitude_ratio;
    }
    result /= sum_amplitude;
    return vec4(result, result, result, 1.0);
}
```

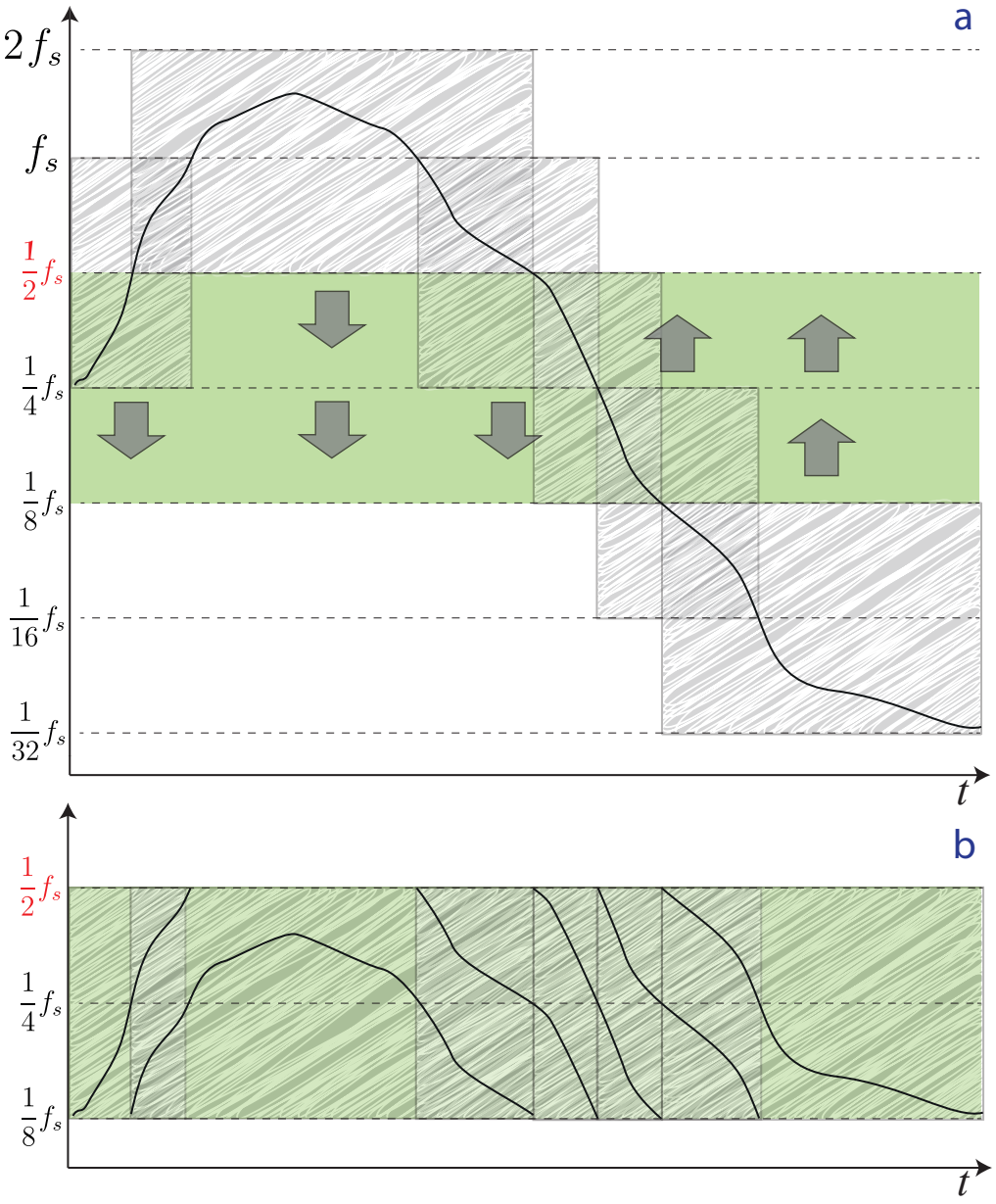

FIGURE 3.6: The same setting as Figure 3.5, except the target frequency band (green) and the slicing rectangles are extended and so the signal segments overlap when shifted into the target band.

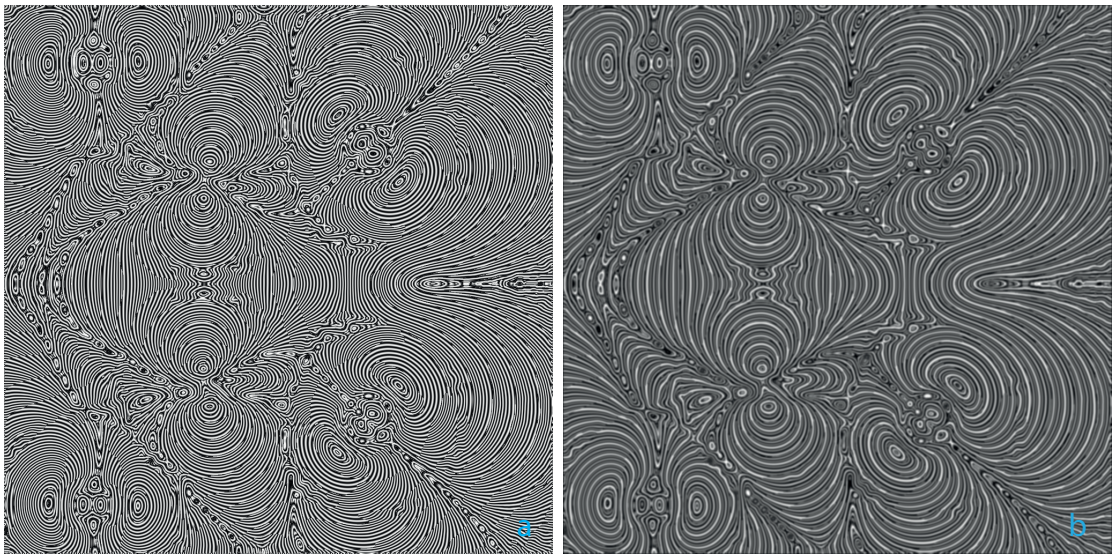

FIGURE 3.7: One slice of Hipip dataset, visualized with a) our transfer function with one harmonic and b) our transfer function with multiple harmonics. Note that a) features segmentation of the domain into segments of the same frequency scale. Visually it results in that on the boundary of segments the thick white lines become thinner and one new thinner line is introduced for each thick one.

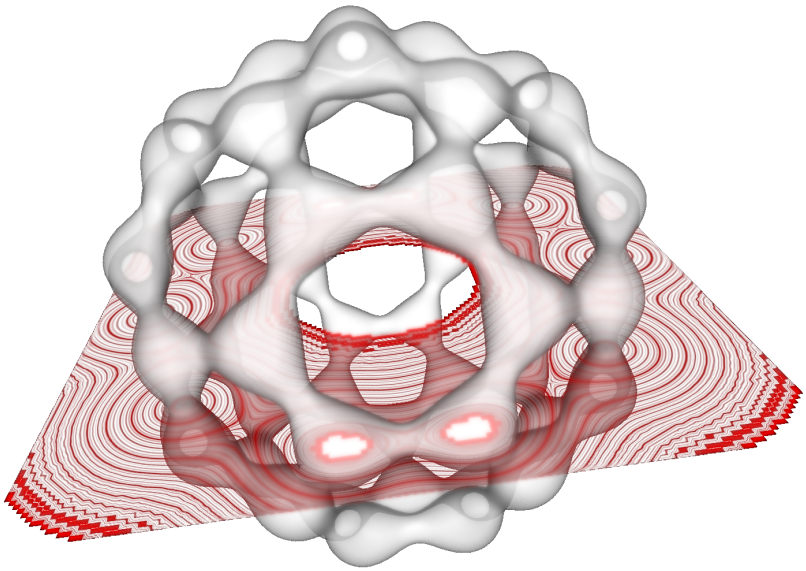

FIGURE 3.8: A isosurface section plane interface for visualizing isosurface sections combined with direct isosuface rendering of a c60 dataset.

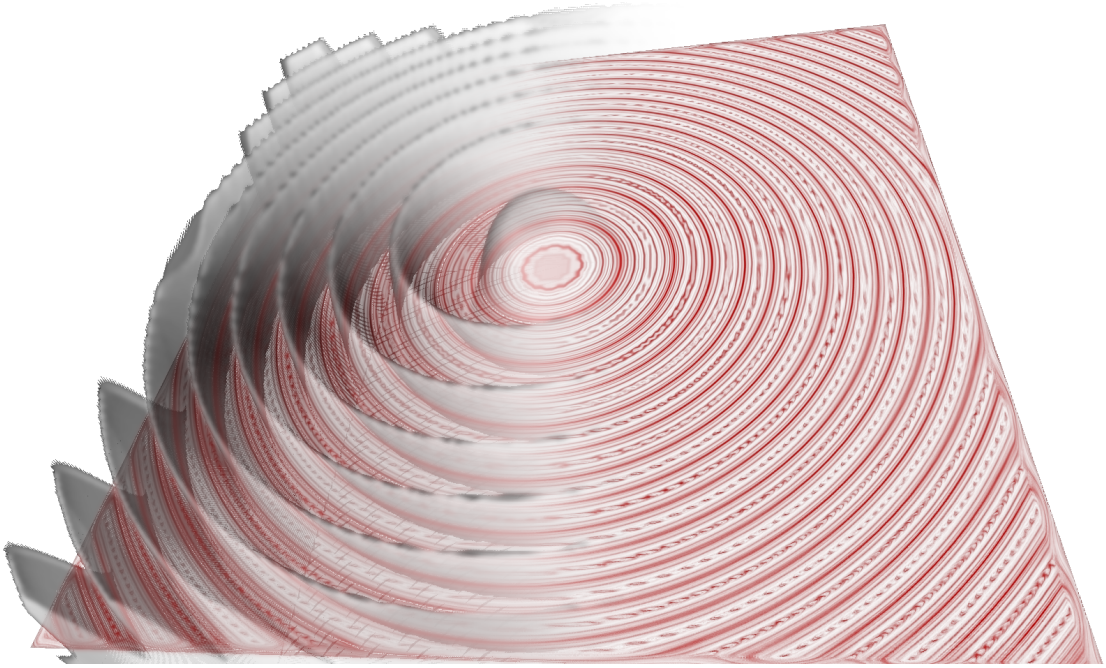

FIGURE 3.9: A isosurface section plane interface for visualizing isosurface sections combined with direct isosuface rendering of a Marschner-Lobb signal [68].

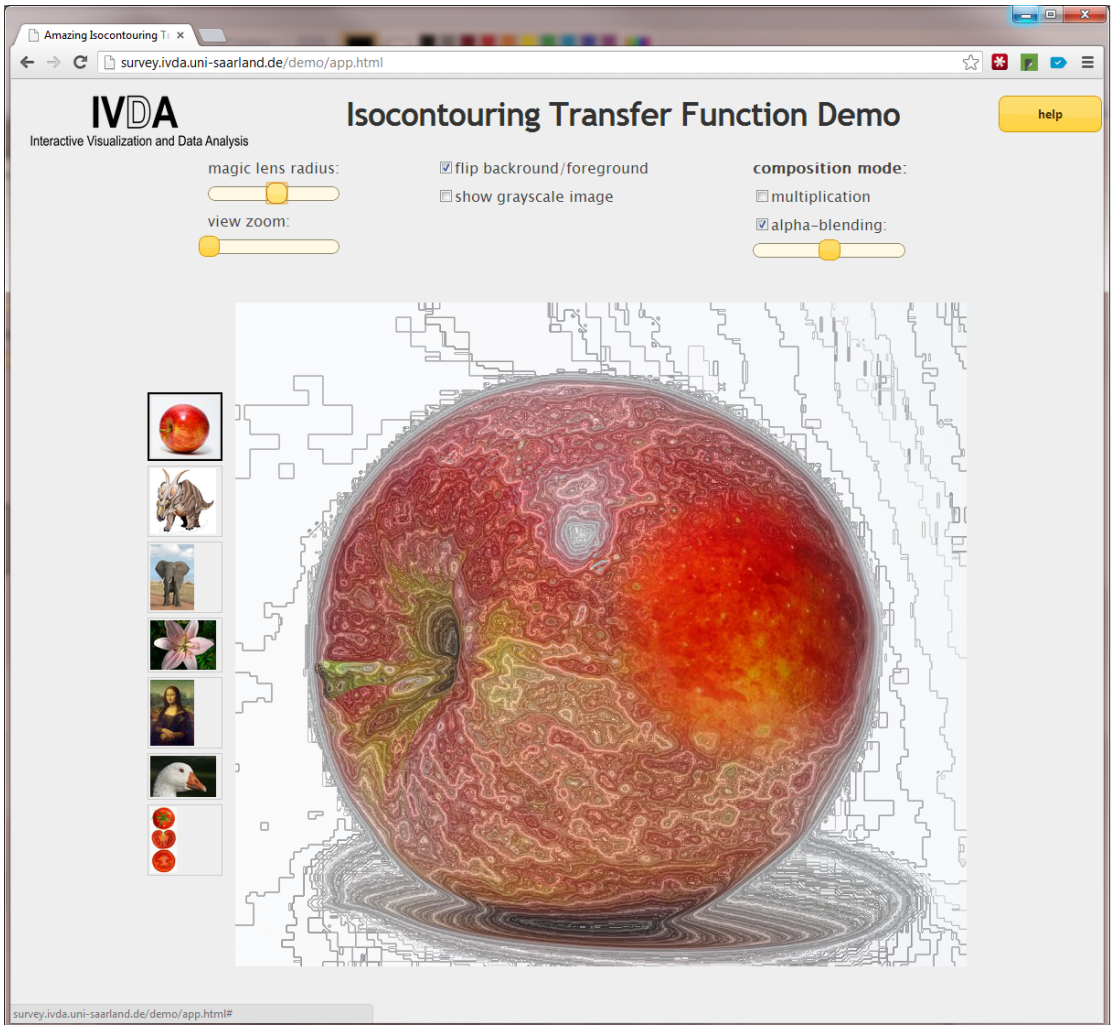

FIGURE 3.10: The interface of the web demo of the isocontouring transfer function.

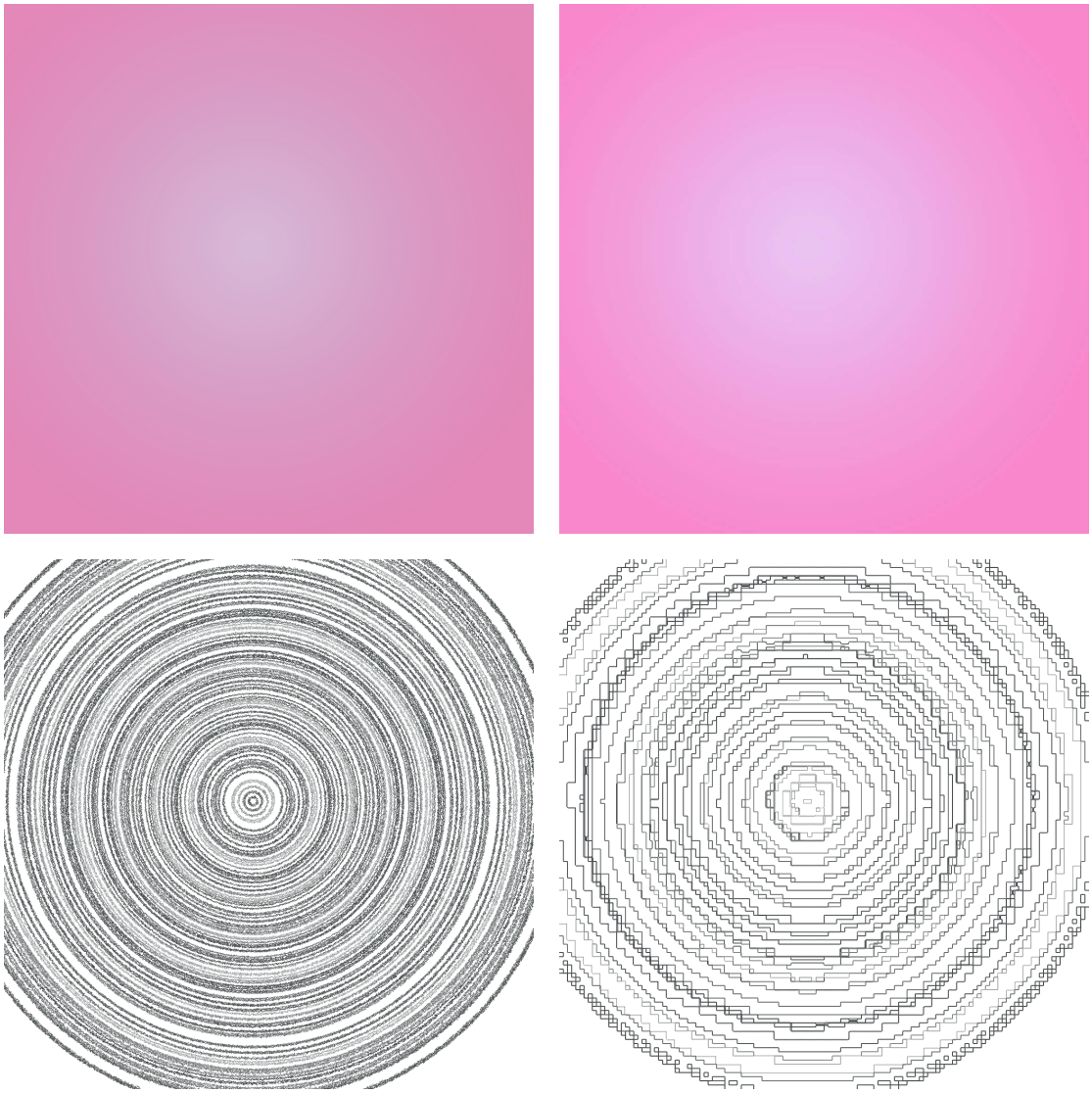

FIGURE 3.11: The same radial gradient image stored as PNG image (top left) and JPEG (top right), with corresponding transfer function images (bottom). The isoline distortion resulting from JPEG compression makes a big difference for transfer function images.

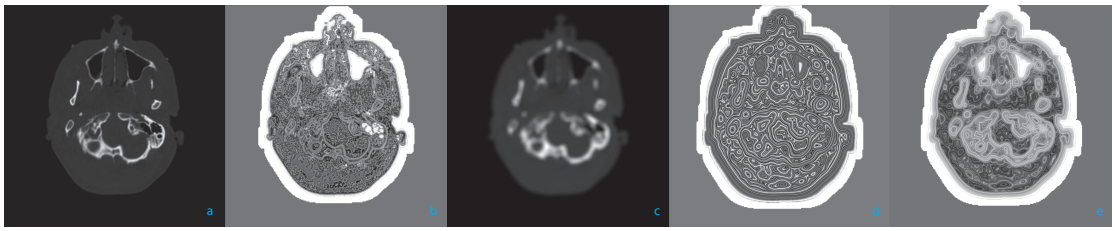

FIGURE 3.12: a) One slice of the human head ct scan dataset . b) The isocontouring transfer function applied to a) highlighting low amplitude noise isocontours. c) Same image as a) with Gaussian blur applied for noise suppression. d), e) Isocontouring transfer function with different number of harmonics applied to c)

Chapter 4

2D Vector Fields: Texture-Based Visualization With Wave Interference

In this chapter we discuss a technique for dense flow visualization, which allows to render evenly spaced streamlines, combining the benefits of two worlds: dense flow visualization and streamline placement (geometric visualization). We develop the ideas applied to the gradient and tangent fields in Chapter 3, now extending the approach to arbitrary flows. The results of the method are high-contrast images with controllable spatial frequency. Relying on a fully automated and well-studied eigenvalue computation scheme, a simple-to-use framework is presented, requiring only two intuitive parameters: the base streamline thickness and thickness variation. The ability of straightforward and (possibly) adaptive control of the spatial frequency of the resulting visualization, related to line thickness, makes the proposed method stand out in the row of LIC-based and streamline techniques. Unlike most texture-based visualizations the results are not fuzzy and can be easily converted to vector graphics for further post-processing. Depending on the choice of parameters 2 types of results can be achieved: the images with forking streamlines and streamlines of varying thickness. Thus, the main advantages of the technique over existing dense visualization are:

- high contrast results
- adaptive spatial frequency control
- few, intuitive parameters
- straightforward mathematical framework

We will start with the introduction of an affinity function between image pixels, intuitively corresponding to streamline thickness, as the whole notion of thickness of a streamline is not defined in calculus and applies only to the visualization images. Next, the color of each pixel in the desired visualization image is represented as a sum of coherent sinusoidal waves propagating from its neighbourhood in the space induced by the constructed function. Finally, a highly scalable iterative numerical scheme is proposed for finding the optimal image conforming to the derived model is presented and results are discussed. In particular, we will explore the possibles the extension of the method to 3D.

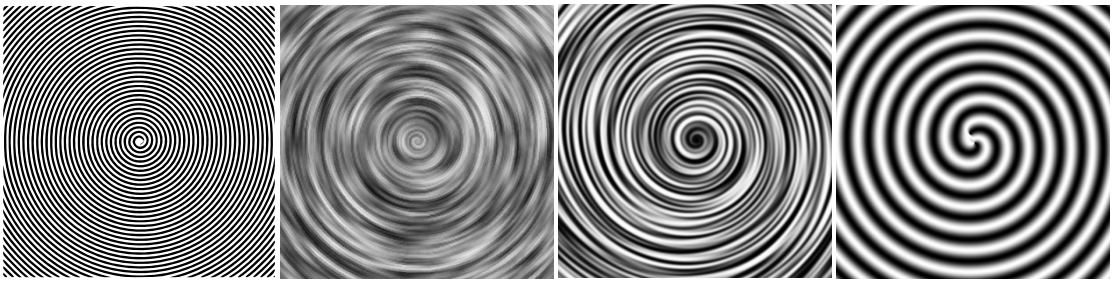

FIGURE 4.1: The results of the visualization technique (leftmost and rightmost) compared to LIC images for the same field (middle-left and middle right).

4.1 Wave Interference Model

4.1.1 Motivation

The result of virtually any dense flow visualization technique is an image with the gradient related to underlying field, since—as known from image analysis—the most important information about image contours is provided by the edges. In particular the direction of the vector field perceived from the picture is mainly orthogonal to the gradient of the image, assuming that the contrast of the image (i.e. the gradient magnitude) is high enough.

The natural idea, would be to drive this property to the extreme, synthesizing an image with high gradient everywhere orthogonal to the vector field. The ideal representation in this sense is hardly reachable in general case, since most known DFV techniques either introduce some kind of angular error between the gradient and the vector field or create regions with zero gradient and thus no information about the flow. Commonly used dense techniques such as LIC distribute this error randomly over the whole image with noise, whereas flow visualization with streamlines and other sparse techniques can be seen as introducing zero gradient everywhere except the streamline edges.

In this chapter a way to model the image consisting of cosine waves orthogonal to flow as well as the process to achieve a descent approximation of such image is proposed. As opposed to commonly used techniques, the proposed method allows automatic elimination of the angular error for certain (basic) types of fields without introducing zero gradients.

4.1.2 Outline of the Visualization Technique

The proposed method can be broken down into several essential steps, pictured in the Figure 4.2. Starting with a vector field and a regular image grid an affinity function (or relative thickness) is computed between cells of the image grid. This process, discussed in details in the next section, involves he computation of area bounded by streamlines and can be done efficiently by precomputing certain line integrals. The essence of the modeling phase of the method is exploiting the relative thickness of any two pixels of the image in order to build a sparse matrix, describing propagation and interference of cosine waves in a net of paths related to thickness function. By the nature of this function such waves depict black and white flow lines in the image space at least locally close

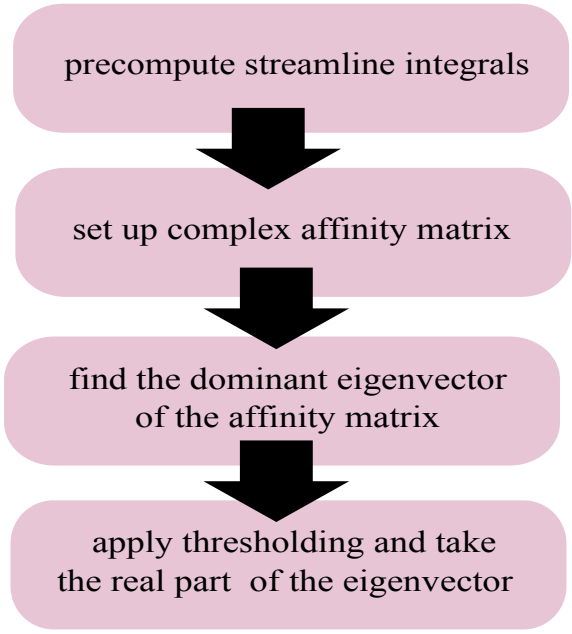

FIGURE 4.2: The main steps of the proposed flow visualization method

to the wave source. Then representing the image as a set of independent wave sources that are placed in the center of pixels a globally consistent image is computed by making them phase-coherent. Showing that the dominant eigenvector of the constructed matrix is the best approximation of the desired image the power iteration scheme is employed for its computation. Finally, the obtained result is refined by separating the frequency and the amplitude parts of the solution.

4.1.3 Idea of a Heuristic Thickness Function

The formulation of the thickness function is very simple and is inspired by the Line Integral Convolution [9] method. The essential fact about this visualization technique is that any two pixels in the resulting image are correlated if they belong to the same streamline. It is often explained by that streamlines traced from the centers of those pixels overlap in the input texture. One interesting observation though is that for a non-trivial field the streamlines starting from different pixels do not exactly follow the same paths, but bound a region with a small area in the texture space, thus producing correlated values as long as the noise is correlated in space as in Figure 4.3. Specifically given two pixels, the discrepancy of the line traces started in their centers has very little effect on the difference of the resulting pixel colors if the distance between lines is less than the noise pixel size.

In the presented approach a function is proposed to measure the signed distance between pixels orthogonal to the flow field, which can be thought as a minimum thickness in pixels of a rendered streamline segment that covers the centers of two pixels. It's done in two computational steps, first tracing for each pixel a streamline in both directions of the flow using a reasonably small number of steps, similar to the LIC method and

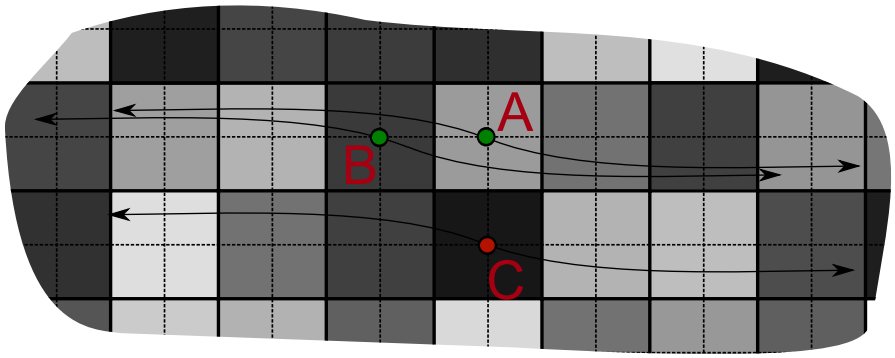

FIGURE 4.3: Streamlines traced from pixels A and B though not the same, sweep the same region in the texture, while streamline traced from C sweeps completely different region.

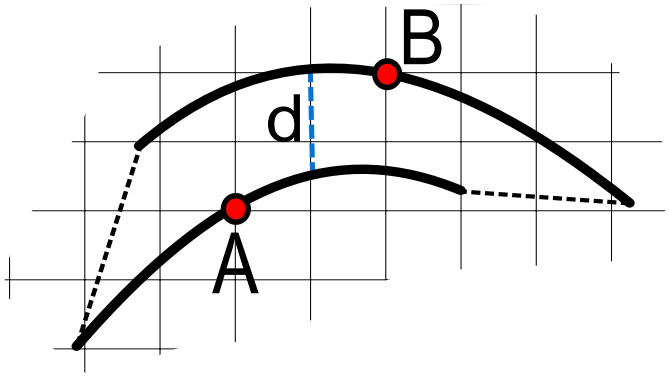

FIGURE 4.4: The relative streamline thickness d for pixel centres A and B is measured as the average height of the pictured polygon.

second, estimating the line thickness d_{AB} between two pixels A and B using the signed area S_{AB} of a small polygon, bounded by the streamlines l_A and l_B as in Equation 4.1,

$$d_{AB} = \frac{2S_{AB}}{|l_A| + |l_B|} \tag{4.1}$$

where $|l_A|$ stands for the length of the streamline l_A. While this heuristic only approximates the distance between streamlines if the change in distance between traced segments is low compared to the image grid size, it has proven to be sufficient for the purpose of this work and can be be efficiently computed.

Since the streamlines l_A and l_B do not intersect the considered polygon is simple and its area S can be computed with the formula, commonly known from computational geometry:

$$S = \frac{1}{2}\sum_{i=0}^{n}(x_i y_{i+1} - x_{i+1} y_i) \tag{4.2}$$

where (x_i, y_i) are the points of the polygon and $(x_0, y_0) = (x_{n+1}, y_{n+1})$.

The advantage of this formulation is that it allows to compute thickness function of

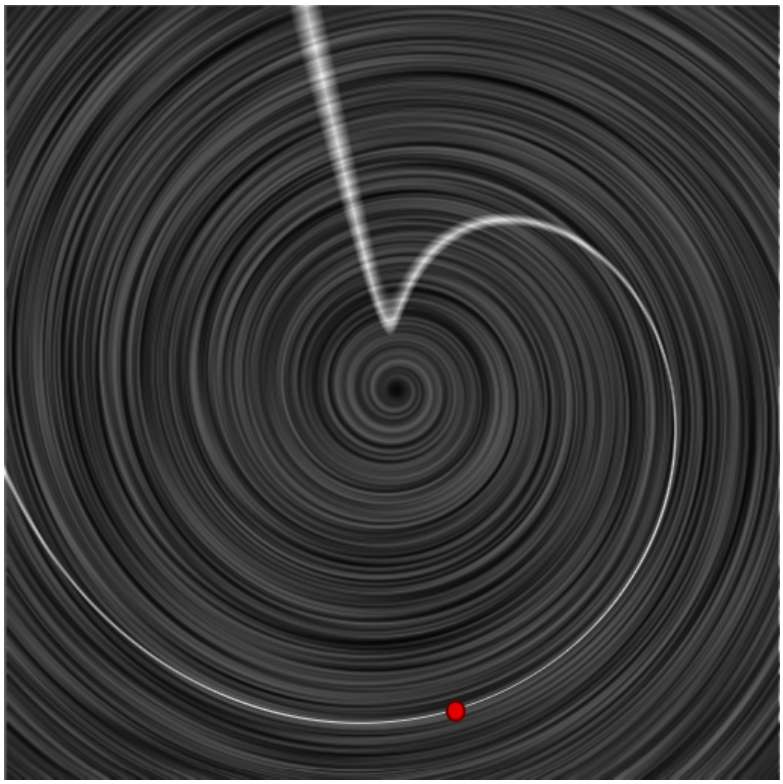

FIGURE 4.5: The set of all pixels with relative thickness to the red point less than a pixel size underlaid by the LIC image of the same field.

any two pixels A and B given the boundary points for l_A and l_B and the precomputed sums over l_A and l_B without computing the sum over the whole polygon. Indeed, let P_i be ordered set of points of l_A, and Q_i are ordered set of points of l_B and for brevity $\delta(P,Q) = PxQy - QxPy$. Then,

$$S = \frac{1}{2}\left(\sum_{i=0}^{n-1}\delta(P_i,P_{i+1}) + \delta(P_n,Q_n) + \sum_{i=0}^{n-1}\delta(Q_{i+1},Q_i) + \delta(Q_0,P_0)\right)$$

or, keeping in mind that $\delta(P,Q) = -\delta(Q,P)$ and defining $S[P] = \sum_{i=0}^{n-1}\delta(P_i,P_{i+1})$:

$$S = \frac{1}{2}\left(S[P] + \delta(P_n,Q_n) - S[Q] + \delta(Q_0,P_0)\right) \tag{4.3}$$

Thus, for computation of the area S only the integral sums $S[P]$ and $S[Q]$ for lines l_A and l_B as well as starting and ending points P_0, P_n , Q_0, Q_n are required.

Let us now explore some useful traits the constructed function. Its key benefit is that it can be employed as a simple means to check if two pixels belong to the same streamline. The relative thickness d_{AB} of such pixels A and B is zero, since basically the area between streamlines l_A and l_B is measured. It is important to mention that the closer given points are to each other compared to the integrated lengths $|l_A|$ and $|l_B|$ the more accurate is the computation of d_{AB}. As shown in the Figure 4.5 choice of too small $|L_A|$ and $|L_B|$ may cause significant deviation from zero for the points that are far apart. Nevertheless, the constructed function might find its applications for example in

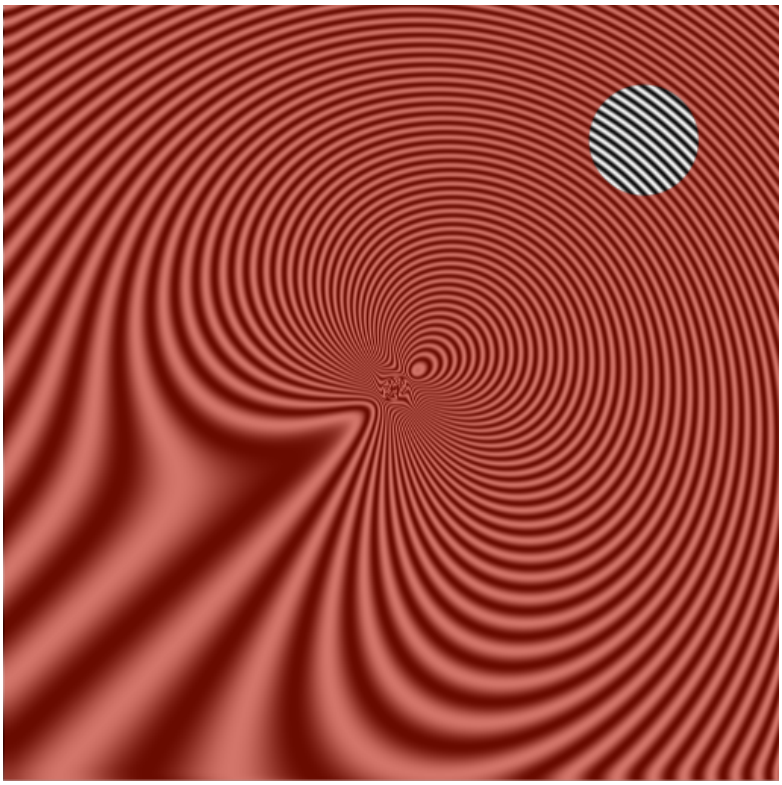

FIGURE 4.6: Outside a local neighborhood (in the red area) the drain field is severely distorted. In the actual visualization method much smaller neighborhoods of one pixel are used.

a computationally cheap (not requiring any vector field integration apart from distance calculation) algorithm for multiple streamline tracing, that in a greedy manner adds a new pixel to the line with smallest thickness relative to the pixels on the line.

Another property, that is naturally to associate with line thickness is that $d_{AB} = d_{AC} + d_{CB}$ for any three pixels A, B and C, which generally doesn't hold in the entire image unless the distance between streamlines is constant. Nevertheless, assuming that in a small image segment u around pixel k in Figure 4.6 this property is well approximated, one could define a streamline image u as in Equation 4.4

$$u_i = \cos(\varphi d_{ki}) \tag{4.4}$$

where thickness of lines and spacing between them is $\frac{\varphi}{\pi}$ pixels.

4.1.4 Flow Visualization With Relative Thickness And Wave Interference

In this section a model is proposed for the flow visualization image with image gradient is orthogonal to the flow field using Equation 4.4 as a descent approximation of small region of the target image. This equation can be interpreted as following: the pixel colors in the neighbourhood of pixel k are computed as the amplitude of a cosine wave, emitted at position k propagating in the space induced by thickness function d. The Figure 4.7

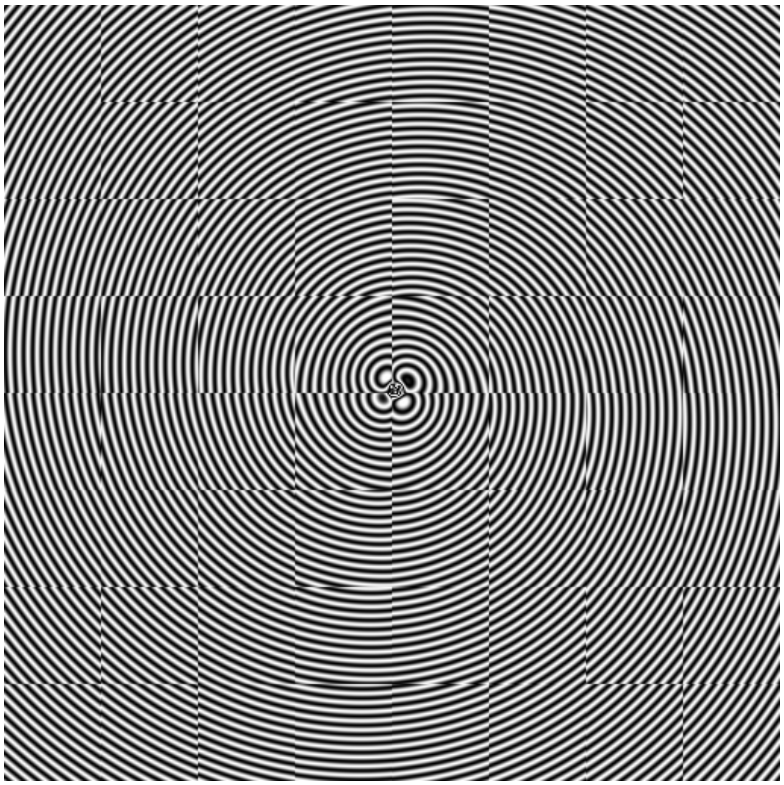

FIGURE 4.7: Image of a drain field consisting of 64 wave blocks blocks with different phases on each block.

illustrates a grid of 8x8 independent wave sources producing 64 independent regions, each locally approximating a 64x64 segment of the visualization image. To produce a globally consistent image wave sources have to be made coherent. For this reason let's introduce for each pixel k a phase shift ψ_k in Equation 4.5:

$$u_i = \cos(\varphi d_{ki} + \psi_k) \tag{4.5}$$

And as $d_{ki} = 0$ it immediately follows:

$$u_i = \cos(\psi_i) \tag{4.6}$$

Now, let us extend the Equation 4.5 and Equation 4.6 to the complex plane, setting $A_{ki} = e^{\imath\varphi d_{ki}}$ and $v_k = e^{\imath\psi_k}$. Now $u_i = \Re[v_i]$ and the Equation 4.5 and Equation 4.6 can be rewritten as:

$$v_k = A_{ki} v_i$$

Instead of forcing this requirement to be true for every k and i it can be relaxed by requiring the pixel color v_k to be the weighted sum of the incoming waves from its neighbourhood as in Equation 4.7:

$$v_k = \sum_{i=1}^{n} w_{ki} A_{ki} v_i \tag{4.7}$$

with some non-negative weights w_{ij}.

To compensate for the inaccuracy in distance d computation weights can be set $w_{ij} = 0$ if $i \notin N(j)$ and $\sum_{i \in N(j)} w_{ij} = 1$ for some neighbourhood $N(j)$ of pixel j. Thus the color of the pixel is required to be consistent with the colors of its neighbour pixels.

The target image can now be computed as the eigenvector of matrix $H_{ij} = w_{ij}A_{ij}$ corresponding to the maximum eigenvalue. Indeed, since $d_{ij} = -d_{ji}$ setting $w_{ij} = w_{ji}$ makes H a Hermitian matrix with real eigenvalues. With $w_{ii} = 0.5$ eigenvalues of H are as well non-negative obtaining the values in the range $[0,1]$ by the Gershogorin's Theorem with at least one eigenvalue not less the 0.5 since the trace of the matrix is $0.5 \cdot n$.

An optimal solution can be found in a sense that the interference of the emitted wave in each pixel and the waves, coming from its neighbourhood is constructive. Eigenvalue λ can be thought of as an amplification factor.

4.1.5 Scalable Numerical Solver Implementation

The model setup in previous section results in a large sparse matrix. For instance, given a typical image of 2^{10}x2^{10} pixels the matrix numbers 2^{40} entries with only 2^{20} x $|N|$ non-zero elements where $|N|$ is the size of the considered neighbourhood. To find the eigenvector, corresponding to the largest eigenvalue, the power iteration method can be used in the following form

$$v^{(k+1)} = c^{(k)} H v^{(k)} \tag{4.8}$$

with the normalization factor $c^{(k)} = \frac{1}{||Hv^{(k)}||_\infty}$

For the sake of completeness a brief justification of this method is provided here, whereas the detailed discussion can be found in the textbook by Quateroni et al. [88]. Since H is is a Hermitian matrix, it has a full system of orthogonal eigenvectors e_i with corresponding real eigenvalues λ_i, assuming that eigenvalues are ordered in an decreasing sequence such that $\lambda_1 \geq \lambda_i$. This means that for any vector v there exists an expansion $v = \sum_{i=1}^{n} v_i e_i$. Since e_i is an eigenvector and $He_i = \lambda_i e_i$ for any power k of matrix H it holds that:

$$H^k v = \sum_{i=1}^{n} \lambda_i^k v_i e_i = v_1 \lambda_1^k \left(\sum_{i=2}^{n} \frac{v_i \lambda_i^k}{v_1 \lambda_1^k} e_i + e_1 \right). \tag{4.9}$$

thus, for $k \to \infty$ the method converges to a multiple of e_1

Although, this method is usually considered if there exists a dominant eigenvalue such that strict inequality holds $|\lambda_1| > |\lambda_i| \forall i$ in this case this restriction is not necessary. Since all the eigenvalues are real and non-negative, there could not be two eigenvalues with opposite sign and same absolute value so the method can not diverge. In fact, if two maximum eigenvalues are equal the method still converges to a linear combination of their corresponding eigenvectors, which is still an eigenvector.

A bit of caution should be exercised on the initialization step, since the starting vector $v^{(0)}$ should necessarily contain a e_1 component. A good choice for it is therefore a random vector.

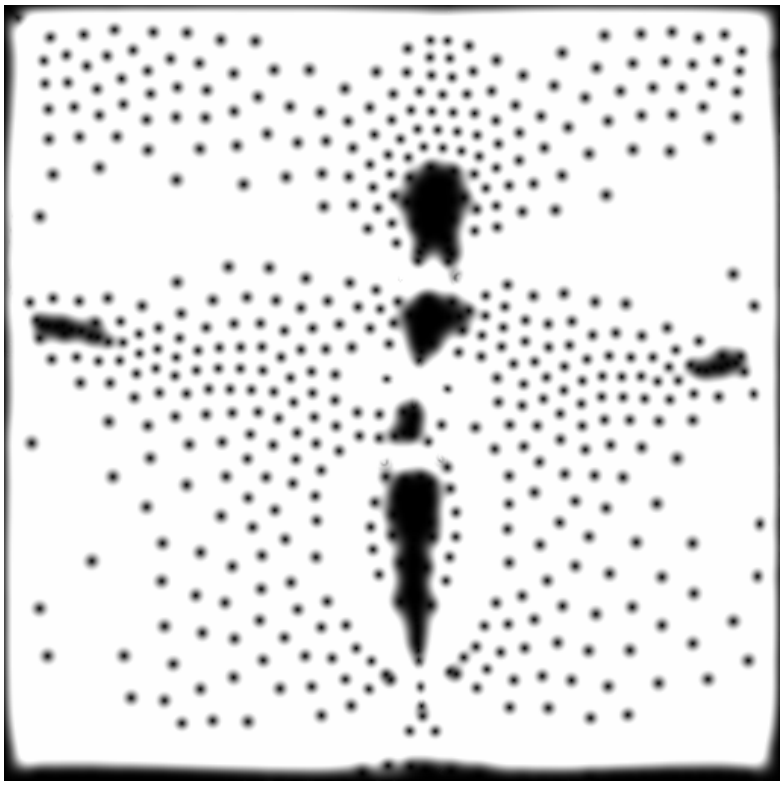

FIGURE 4.8: Amplitude image on one of the early iterations

The rate of convergence of considered scheme is equal to the ratio of sub-dominant and dominant eigenvalues of matrix H and further speed improving modifications are possible, for example with the use of inverse iteration. The main advantage of this method is that compared to more sophisticated techniques for the eigenvector computations like QR-decomposition this scheme is extremely simple and memory efficient, since it requires to store only the original matrix H and the latest solution $v^{(k)}$. Moreover, it involves only one (sparse) matrix-vector multiplication which allows a fast parallel implementation on the modern GPU hardware.

A CUDA-based moderately optimized realization of the solver on GeForce GTX 580 video card, developed as a proof of concept, computes a typical 512x512 image in about 20 seconds, making roughly 20000 iterations.

4.1.6 Interpretation of the Results

After the computation of the complex vector v the target real-value image u has to be obtained. To get the desired phase image, let's represent v in the polar form obtaining the amplitude part r and phase part ψ such that $v_i = r_i e^{\imath\psi_i}$. The amplitude part is pictured in the Figure 4.8. It can be thought of as a confidence measure of the achieved wave representation, since it shows how large is the mutual influence of the pixel and its neighbourhood on each other when propagating waves with the chosen frequency. For example Figure 4.9 illustrates that the amplitude stays low close to the image boundary as well as at the bifurcation points. This indicates the region where the flow field is distorted in the representation.

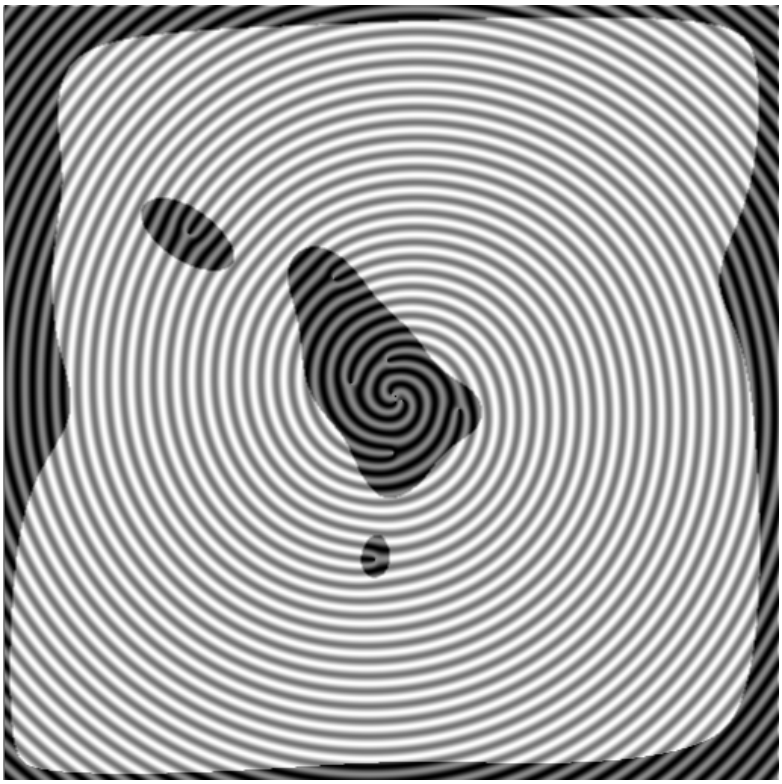

FIGURE 4.9: Image computed on one of the iterations, with the corresponding amplitude map overlay (6% threshold) . The image reveals that bifurcations as well as boundaries are regions with low certainty.

To obtain final visualization images first a threshold should be applied to the amplitude factor, setting $r'_i = \lceil r_i \rceil$, and then taking the real part of the image. Since v is normalized with $||.||_\infty$ norm the resulting image has amplitude of zero and one.

4.1.7 Managing Frequency

There considered method needs two parameters: the length of the traced line segments L and the base frequency φ. While the short integration length allows to keep frequency slightly varying around the base frequency at the cost of introducing black and white line bifurcations, increasing the length causes significant deviations from the base frequency, taking into account long-term flow behaviour as in Figure 4.10. Consequently, there is a need to limit the frequency for large L to avoid aliasing in the resulting image. This can be achieved by rescaling frequency in Equation 4.5 applying adaptive smooth thresholding $F(f)$ such that $F(\varphi d_{ij}) < f_N$ where f_N is Nyquist frequency. A sigmoid function can be used of type $F(f) = \frac{2}{e^{-8\ln(3)f}+1} - 1$, where f is normalized such that $F_N = 1$.

4.2 Some Implementation Details

The described technique was implemented as a part of a interactive flow visualization application. It was based on different principles compared to the software discussed

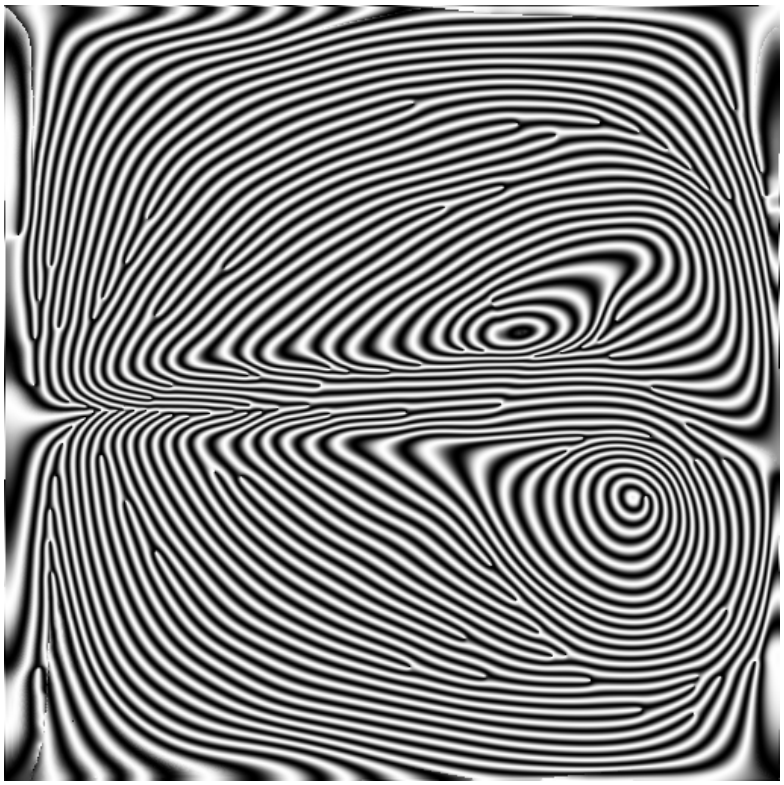

FIGURE 4.10: Variation in frequency in the image obtained with a large integration lengths (256 steps)

in the previous chapter. In particular—as opposed to quality metric computation—the method implementation details were not clear from the very start and evolved as a result of a long trial and error process. That is, the software was meant to be a sort of sandbox for exploration of different flow visualization methods, image filters and their combinations and eventually the gradual development of the new technique. To achieve this goal a vivid graphical user interface with instantaneous feedback from the parameter change was necessary. It was the reason to plan GPU implementation of the time-consuming algorithms. The need to experiment with different computational modules suggested a modular design. The batch processing of large amounts of data was not necessary.

The resulting system integrated several flow visualization methods, and basic image processing including, but not limited to LIC, streamline placement and the proposed wave interference technique. Flow visualization and metric computation shaders, developed for the system described in the previous chapter, were reused, as well as the collected database of flow samples and noise textures.

The application was built on top of the Microsoft .NET framework version 4.0 in C# programming language. The object-oriented architecture, vivid user interface design utilities, rich standard library and unique language features make it an ideal framework for rapid prototyping of windows applications.

The interface design, as presented in Figure 4.11, was based on a canvas metaphor: the visualization image is drawn in the frame buffer and can be used as an input to various image processing effects and other visualization algorithms. The adjustment

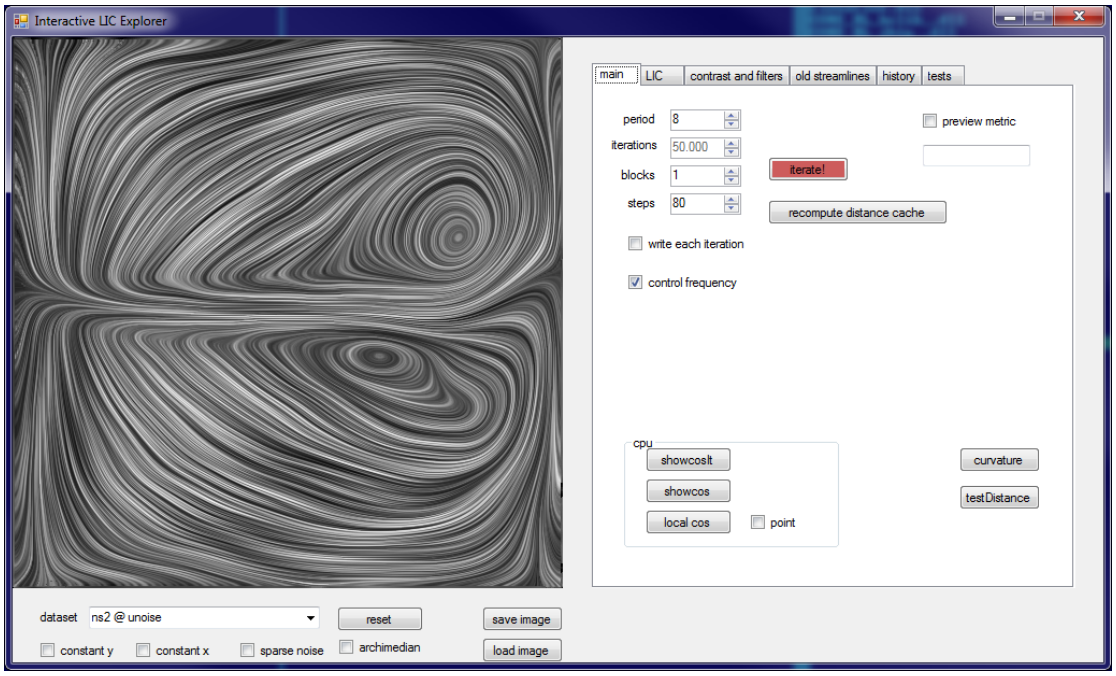

FIGURE 4.11: User interface of the visualization software, featuring implementation of the proposed technique.

of parameters took immediate effect on the canvas image. The basic image processing (histogram equalization, image filters) was done with the means of ImageMagick library. The flow visualization modules contained the mixture of OpenGL and Cuda code. The computational modules were implemented as objects of specific classes, exposing parameters to the GUI interface via Windows Forms data binding mechanism [74]. These classes, providing a common abstract interface, encapsulated launches of the OpenGL shaders, ImageMagick library calls and CUDA kernel launches.

The prototype of the visualization technique was first implemented in a form of a Microsoft .Net assembly. The choice of the platform allowed as well to exploit the data-parallelism of the method without additional effort, being possible because Microsoft C# language natively supports such elements of functional programming as map and reduce operations with type-safe lambda-expressions, with parallel language integrated queries mechanism (PLINQ)[73]. It made computing each pixel in parallel, using a CPU thread pool, a trivial task, making almost no difference to sequential loop implementation. This way, subtle runtime overhead, introduced with the use of .NET framework was dramatically overcompensated with huge performance multiplication gained from parallelism.

The implementation was further the visualization modules to NVIDIA CUDA platform, since hardly any CPU implementation can compete with a GPU program when it comes down to image computation where operations are relatively simple and independent for every pixel. Indeed, the achieved speed-up was about a factor of 50. The CUDA library, responsible for flow visualization with wave interference implemented the following services on the GPU:

- streamline tracing, storing end points and accumulated integrals in the GPU memory for each pixel as discussed in Section 4.1.3

- complex distance matrix computation for a 3x3 stencil, using the precomputed streamline data frequency parameters; only the non-zero elements of the matrix were stored;
- iterative eigenvalue computation (dot-product of the sparse matrix with the solution vector)

The mentioned operations were implemented as cuda kernels, as well as several other algorithms for visualization, including streamline placement algorithms.

After a thorough profiling, it appeared to be necessary to optimize the memory access in the initial implementation, taking into account memory coalescing [72], which resulted in 4 fold performance increase compared to non-optimized version. However, the effect of this preliminary optimization diminished after migration to high-end graphics hardware with computation capability 2.0 (instead of 1.3). For the the typical GPU-powered map-reduce operations and memory allocation Thrust template library [116] was used, which has since then become a part of NVIDIA GPU Computing SDK. Integration of the NVIDIA CUDA implementation in the .NET project required a little additional effort. The main GUI shell is a managed assembly in the .NET terminology, running under the Common Language Runtime virtual machine. It required a small wrapper assembly in Microsoft Managed C++, which is capable to mix CLR and native code, bridging the GUI and the native libraries.

4.3 Conclusion

The dense visualization approach presented here on its own might be a valuable tool in producing flow images with manageable properties. Nevertheless it has its limitations and can benefit from certain additional post-processing operations and as we see in Chapter 5 can be substantially refined, and reformulated 1using traditional image filters, without the need for thickness heuristic.

One feature of the method is that the choice of the base frequency dictates the size of the details, that can be represented by the visualization. That may be a desirable effect, when it is known from a certain a priori information about the dataset that everything above certain frequency is noise. Otherwise, image resolution and φ should be chosen high enough to capture small flow details, which in turn may lead to very high-frequent high-contrast images that are disturbing to a human eye. Second downside of the method is that the field is obviously distorted in the line bifurcation points, especially for small integration lengths. This effect can be largely attenuated with smoothing at one of the following post-processing steps.

A threshold can be applied to the output to achieve black and white images, which can then be easily brought into the vector graphics form. Figure 4.12 illustrates that general purpose bitmap tracing algorithm can be used for this purpose. An open source "Potrace" [98] utility was used to produce this sparse visualization somewhat similar to streamline image, though not exactly the same. It can be further analysed and enhanced with geometry processing algorithms or just used as a scalable visualization free from limitations of a raster picture. The advantages of this picture are especially obvious when printed on paper.

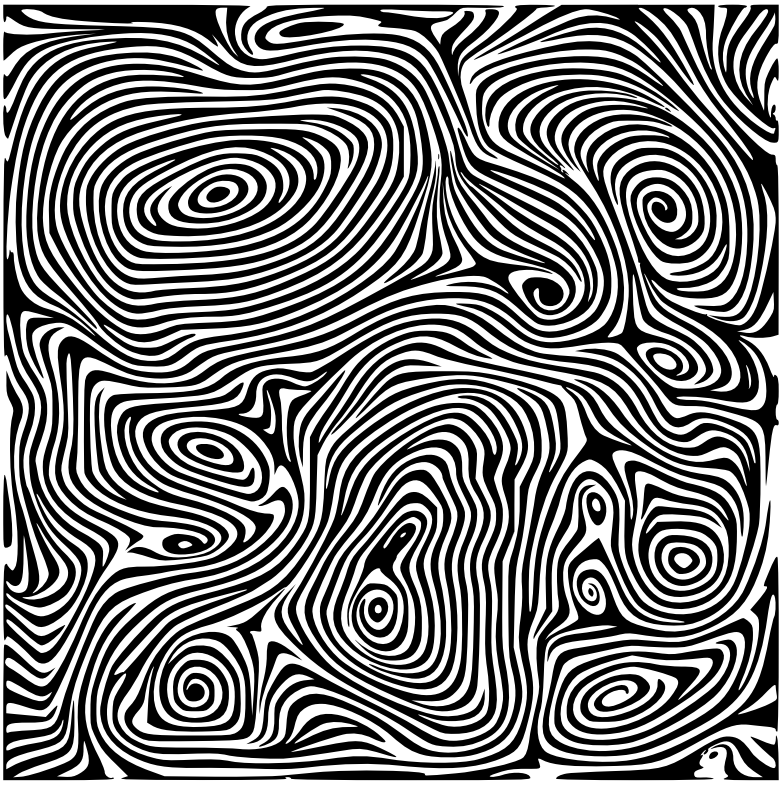

FIGURE 4.12: Output of the method converted to vector graphics.

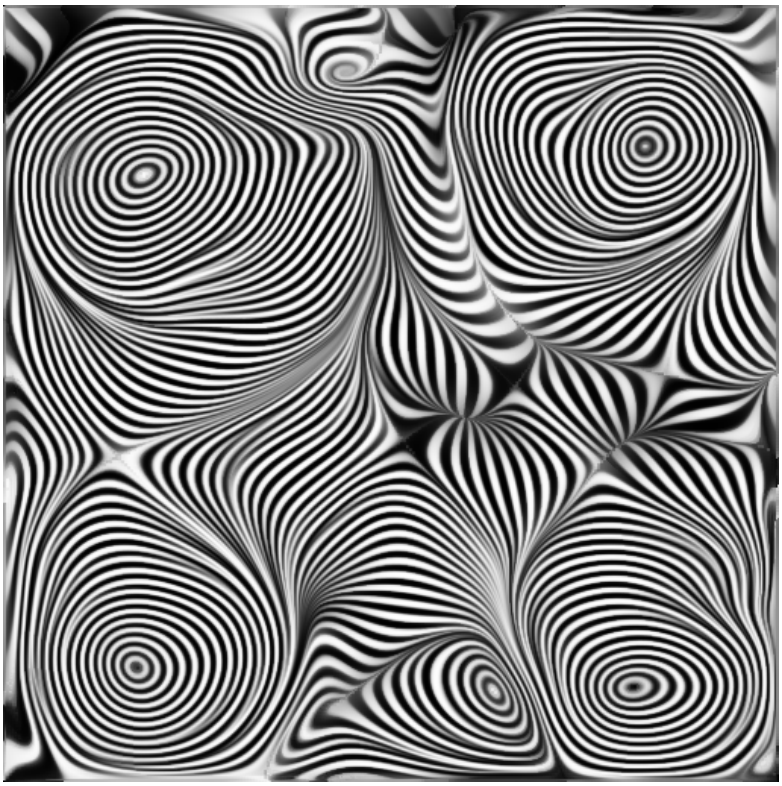

FIGURE 4.13: Output of the method smoothed by a LIC post-processing step.

An alternative approach to create high-quality images is to apply the proposed method following a LIC processing, which add smoothness in the direction of the flow. This scheme results in a much better visual quality than any other LIC application based on a noise texture as can be seen in Figure 4.13.

The presented method allows to control the spatial frequency of the resulting visualization based on the reconstruction of a scalar field, which was naturally available in Chapter 3. This comes at a cost of introducing a thickness function heuristic, making the suggested method stand out from the cohort of conventional dense flow visualization techniques. Departing from this intermediate result, we substantially improve the approach in Chapter 5, not only achieving significantly better control over the resulting visualization using well-studied Gabor filters, but also fitting the new method in the traditional LIC operator framework.

Chapter 5

2D Vector Fields: Texture-Based Visualization With Explicit Frequency Control

In this chapter an effective method for frequency-controlled dense flow visualization is formulated. It is derived from a generalization of the Line Integral Convolution (LIC) technique. The approach consists in considering the spectral properties of the dense flow visualization process as an integral operator defined in a local curvilinear coordinate system aligned with the flow.

Exploring LIC from this point of view, a systematic way is suggested to design a flow visualization process with particular local spatial frequency properties of the resulting image. The method is efficient, intuitive, and based on a long-standing model developed as a result of numerous perception studies. The method can be described as an iterative application of line integral convolution, followed by a one-dimensional Gabor filtering orthogonal to the flow. The utility of the technique is demonstrated on several novel adaptive multi-frequency flow visualizations, that according to the evaluation, feature a higher level of frequency control and higher quality scores than traditional approaches in texture-based flow visualization.

5.1 Frequency Control in Texture-Based Flow Visualization

The domain of flow field visualizations is one of the great success stories of scientific visualization. Practically all engineering and natural sciences benefit from this work. Consequently, a wide variety of specific techniques have been developed and improved over the last few decades.

Within the realm of available vector field visualization methods, there is a number of distinct subareas. One of those subareas is *dense* vector field visualization. Dense vector field visualization techniques in general are known for their ability to give an excellent quick overview of the entire vector field.

The most prominent method for dense vector field visualization is the Line Integral Convolution (LIC) method presented by Cabral and Leedom [9]. This basic method has been analyzed and improved in a large number of publications (see the related work overview in Section 1.6 for details). Though the basic concept behind many extensions to LIC remains the same, the performance and quality of the resulting visualizations has

FIGURE 5.1: A comparison of standard LIC (left), previously proposed high-contrast limited-frequency wave inference method [70] (center), and the novel technique (right). For ground truth comparison, the images are overlayed with semitransparent streamline segments. Using CUDA on a commodity PC, the LIC image took about 5 milliseconds to compute, the wave inference image about 20 seconds, and the proposed method 76 milliseconds.

been significantly improved. The well-known basic concept is for each pixel to use a convolution of a well-chosen noise signal along the flow with a convolution kernel, such as a Gaussian. This operation creates a coherent pattern along the flow while producing little coherence perpendicular to the flow.

An inherent property of this approach is that the result exposes some of the features of the underlying noise. In some works it has been successfully utilized for transferring additional information by means of noise spatial frequency. However, the randomness, introduced by the input, does not serve any particular visualization purpose. The questions we want to address in this chapter are whether a noticeable amount of noise is a necessary ingredient for dense flow visualization (DFV) and whether there are options better than altering the noise injection scheme to control such visualization characteristics as spatial frequency and contrast.

The approach described in Chapter 4 suggests that the explicit presence of the input noise can be avoided, although at a high computational cost. In this chapter we consider a more intuitive and perception-motivated approach, based on the popular LIC algorithm, offering better frequency control with efficiency comparable to LIC. We call this new method *Orthogonally Gabor-Enhanced Repetitive LIC* or OGR LIC for short. The idea is to consider the traditional LIC from the point of view of integral operators in the local flow-aligned coordinates, which allows us to extend it with Gabor kernels, understanding the implications in the frequency domain. Then, we take spatial frequency-oriented visualization a step further, demonstrating how single-frequency images can constitute a basis for richer adaptive multi-frequency flow visualizations. In summary, the key features of the presented method are:

- dense noise-insensitive flow visualization
- adaptive spatial frequency control
- perceptional motivation

- straightforward mathematical formulation
- straightforward implementation, reusing the LIC procedure
- run-time efficiency
- few, intuitive parameters

5.2 Iterating Line Integral Convolution

The method described in this chapter falls into the category of dense vector field visualization, one of four categories of flow visualization: *direct*, *texture-based*, *geometric*, and *feature-based methods* [86, 61] or one of five according to the state-of-the-art report of Salzbrunn et al. [95], who also include *partition-based* techniques. For details on methods other than dense flow visualization techniques, we refer the interested reader to one of these state-of-the-art reports. In the remainder of this section, we focus on dense flow visualization.

The first method to demonstrate the power of dense flow visualization was the pioneering work on Spot Noise by Van Wijk [131]. Only two years later, Cabral and Leedom introduced the Line Integral Convolution (LIC) method [9]. Since then, texture-based methods have proven to be effective flow analysis tools in a wide variety of fields. Forssell [28] extended the method to arbitrary surfaces and Rezk-Salama et al. [91] demonstrated its use in 3D. Stalling and Hege [105], as well as Stalling in his thesis [104], performed a thorough analysis of the influence of various LIC parameters on the signal-processing level. Weiskopf [127] demonstrated a fast GPU implementation.

An important aspect of the research on LIC is iterative application, proposed by Okada and Lane [78] for improving the visual quality of the method. This idea was later revised by Weiskopf [128] and Hlawatsch et al. [38], mainly in the context of performance optimization. The discussed technique utilizes intense iterative low- and high-pass filtering to suppress the features of the original noise input completely, thus resulting in a visualization with controlled frequency characteristics.

As the above review suggests, a significant amount of work in dense flow visualization has been focused on extensions to the LIC method. There are also a few methods based on different algorithms. Examples include Taponecco and Alexa's work [111] on vector field visualization with Markov random fields and Van Wijk's image-based flow visualization method [130].

5.3 Multi-frequency Noise for Dense Flow Visualization

There have been efforts [51] [106] to adaptively control frequency in DFV images with multi-frequency noise for various practical purposes, including extra scalar value mapping, level of detail control, and uncertainty visualization.

The fact that the result of LIC filtering depends not only on the input flow, but also on the input texture, was recognized and used for visualization purposes soon after the introduction of the original LIC algorithm. Kiu and Banks [51] suggested using noise

with locally varying spatial frequency to enhance perception by mapping noise frequency to the flow velocity magnitude.

This approach was then utilized for LIC on surfaces [106], to compensate for texture distortions due to a varying level of detail. The original schema using LIC with multi-frequency noise was further revised by Urness et al. [120], who suggested the *texture stitching* technique, which combines the output of LIC applied to several separate noise textures featuring different noise scales. That is, the spatially varying frequency is achieved *after* LIC, not *before*, as in the approach of Kiu and Banks. We use a variation of this idea with the presented technique in Section 5.7.2.

The spatial frequency control is a very important topic in unsteady flow visualization. The main issue here is the low-pass filtering that consistently destroys high-frequency components of the advected texture over time. Up to this point, the efforts have been concentrated on the preservation of the spectrum of input texture [137, 76]. An interesting two-phase scheme has been recently proposed withing the dynamic LIC approach [107]. It consists of generating for each frame the time-coherent high-frequent noise texture by tracing particles along the streamline evolution trajectories (which is given as an independent flow) in the first stage, followed by ordinary LIC afterwards. In the context of the presented method we discuss a constructive way not just to preserve, but to enforce the desirable spatial frequencies in the resulting images.

The recent applications of multi-frequency noise are connected to the rising interest in uncertain flows [5, 79]. The noise frequency here is in inverse proportion to the uncertainty, resulting in blurred regions in visualization where high error is expected.

One common property of the above methods is that the varying noise frequency is obtained by smoothing a high-frequency texture, thus effectively achieving varying noise *scale*, but not concerning its *orientation*. Recent work on Gabor noise [58] demonstrates how a noise texture with specific spatial frequency and orientation can be generated, with the means of Gabor functions in the context of computer graphics applications.

In the next section we briefly discuss the role of Gabor functions in visualization and human perception, motivating their choice for frequency-control filtering in the presented method.

5.4 Gabor Filters in Visualization and Perception

The extensive literature on human perception suggests that the human visual system processes spatial information by means of arrays of neurons that can be modeled with Gabor functions. This model of the visual cortex edge detection mechanisms accounts for a large variety of experimental results [16, 48, 67, 84, 77]. An important feature of the Gabor functions is that they, having a localized distribution (Gaussian shape), jointly optimize the sensitivity in frequency and spatial domains, which is a property believed to hold for the human visual system as well [23]. The idea is that human vision examines frequency and orientation on a local basis, by a windowed rather than by a global Fourier transform.

Ware and Knight [123], in their pioneering work on texture-based visualization using the Gabor model, formulate three fundamental dimensions of texture: frequency or size, contrast, and orientation. These dimensions can be used to transfer information and are measured by 2D Gabor filters.

In flow visualization, the properties of Gabor functions for measuring local frequency and orientation were exploited in the work of Jänicke et al. [43], which introduces a quality metric for DFV images. This metric relies on the ability of Gabor filters to extract the perceived flow direction from the visualization image.

The conventional 2D Gabor filters are products of the Gaussian filtering in one dimension and 1D Gabor filtering in the other direction. In this chapter we utilize 1D Gabor filters to enforce a particular frequency in the direction orthogonal to the flow, combined with ordinary LIC Gaussian filtering along the flow, resulting in a 2D filter, analogous to the 2D Gabor filters in perception and visualization.

5.5 Flow Visualization With Explicit Frequency Control

The technique presented in this chapter can be seen as an extension of LIC and of the method presented in Chapter 4. The conceptual difference to it is that given a an integral curve per pixel, instead of computing a pairwise heuristic distance between pixels and thereby compute wave interference patterns we iteratively compute the line integral convolution with a noise signal with specific filters.

Analogous to this technique, we use the complex exponentials and iterated filtering as basic building blocks of the method. However, employing a completely different process allows us to achieve significant improvement over its predecessor. In particular, OGR LIC differs from the previously presented method in the following features:

- It admits an intuitive representation in terms of the widely adopted LIC procedure by plugging in appropriate kernels. This avoids the introduction of additional heuristics.
- The choice of filtering kernels has a solid motivation in perceptional studies and is related to flow visualization metrics.
- The filtered images feature less deviation from the original flow, requiring a fraction of the computational cost of the predecessor (see Figure 5.1).

Moreover, we discuss how the frequency-filtered output of the presented method can be further used to achieve flow visualizations with multiple controlled frequencies, including locally adaptive frequency, and address the issues in the perception of high contrast.

5.6 Details of the Visualization Technique

In this chapter a technique for dense flow visualization is suggested with finer spatial frequency management than is feasible using noise-based approaches. The common feature of many available dense flow visualization methods can be described as low image variation in the direction of the flow and at the same time high variation in the orthogonal direction. This perception-related principle is also acknowledged by existing DFV quality metrics [69, 43].

The source of variation in both directions in LIC output is the input texture, smoothed in one direction by the convolution kernel. Thus, for example, changing the scale of the

input noise (for the overview of methods using multi-frequency noise see Section 5.3) affects both directions in the output, whereas an independent control is desirable. Moreover, altering a noise generation scheme admits only indirect spatial frequency control, which is non-linearly affected by the posterior filtering. In this work we address the spatial frequencies directly in the visualization image.

One novel idea of this work is, to the best of the author's knowledge, to explicitly consider and manipulate the frequency characteristics of the visualization image in *both* directions: along and orthogonal to the flow. We avoid the variation in the direction of the flow originating from noise, yet maintain high contrast and variation in the second direction (in static images) or on the contrary enforce a periodic wave pattern along the flow (in the animations as in Section 5.7.3).

Gabor filters are the natural choice for implementation of this idea. These filters constitute the building blocks of the short-term Fourier transform and have a long-standing perceptual motivation (see Section 5.4). This approach allows us to control all three dimensions of the information visualization texture model, suggested by Ware and Knight [123]—contrast, frequency and direction—avoiding the randomness introduced by noise injection.

The discussed approach is based on the commonly adopted LIC scheme and in order to explain its technical side we first need to describe the generic framework we use for analysis of LIC in terms of directional frequencies. We take a view at LIC as an integral operator in coordinates tangential and normal to the flow and explore its spectrum in these coordinates. Then, we extend traditional LIC adding Gabor filtering orthogonal to the flow direction and consider the resulting spectrum, thus obtaining an analog of 2D Gabor filter. We demonstrate, that high-contrast frequency-bounded images can be obtained with several iterations of these kernels combined with restoration of the energy lost in each filtering step. In Section 5.7 we demonstrate how a combination of this images can result in interesting and novel visualizations and discuss possible issues.

5.6.1 LIC as an Integral Operator in R^2

We consider LIC as an image filter applied to some noise image, staying in the continuous setting throughout our analysis for simplicity. Formally, a continuous formulation of an image filtering with a LIC kernel can be described as an integral transform of the image $f(x,y)$ into the image $g(x,y)$ with $(x,y) \in \Omega \subset R^2$. Such a transform is formulated in the most general form with Equation 5.1.

$$g(x,y) = \int_{\Omega} f(x',y')K(x,y,x',y')dx'dy' \tag{5.1}$$

where $K(x,y,x',y')$ is some operator kernel that is specific to the operator.

For our purpose it is convenient to define the LIC operator in curvilinear coordinates, instead of the more commonly used Cartesian coordinates. We introduce the curvilinear coordinates $\xi = u(x,y)$ in the direction tangential to the flow at every given point (x,y) and $\eta = v(x,y)$ orthogonal to the flow at this point. The new coordinates then are defined in the domain $\Omega' \subset R^2$ Note that we use this coordinate system merely for convenience of analysis, in the actual implementation the curvilinear grids are not constructed. Further,

we deal with the integral transform in Equation 5.1 in the following form:

$$g(x,y) = \int_{\Omega'} f(\xi,\eta) K_{lic}(x,y,\xi,\eta) d\xi d\eta \tag{5.2}$$

where the LIC kernel K_{lic} already accounts for the coordinate transformation. The traditional LIC operator based on a Gaussian kernel G_{σ^2} with standard deviation σ can be described by plugging the K_{lic} kernel into Equation 5.2. K_{lic} is specified by Equation 5.3 and Equation 5.4.

$$K_{lic}(x,y,\xi,\eta) = G_{\sigma^2}(u(x,y)-\xi)\delta(v(x,y)-\eta) \tag{5.3}$$

or, denoting $\eta_0 = v(x,y)$ and $\xi_0 = u(x,y)$

$$K_{lic}(x,y,\xi,\eta) = G_{\sigma^2}(\xi_0-\xi)\delta(\eta-\eta_0) \tag{5.4}$$

In the previous two equations, $\delta(t)$ is a Dirac delta pulse. Now, Equation 5.2 can be written as a separable convolution:

$$g(\xi_0,\eta_0) = \int_{\eta_1}^{\eta_2}\int_{\xi_1}^{\xi_2} f(\xi,\eta) G_{\sigma^2}(\xi_0-\xi)\delta(\eta-\eta_0) d\xi d\eta \tag{5.5}$$

or, in a shorter notation:

$$g(\xi_0,\eta_0) = [[f(\xi,\eta) * G_{\sigma^2}(\xi)](\xi_0,\eta) * \delta(\eta)](\eta_0,\xi_0) \tag{5.6}$$

Here the star inside the square brackets denotes a convolution. Thus, the LIC operator can be seen as a separable convolution in the coordinates tangential and orthogonal to the flow (up to the coordinate scale). In other words, the 2D filtering can be reduced to sequential application of two 1D filters. This observation allows us to intuit the effect of this operator in the frequency domain.

Application of the Fourier transform $\mathscr{F}$ to the operator provides a comprehensive representation of its result in terms of the amount of energy contained in frequencies l along and q across the flow. The Fourier transform $\mathscr{F}$ for the LIC-filtered image g in the coordinates l and q is:

$$\begin{aligned} &\mathscr{F}\{g\}(l,q) = \\ &= \mathscr{F}\{f(\xi,\eta)\}(l,q) \cdot \mathscr{F}\{G_{\sigma^2}(\xi)\}(l) \cdot \mathscr{F}\{\delta(\eta)\}(q) \\ &= \mathscr{F}\{f(\xi,\eta)\}(l,q) \cdot G_{\sigma^{-2}}(l) \end{aligned} \tag{5.7}$$

since $\mathscr{F}\{\delta(\eta)\} = 1$. Here and later we consider the Gaussian kernels up to the normalization factors.

The last equation basically states that LIC performs a low-pass filtering in the direction tangential to the flow with Gaussian $G_{\sigma^{-2}}$ independently of the frequency in the orthogonal direction, illustrated in Figure 5.2a.

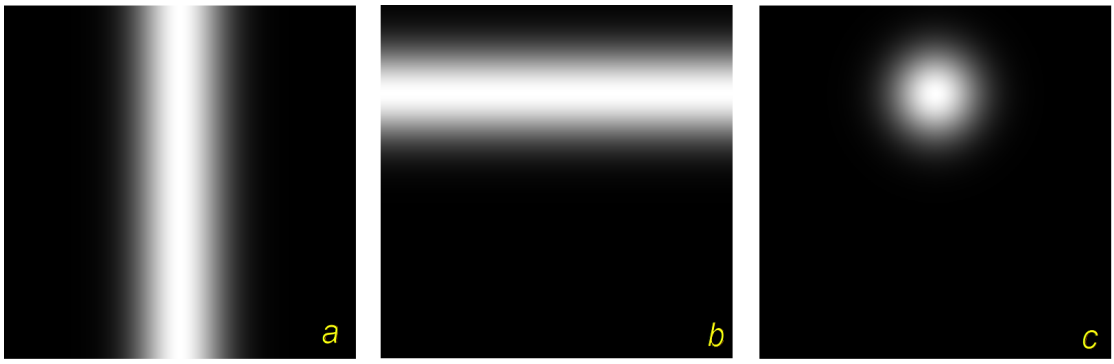

FIGURE 5.2: A sketch of the Fourier spectrum of the LIC kernel (a), Gabor kernel (b), and the convolution in the spatial domain of LIC and Gabor filters (c). The horizontal direction denotes the frequency in the direction tangential to the vector field corresponding to coordinate ξ. The vertical direction denotes the frequency orthogonal to the vector field corresponding to coordinate η. The LIC kernel spectrum is a Gaussian centered around zero frequency changing only in ξ independent of η. The Gabor kernel spectrum is a Gaussian centered around a certain (user-selected) frequency changing only in η independent of ξ. The spectrum of the convolution of both filters in the spatial domain is a multiplication of their spectra.

5.6.2 Extending LIC With Gabor Filtering

As we have seen, Line Integral Convolution can be generalized as a basic framework composed of two filters: in the direction of the flow and orthogonal to it. Furthermore, our insight is that the frequency-domain implications of the particular choice of these filters can guide the visualization design. The original LIC performs one-dimensional smoothing in the local flow-aligned direction, resulting in an image spectrum, centered around zero frequency. We demonstrate how an image, centered around *any* specific spatial frequency, can be obtained with iterative Gabor filtering. This process, which we will call OGR LIC, is a contribution to the toolbox of visualization methods.

Apart from their strong perceptional motivation (see Section 5.4), the Gabor functions are the natural choice for picking up a particular frequency due to their unique spectral characteristics: the Fourier transform of a Gabor filter is a Gaussian, centered around the chosen frequency. Due to this property, Gabor functions are suitable for local-frequency analysis, forming the basis for the windowed Fourier transform.

Formally, a one-dimensional Gabor filter with the wavelength λ and angle θ is defined as in Equation 5.8:

$$W(t) = G_{\sigma^2}(t)e^{i(\frac{2\pi}{\lambda}t+\theta)} \tag{5.8}$$

We set $\theta = 0$ since the phase is constant along the stream line.

Using the notation of the operator framework, presented in the previous section, the Gabor kernel orthogonal to the flow can be defined as:

$$K_{gb}(x,y,\xi,\eta) = W(\eta_0 - \eta)\delta(\xi - \xi_0) \tag{5.9}$$

Generally a 2D Gabor filter is a combination of 1D Gabor filter in one direction and a Gaussian filter in the orthogonal direction. The filtering operator we suggest consists of the LIC (Gaussian)smoothing along the flow and the Gabor filtering orthogonal to the flow. It is described by Equation 5.10:

$$g = f * K_{lic} * K_{gb} \tag{5.10}$$

Or in the frequency domain:

$$\begin{aligned} &\mathscr{F}\{g\}(l,q) = \\ &= \mathscr{F}\{f\}(l,q) \cdot G_{\sigma_1^{-2}}(l) \cdot [G_{\sigma_2^{-2}}(q) * \delta(q - \frac{1}{\lambda})](q) \\ &= \mathscr{F}\{f\}(l,q) \cdot G_{\sigma_1^{-2}}(l) \cdot G_{\sigma_2^{-2}}(q - \frac{1}{\lambda}) \end{aligned} \tag{5.11}$$

since $\mathscr{F}\{G_{\sigma_2^2}(t) \cdot e^{i\frac{2\pi}{\lambda}t}\} = G_{\sigma_2^{-2}}(q) * \delta(q - \frac{1}{\lambda})$ and $h(q) * \delta(q - z) = h(q - z)$ for any function $h(q)$. Here σ_1 and σ_2 are the standard deviations of the LIC and Gabor filters. Thus, our new filter performs a low-pass filtering in the direction of the flow *and* band pass filtering in the orthogonal direction, centered around frequency $\frac{1}{\lambda}$. Visually this filtering corresponds to alternating black and white lines of thickness λ. We describe the resulting images as being centered around a certain frequency or frequency-bounded.

The spectrum of the filters is illustrated in Figure 5.2b and Figure 5.2c. The same plot as in Figure 5.2c represent the spectrum of a 2D Gabor filter used the texture model of Ware and Knight [123] and other common applications except for the different coordinate system (Cartesian instead of flow aligned in our model). In fact, for a locally constant flow these types of filters are identical.

5.6.3 Implementation Notes

Using the framework described above, we suggest the following filtering process:

Starting from a random input image, we apply the discussed filters in two passes. Due to the associative nature of convolution, the kernels in Equation 5.10 can be applied in a sequence one after the other. We first run an ordinary LIC shader (i.e., a convolution with a Gaussian). Then, we rotate the vector field by $\frac{\pi}{2}$ and run the same shader on the output of the previous pass with a Gabor kernel instead of a Gaussian.

To achieve a consistent smooth line pattern, several iterations of the described procedure (normally 15 - 20) are sufficient. However, it is important to take into account the contrast normalization between iterations. This problem can be described as energy (contrast) loss in the processed image due to filtering. In the frequency domain, the filter multiplies all frequencies with a Gaussian, thus decreasing the energy of the resulting image.

There are several ways to restore the energy in the original range. One can rescale the image values on each iteration, use histogram equalization transform, or even apply a binary threshold. We believe the best contrast normalization effect is achieved with the following method. We start from a random complex image z normalized such that $|z| = 1$. We perform our computation in complex numbers, and after each three iterations of filtering divide the resulting image z' by $|z'|$ in each pixel. This simple operation

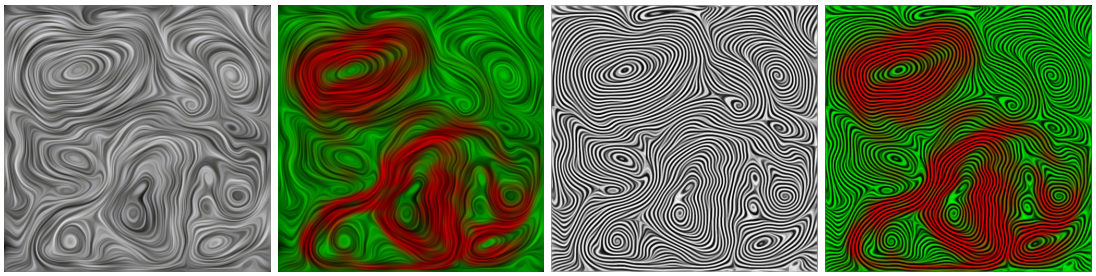

FIGURE 5.3: Plain dense flow visualizations and velocity color-coded versions. LIC is shown on the left and the presented method on the right. Especially with color coding, the higher contrast of the new method makes the image much clearer, which becomes even more apparent in print.

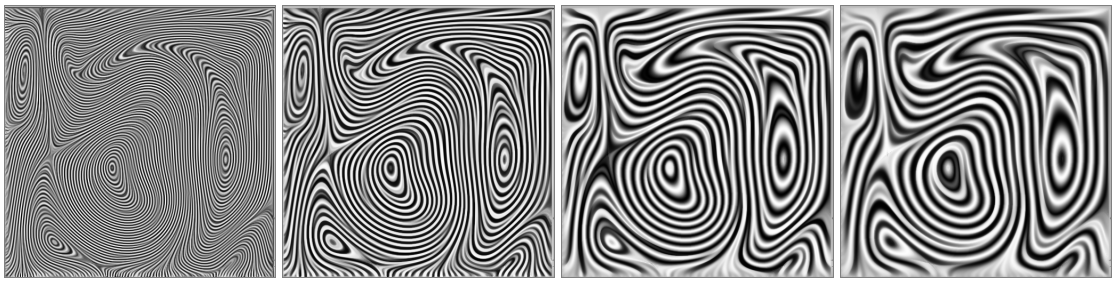

FIGURE 5.4: A single vector field, visualized with OGR LIC using decreasing frequency from left to right, 20-30 iterations of the filter.

produces an image with the same total energy as $|z|$. The result is then either the real or imaginary part of the complex image.

5.7 Method Applications and Modifications

In this section we first illustrate the visual effect of OGR LIC method, applying it to a number of representative vector fields. Then, we discuss possible approaches how the textures synthesized by the presented method could be further used as components for effective flow visualizations.

The example images and vector fields are sampled at a 512^2 grid and are computed in 20 iterations of Gaussian and Gabor filters. The first set of images in Figure 5.5 demonstrates the method in a number of vector fields. For comparison, we paired each image with a LIC visualization of the same flow field. The second set of images in Figure 5.4 depicts the same vector field visualized with different frequencies. The third set of images in Figure 5.16 shows the method applied to the same flow field a number of times with varying noise to demonstrate the invariance of the desired frequency with respect to the input noise. In the rightmost image of this figure, we used uniform gray with only 10 arbitrarily placed white pixels as input. Even in this extreme case, OGR LIC converges to a result similar to the other images, although it does require more iterations (100 instead of 20). Figure 5.3 demonstrates the distortion of color brightness, due to noise in LIC images combined with color-coding, compared to OGR LIC output.

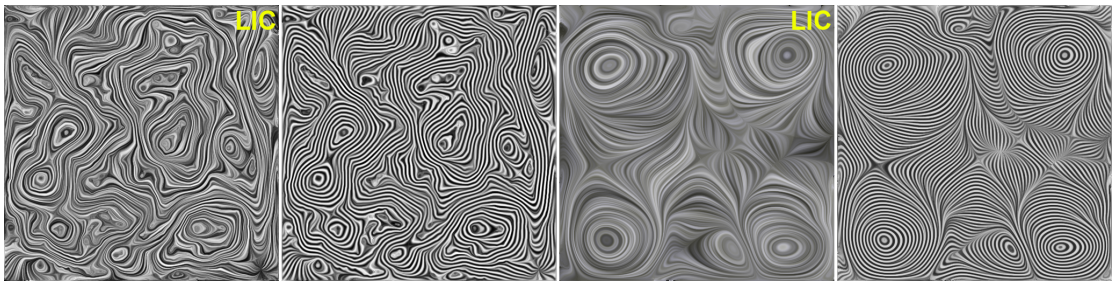

FIGURE 5.5: OGR LIC applied to a number of vector fields. For comparison, each image is paired with a LIC image of the same field using the same noise input.

FIGURE 5.6: A vector field $(1, cos(x))$ visualized with high-frequency OGR LIC with high amplitude at the left and low amplitude at the right. Note that frequency characteristics of the images are the same, but the second image is more comfortable to look at.

5.7.1 Contrast In Bounded-Frequency Images

One important aspect we need to address concerning the images with one central frequency is the contrast perception. It can be observed that high spatial frequencies combined with high contrast have a disturbing effect, especially when being looked at for a long time. The left image in Figure 5.6 demonstrates an extreme case of this problem. In this respect, the original LIC images that have a wide frequency spectrum and relatively low contrast are more comfortable to look at.

Before we identify the source of additional visual stress arising from these images and suggest a remedy, we would like to note that high-contrast bounded-frequency images are not themselves necessarily the target visualization, but rather are a useful building block for producing novel effective visualizations. In this and the following sections, we discuss how to benefit from the advantages of frequency control and at the same time avoid the downsides of the high-frequency, high-contrast visualization.

Importantly, several decades of contrast perception studies [92] suggest that the

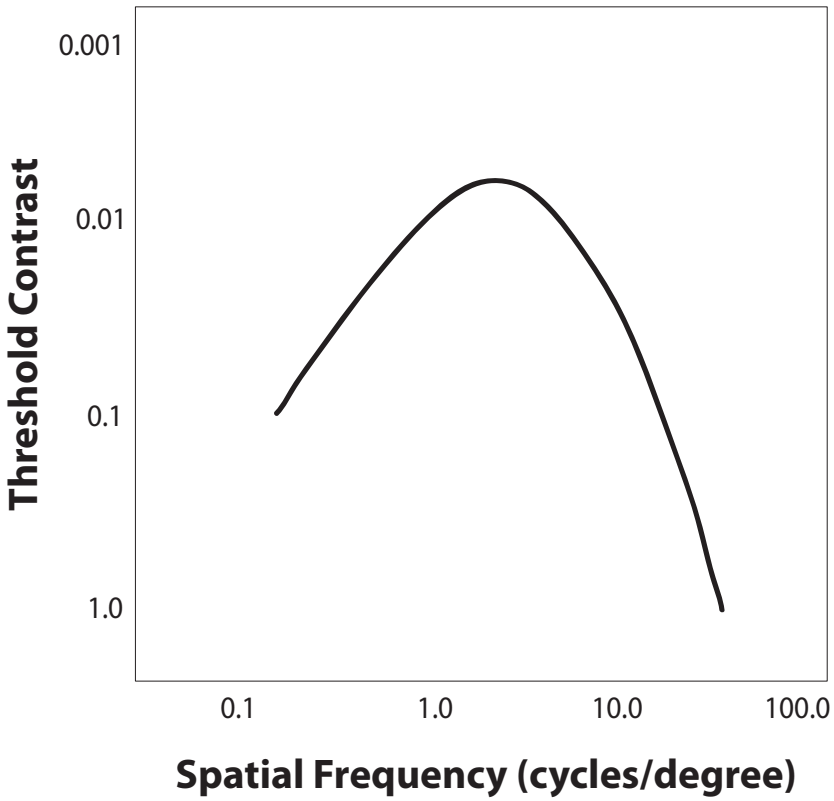

FIGURE 5.7: Human visual sensitivity of spatial patterns [123], varying with amplitude (contrast) and spatial frequency. Note the fall-off at the low and high ends of the curve. The reduction of sensitivity in low-frequency range is a more common issue due to the inability of finite-resolution displays to reproduce high frequencies beyond the Nyquist limit.

human contrast sensitivity (the reciprocal of the threshold contrast) is a function of the frequency of the visual signal. Figure 5.7 demonstrates the sensitivity to different spatial frequencies at the same amplitude level, suggesting that the sensitivity to a particular image can be decreased by either decreasing or increasing the spatial frequency. However, the latter option is hardly feasible with finite-resolution displays due to the Nyquist limit [33]. The high-frequency OGR LIC images, falling in the vicinity of the peak of the curve in Figure 5.7, thus require much less amplitude to achieve the same ratio to threshold contrast than the lower-frequency images (e.g., generated by traditional LIC), represented by the left end of the curve.

One option to avoid the disturbing effect of the high-frequency high-contrast images is to rescale the gray values to a smaller range, effectively decreasing the amplitudes closer to the threshold contrast range. Consider the resulting change in perception in Figure 5.6. In the next section we suggest a more practical solution, which combines contrast reduction with additional frequency injection, leading to adaptive frequency control.

FIGURE 5.8: Two flow visualization images centered around different frequencies are averaged to obtain the image at the right. The result features the frequencies of both input images.

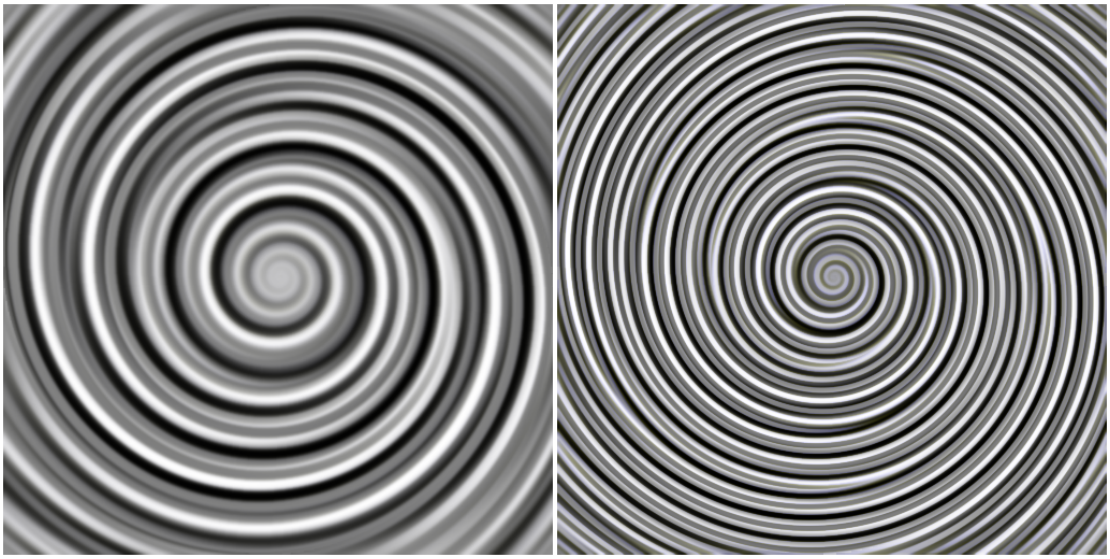

FIGURE 5.9: The vector field around a spiral vortex, visualized with different combinations of frequencies (period in pixels $T = 10, 20$ at the left and $T = 5, 10$ at the right)

5.7.2 Frequency Combination

Apart from just reducing the amplitude of the images, we suggest a comparably simple, yet much more powerful technique: blending several frequency-bounded images. This trick both reduces the local contrast and enriches the visualization. Taking advantage of the fact that the frequency spectrum changes linearly with the change of input images, we average images, centered around different (usually divisible) frequencies. Figure 5.8 and Figure 5.9 illustrate this idea. An important additional benefit of the resulting visualizations compared to the single-frequency centric images is that line splittings, which are noticeable in the "pure" frequency images, are almost completely indistinguishable in the "mixed" frequency version. In particular this results in reduced average angular error (AAE), as the evaluation results in Section 5.8 indicate.

The local frequency was identified to be one of the fundamental information-transferring texture dimensions [123]. There have been efforts to adaptively control frequency in LIC images with multi-frequency noise for various applications, including uncertain flow visualization and additional parameter mapping (see Section 5.3).

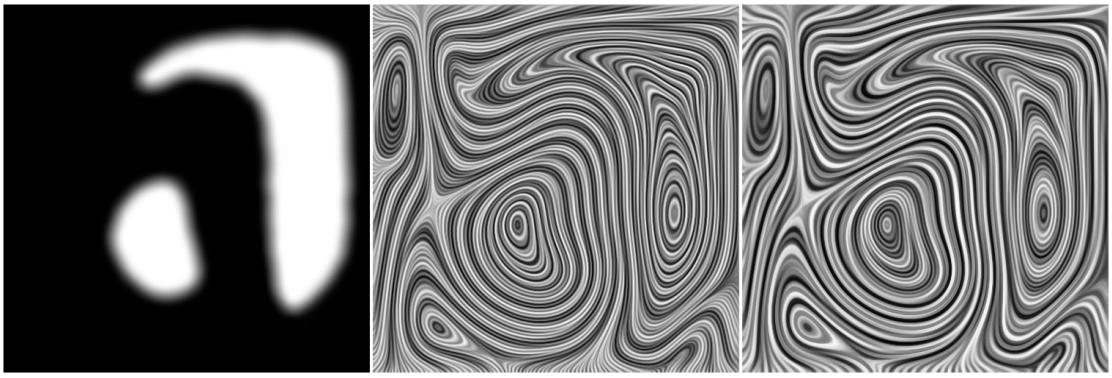

FIGURE 5.10: Left: mask image, highlighting the area of interest. Middle and right: two visualizations featuring different combinations of frequencies for the same flow. Note the masked region frequency differs from the rest of the visualization.

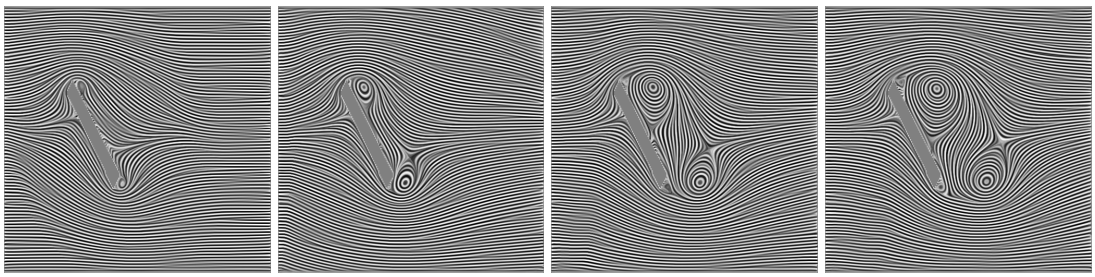

FIGURE 5.11: A number of frames from a Kármán vortex street simulation. The image has to be viewed in full resolution, since subsampling of the high-frequent details present in the visualization leads to aliasing effects.

The presented approach, compared to noise-based solutions, avoids introducing high frequencies along the flow and allows more control over frequencies in the orthogonal direction. The locally adaptive frequency can be achieved easily using a mask when blending frequency-oriented images, as demonstrated in Figure 5.10. The mask provides the weight of a certain frequency image in each pixel. That is, first using OGR LIC, we compute images centered around different frequencies and then form one visualization image as a weighted sum of these input images.

5.7.3 Animation and Unsteady Flow

Here we consider two scenarios that describe how the presented technique can be used for flow animation. First of all, we would like to note that the low-pass filtering in the flow direction is not necessarily a prerequisite for the visualization. Instead, using a Gabor filter along the flow can serve visualization purposes. The original Cabral and Leedom paper [9] mentions that LIC can be animated by shifting the phase of a periodic kernel.

A frequency-oriented view of these images allows us to combine images with specific frequencies *both* along and across the flow. The animation of checkerboard-like flowing patterns can be generated by incrementing the phase of the waves in the direction of the

FIGURE 5.12: Left: OGR LIC image with period 12 pixels across the flow. Middle: OGR LIC image with period 24 pixels along the flow. Right: One animation frame, obtained by blending the first two images. The subsequent frames are computed by incrementing the phase of the image in the middle.

flow. The achieved visual effect is similar to the recent geometric approach to streamline animation of Yeh et al. [136]. Figure 5.12 illustrates how one frame of such an animation can be computed. In the frequency-oriented images filtered only along the flow, the sharp edges aligned with the flow allow to distinguish separate flow-aligned bands visualizing the flow structure. At the same time the presence of these edges can be confusing, since they do not represent any specific flow structures and thus is a drawback of this method.

Another visualization scenario is the application of the OGR LIC method to unsteady flow. For each animation frame, the target visualizations are instantaneous streamlines. This approach, though not free from limitations has found its applications in several problem domains. Among the most advanced implementations is the work of Jobard and Lefer [46]. The method has particular value for magnetic field visualization, where one proposed texture-based implementation is the dynamic LIC of Sundquist [107].

The discussed approach is not subject to the loss of high-frequent components of the visualization with time, since the resulting image is not directly dependent on the input texture. For this reason we do not need to regenerate the input noise for each frame, as in dynamic LIC. However, the independent computation of animation frames creates a disturbing flickering effect in the vortex cores along their movement.

In order to achieve the temporal consistency between frames, we employ basic LIC filtering along the streamline motion field. In the work of Sundquist this motion field is introduced a second vector field that describes the evolution of streamlines. It can be available as an independent input for some applications. We suggest a scheme to construct this field using basic vortex tracking. For each frame we compute a feature map, corresponding to the commonly used λ_2 vortex criteria [44] to capture the vortex region's structure. In the next step we apply the optical flow algorithm of Horn and Schunk [39] to estimate the direction of the vortex movement in space-time. We apply line integral convolution with a Gaussian kernel along this flow (across frames) in addition to the Gaussian smoothing along the original flow and the Gabor filtering orthogonal to flow (within frame). A limitation of the current implementation is that still a significant amount of oscillation persists in higher-frequency visualization in the regions close to the vortex cores, which can be distractive.

Experimental observation is that the most coherent visualizations are achieved using low Gabor frequencies (below $f_{Nyquist}$ / 8) and the kernel lengths chosen in proportion

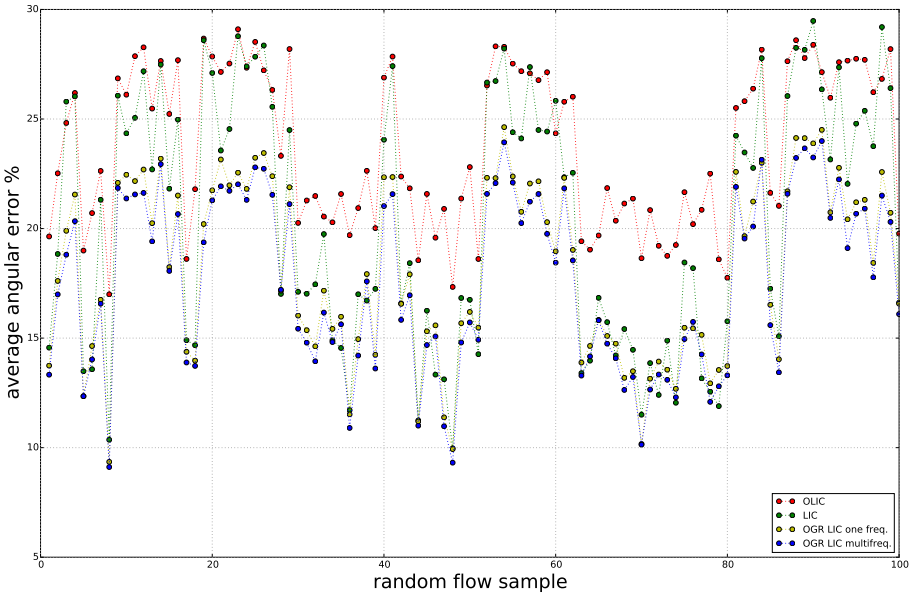

FIGURE 5.13: The AAE evaluation results for different visualizations for each of the 100 random flows.

2:1:2 for filtering along the flow, orthogonal to the flow, and across the frames. The obtained results show time-coherence of streamline and vortex colors between subsequent frames.

5.8 Quality Evaluation

In this section we conduct a statistical evaluation of the OGR LIC method using visual reconstructability of the underlying flow as a criterion. We compare LIC, OLIC and two variations of the method to each other using this metric.

The effect of frequency control is evident from the visualization images; however the accuracy of the underlying flow representation still requires verification. Our goal is to position the work against prior art and in particular compare it with LIC as it is naturally the closest analog. Our expectation is that with OGR LIC it is possible to produce at least as accurate flow representations using only a limited frequency bandwidth.

There has been a great deal of research on the evaluation of general flow visualization effectiveness including user studies [60, 59, 82] and mathematical models and metrics [22, 11, 132, 26, 69].

A viable approach would be a user study, which, however, comes at the price that it is tied to a specific visualization purpose (or task conducted by participants) and influenced by individual subject preferences. It is also harder than automated evaluation to reproduce on a large scale. As we do not have a specific user task in mind we are concerned with the introduced directional error compared to closely related **texture-based** methods (LIC, OLIC).

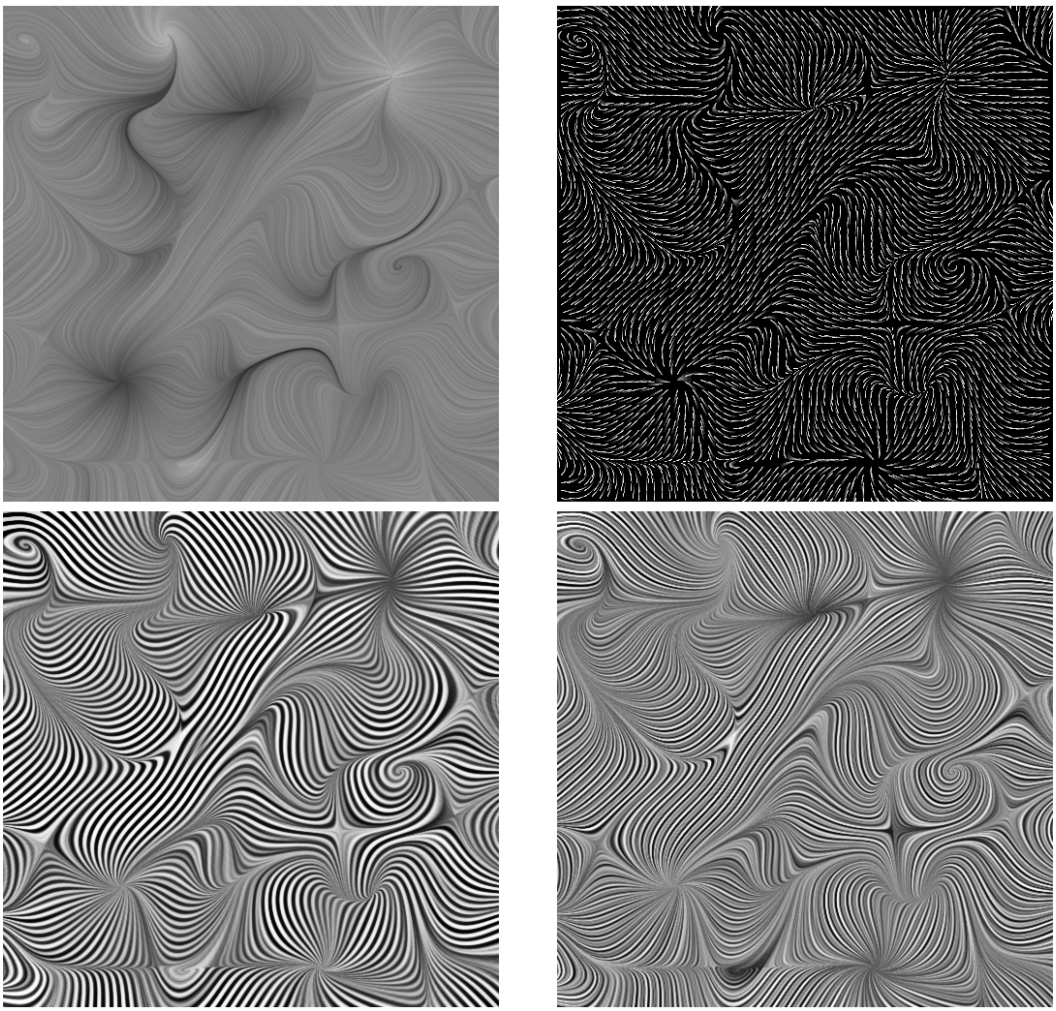

FIGURE 5.14: Different visualizations of one of the random flows used for evaluation. Top left - LIC, top right - OLIC, bottom left - OGR LIC with fixed frequency, bottom right - OGR LIC with combination of frequencies.

For these reasons we decided in favor of a combination of quality metric computations with the methods of statistical inference in order to make general statements about the method.

An important limitation of such evaluation compared to perceptional experiment is that such a purely technical measure can give an insight on how much original directional information is preserved in the image, but not how accessible this information to the human. So, the output of the evaluation is then the assessment of **the accuracy** of representation, not the ultimate **usefulness** of the visualization, which might be addressed in a subsequent user study.

Other limitation of the presented image-based approach to evaluation is that a great body of research on geometric 2D flow visualization is left behind. In particular, several prominent even streamline placement methods (e.g., [13], [45], [135]) allow to achieve similar visualizations. With certain additional assumptions (on flow direction interpolation [43]), the error metric we employ can be generalized for comparison image-based to geometric methods. Combined with streamline rendering adjusted in such a way that thickness of streamlines is close to the distance between streamlines, the obtained streamline visualizations have been reported to yield a better score than LIC under this metric [43]. Although these methods also have other advantages over texture-based flow visualization (e.g., better control over magnitude), a fair comparison of geometric approach to texture-based approach is beyond the scope of this work, which is focused on images synthesis.

We suggest a more focused evaluation setup, comparing the error introduced in 3 specific implementations within the same LIC filtering framework (LIC, OLIC, OGR LIC). As a quality score we choose a metric suggested by Jänicke et al. [43]. The main advantage of this metric for our purpose is that it is not highly sensitive to contrast (as opposed to [69] for instance) and we do not want high contrast of OGR LIC by itself to give a preference to the method discussed in this chapter.

The approach of Jänicke et al. consists in extracting the flow direction from the visualization image using a standard 2D Gabor filter bank (which only difference to our model in that is not aligned with the flow). The extracted direction is then compared against the actual flow direction by the computation of the average angular error (AAE). Thus, the ultimate score is the measure of similarity between the original flow and the flow reconstructed from visualization.

As a minor technical simplification of the original algorithm, we consider the average angular error among all pixels, instead of introducing a threshold on the minimal considered filter response and further interpolation of the sparse values. In other words we avoid making a decision about the relevance of different pixels to the whole visualization.

With this quality evaluation method at our disposal, we do statistical inference for the mean value of the sample of the randomly generated vector fields. Our statistical simulation is as follows. We generate 100 random flows, using only a uniformly distributed flow direction, since the flow magnitude does not affect either the presented method or LIC. The resulting flows are chaotic, and in order to ensure that flow features are representable, we choose the visualization image resolution 16 times the flow resolution (32×32). The image resolution is 512×512 and the LIC half-kernel length is 160, as the observation of Jänicke et al. is that the lowest error is achieved using large kernels. The experiments show that the average angular error is within range of the previously

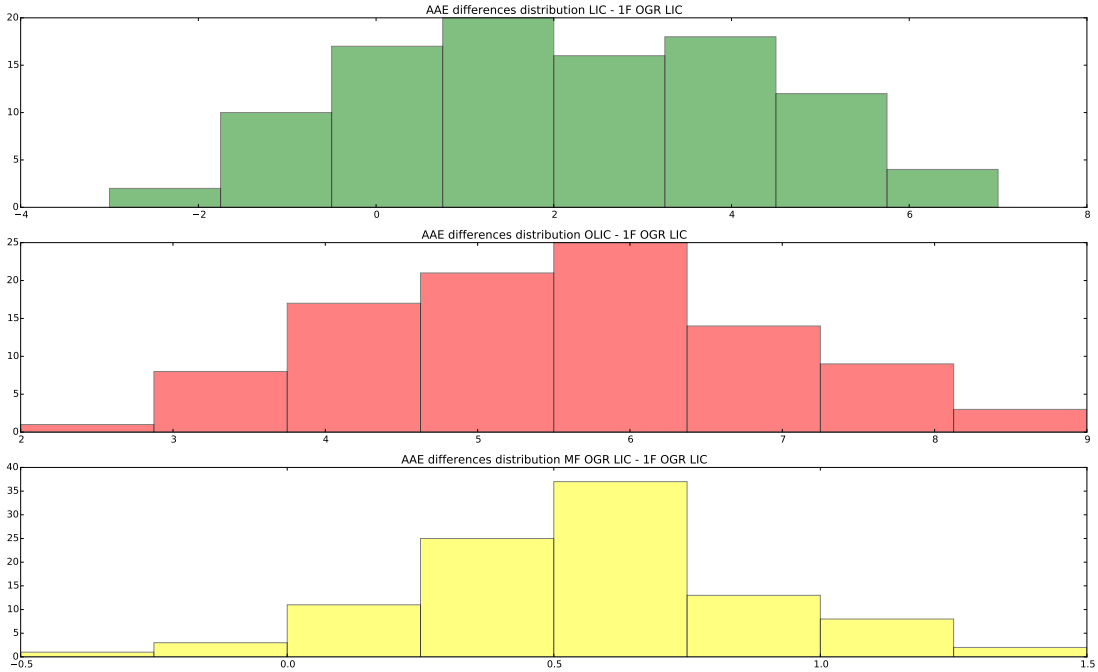

FIGURE 5.15: The histograms of pair-wise differences in AAE between pairs of visualizations for different random flows used for evaluation. As samples are independent, the pairwise distances follow the normal distribution (centered at the mean difference)

reported error 10%–30%. For randomly generated flows due to their higher turbulence, the observed error is larger (below 20%) than for the flows with simple topology (around 10%).

For each flow we compute four visualizations: LIC, OLIC, OGR LIC with single frequency (period 8 pixels), and OGR LIC with two frequencies combined (periods 4 and 6). Next we estimate the quality score as suggested by Jänicke, computing the average angular error between the flow and each of the three visualizations.

We conduct a paired differences z-test for the the AAE estimates within the following scenarios: LIC vs. single-frequency OGR LIC images, OLIC vs. single-frequency OGR LIC images, and single-frequency vs. multi-frequency visualization assuming as a null hypothesis that the estimated means are equal in each pair.

That is, for each test the statistic of interest D is the pair-wise difference in AAE for different visualizations, the estimate of the expected value of D is the empirical mean of the observed differences. Given (by construction) independence of the random samples and a sufficient number of samples (n=100), the central limit theorem suggests approximately a normal distribution of the test statistics D (around zero if the null hypothesis is true), thus meeting the z-test conditions. The actual histograms for the paired data in Figure 5.15 do indeed have a shape of normal distribution, but centered around values different from zero.

The results of the test with high statistical significance suggest that for each test the null hypotheses should be rejected in favor of the conclusions that OGR LIC visualization produces lower AAE than LIC, OGR LIC visualization produces lower AAE than OLIC,

visualization	mean AAE (degree)	95% CI
LIC	20.42	[19.28, 21.55]
OLIC	23.80	[23.09, 24.50]
OGR LIC single-frequency	18.18	[17.38, 18.97]
OGR LIC multi-frequency	17.61	[16.83, 18.39]

TABLE 5.1: Interval estimates of the mean AAE for different types of LIC-based visualizations.

and OGR LIC multi-frequency produces lower AAE than single-frequency filtering with p-values practically indistinguishable from zero (z-scores 9.92 38.9, and 18.10).

Finally, we estimate the mean AAE within 95% confidence intervals (CI) based on the 100 samples for each of the four visualizations. The results can be found in the supplementary material and are summarized in Table 5.1.

In summary the evaluation suggests strong evidence supporting our initial assumption that frequency-centric images preserve at least the same amount of the directional information of the original flow with respect to the chosen metric as traditional LIC methods. Importantly, this outcome refers to the verification of correctness of the developed technique rather than implying the usefulness of the resulting visualization for a any particular problem. The latter has to be addressed with a specific user study.

We interpret the achieved results as follows: low spatial-frequency (global) features in LIC images account mostly for noise, while the directional information is captured by high-frequency (localized) components of the visualization. One interesting observation is the observed lower error score of the multi-frequency OGR LICcompared to one-frequency OGR LICvisualization. Clearly, broader spectrum does not automatically yield lower error, since a sum of several frequency-oriented images converges towards plain LIC. A possible explanation for this effect is that refined (single-frequency) visualization can not fully represent the flow detail, while a combination of several (high) frequencies allows for increase in accuracy. Subsequent widening of the spectrum towards the lower end, introduces less useful signal and higher amounts of noise, degrading the overall visualization quality. An optimal range in the spectrum is likely to appear in the high frequency band (naturally bounded by the Nyquist limit). A particular optimal choice of frequencies depending on the flow field could be further investigated in the future work.

5.9 Conclusion

In this chapter we presented OGR LIC—a novel technique to produce frequency-oriented LIC images. It has been demonstrated that by using this technique it is possible to achieve adaptive local spatial frequency control in flow visualization images not relying on noise injection. Unsurprisingly the work has the potential for a number of future improvement directions.

The proposed method features high applicability as a drop-in replacement for classic LIC with the possibility of additional adaptive local frequency control. Closely related to LIC, the presented technique inherits some of its difficulties as well. The particular limitations for several specific applications of the suggested ideas are discussed further.

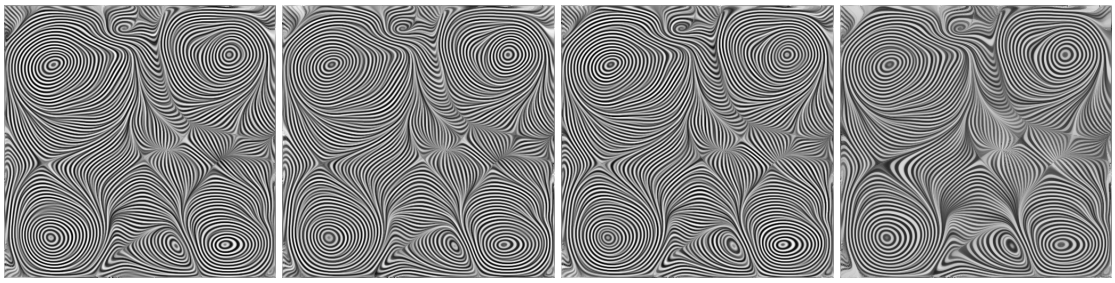

FIGURE 5.16: OGR LIC applied four times to the same vector field but varying the input noise. In the rightmost case, uniform gray with 10 randomly placed white pixels was used as input, and still the method converged.

In particular the technique does not provide an obvious practical solution for rendering the OGR LIC results of volumetric flows. That is, the technique is currently designed to work in the screen-space. Extending it to 3D object space, though possible by itself, requires, to be useful in practice, to additionally solve the problem of efficient rendering of the resulting dense visualization. In Chapter 6 we consider an alternative approach to this problem.

Another feature in common with the traditional LIC is that the presented method is agnostic to the flow magnitude and orientation of the flow. These basic properties can be easily visualized with additional means (e.g., color, glyphs).

The experiments indicate that the suggested method is suitable for generation of continuous visualizations for time dependent data (see the supplementary material) as well, but more research is necessary to validate this property. As mentioned in Section 5.7.3, maintaining the time-coherence between frames requires careful selection of the kernel length in time direction.

As the technical grounds of the method, we have formulated a generalized operator framework that unifies LIC and OGR LIC and allows us to analyze the effect of these methods in the frequency domain. As we show in the next chapter, the a deeper theoretical analysis of the proposed framework is useful for the development of new visualization and flow analysis methods. In particular, the discretization of the generic LIC operator in Equation 5.1 allows to explain and control its effect in terms of matrix and spectral graph theory.

Apart from the presented novel frequency-oriented filtering concept, the presented visualization approach embraces many of the known technical achievements in the DFV: iterated (n-fold) LIC, contrast normalization, and texture stitching (post-LIC multi-frequency texture blending).

The proposed method features the run-time efficiency of LIC, rests on the solid grounds of perceptional studies and the basics of spectral theory, and provides a level of spatial frequency control, to the best of the author's knowledge, unprecedented in texture-based flow visualization.

Chapter 6

3D Vector Fields: A Discrete Probabilistic Framework For Texture-Based Visualization

Traditionally, the various models and methods in texture-based flow visualization are described using a continuous formulation, resting upon the solid grounds of functional analysis. In this chapter, we examine a discrete formulation of common dense flow visualization methods. One of the consequences of such a view is that it allows an expression of Line Integral Convolution and Spot Noise in terms of linear algebra as a matrix-vector multiplication, where the vector represents a noise image and the sparse matrix holds information about the integral curves, the convolution kernel or spot shapes and the interpolation method.

This alternative representation can be interpreted in relation to probability theory, leading to the discovery of a whole new class of visualization models. We propose a novel visualization approach consisting in the computation of spectral embeddings, i.e., characteristic domain maps, defined by particle mixture probabilities. These embeddings are scalar fields that give insight into the mixing processes of flow on different scales. We showcase the utility of the method using different 2D and 3D flows.

6.1 Motivation For Discrete Aproach of DFV

Dense or texture-based flow visualization (DFV) and in particular the Line Integral Convolution method [9, 103] has been proved successful in many scientific and engineering applications. Its wide popularity is the result of such features as suitability for efficient parallel implementation on graphics hardware and the possibility to use adaptive resolution.

Interestingly, despite a vast body of research on the subject [61] and the fact that LIC and dense flow visualization in general is tightly related to the long established branches of mathematics such as numerical methods, to the best of the authors a knowledge there is not yet a consistent theoretical framework that would allow systematic interpretation and exploration of different modifications. For instance, the net effect of the numerous ingredients such as the noise interpolation scheme, the kernel shape and streamline integration sampling methods on the output image cannot always be predicted. Such questions as whether the LIC operator is preserving the average gray value, or whether it is maximum value preserving, cannot be easily answered within the traditional framework.

The wide variety of models and methods in this area are typically formulated within a continuous setting, whereas the gap between the digital nature of the computational world and these continuous models is normally bridged by numerical discretization.

In this chapter, a discrete linear algebra formulation of the common Dense Flow Visualization methods is proposed. It is demonstrated that this formulation is interesting on its own and closer examination can lead to fruitful insights, connections to probability theory and image processing, and new visualization algorithms. In this context the purely discrete algebraic interpretation of the DFV methods such as Line Integral Convolution and Spot Noise can be seen as a step towards a systematic theoretical framework for DFV.

In particular, focusing on LIC, we will take a new look at this commonly used method and reformulate it in terms of conditional expectation computation, using an intuitively arising probability matrix.

This change of paradigm allows for further fruitful exploration. Applying probability modeling to discrete images, we establish a probabilistic relationship between image pixels based on the trajectories of particles seeded in the flow in the cells, corresponding to pixels. Then, we explore the visualization images, constructed as an optimal solution, minimizing expected difference in the color space for cells with similar flow behavior.

The obtained images visualize flow mixture patterns of particles and can be formally described as the eigenvectors of the Laplacian of the particle mixture probability matrix. These eigenvector, widely referred to as spectral embeddings are a powerful tool for the analysis of different types of graphs. In particular, numerous variations of spectral clustering [7], due to its success in computer vision in recent years [66], have gained popularity in image segmentation.

We find the achieved results interesting for applications and encouraging for further research. Briefly, the three major contributions discussed in this chapter are:

- a matrix re-formulation of DFV methods,
- a probabilistic interpretation of LIC,
- a novel discrete probabilistic modeling framework for development and analysis of dense visualization methods.

The remainder of this chapter is structured as follows: In the next sections, we briefly discuss the related work on probabilistic models for flow visualization, flow simplification and segmentation. Then, in Section 6.5 we demonstrate the connection between LIC and linear algebra, deriving the matrix that captures all components of a typical LIC operator and exploring its properties. We show that a similar formulation is valid for the spot noise method as well, shedding some additional light on the nature of these DFV techniques. In Section 6.6, we give a probabilistic interpretation of the previously computed matrix and suggest a novel discrete probabilistic model for flow representation, with applications to dense flow visualization. In Section 6.7, we discuss the technical details of the method. In Section 6.8, we highlight some interesting properties of the resulting visualizations and make a quantitative evaluation of their important features, comparing them to LIC. Finally, the visualization results are demonstrated along with the discussion of possible future work directions.

6.2 Probabilistic Models For Flow Visualization

The dense flow visualization paradigm, starting with the introduction of spot noise by van Wijk [131] and Line Integral Convolution by Cabral and Leedom [9], has undergone significant development over almost 25 years. The state-of-the-art report by Laramee et al. [61] enumerates a large set of methods derived from these two approaches. Mainly, the efforts have been focused on extension to further dimensions, such as 3D LIC [91], LIC on surfaces [28], and efficient implementation [105, 127].

Although new competitive methods have appeared, LIC in its various modifications has remained a workhorse of DFV. Several works concerning its theoretical grounds and improvement have been published. A thorough analysis of the influence of LIC parameters from the signal processing standpoint was done by Stalling [104]. The concept of two-fold LIC was introduced for image enhancement by Okada and Lane [78]. Its value and benefits for computation acceleration were later recognized by Weiskopf [128] and Hlawatsch et al. [38].

Based on the generalization of existing LIC techniques, in this chapter a discrete probabilistic model of flow mixture in the domain is formulated. Notably, the efforts to employ probabilistic methods for analysis of streamline separation were successful previously in the work of Reich et al. [89]. They visualize a measure of convergence and divergence between particles seeded in the neighboring cells (pixels), after a number of iterations of a Markov chain over each particle's initial probability distribution. The matrix model presented here differs from this setting in the following two main aspects: 1) it embraces the information about the whole integral curve, instead of a particle movement in a single time step; 2) the time-consuming iterative eigenvector computation process is required only once, but not for every domain cell. As a result of much lower computational complexity, the presented method is applicable in 3D. From the visualization perspective, we propose a global map of the domain with a progressive level of detail characterizing the mixture relationship between the cells in the domain.

6.3 Flow Simplification

The technique presented in this chapter aims to highlight structures in the underlying data and to provide their visual representation at different scales. Within existing classification, the presented visualization approach can be described as partition-based. The state-of-the-art report on this topic [96] names two main subclasses in this area: based on vector value clustering and relying on integral line analysis. The method combines the features of both approaches: distinguishing the regions of the flow domain and using the information about particle trajectories instead of the vector value.

The principal idea of flow simplification is extensively exploited from a different perspective in topological methods. An outlook on this research is given by Salzbrunn et al. [95] and we name only a few notable results. Helman and Hesselink [37] extracted the critical points and separatrices of 2D vector fields, which provided a segmentation into sectors of different flow behavior. Topological simplification [118, 126] is a way to identify the more salient topological features in a flow. Recent developments include combinatorial vector field topology [90] and streamline clustering using Morse Connection Graphs [108] or streamline predicates [94].

Other approaches simplify or segment the flow without referring to topology. For example, Rössl et al. [93] group streamlines using their projection into Euclidean space with a Hausdorff distance. The methods of Garcke et al. [30], Heckel et al. [36], and Telea and van Wijk [113] cluster cells with similar flow vectors. A DFV approach to accentuating flow structures was suggested by Park et al. [80].

6.4 Spectral Image Segmentation

The technical side of the approach discussed here is inspired by the quadratic form minimization algorithm discussed in Chapter 4, relying on the similar mathematical apparatus of sparse matrix computations. The eigenvector computation, resulting from the analysis of the discussed probabilistic model, has a direct correspondence to the spectral embeddings technique widely used in the image processing domain, where it consists in the representation of the image segmentation as a graph cut problem with its consequent relaxation using spectral graph theory. For instance, the normalized cut method in image segmentation gained wide popularity after the presentation of the approach by Shi and Malik [101]. For introductory reading on the subject, we suggest the tutorial by Luxburg [66]. The original graph formulation was followed by the random walk interpretation of Meila and Shi [71], which assigns intuitive meaning to the spectral embeddings, relating them to the concept of low conductivity sets. Finally, the theoretical aspects of the algorithm were carefully treated by Brand and Huang [7].

6.5 Matrix Representation of DFV Methods

The flow visualization technique suggested here is probably best explained in relation to LIC.

Throughout the literature this method is defined in a continuous setting, so before proceeding with the discrete formulation, please refer to the more familiar traditional formulation, discussed in Chapter**??** Equation 1.7 Although in this section we restrict ourselves for simplicity to the stationary 2D flow case, it is demonstrated later that it is not a restriction for the method, as it is not for LIC. In the subsequent analysis, we explore the discretization of the resulting integral on the image grid.

6.5.1 Matrix Representation of LIC

Let us discuss a formulation of LIC and spot noise techniques as a discrete linear operator, which although quite straightforward, has far-reaching implications on the one hand and on the other hand to the best of author's knowledge has not been discussed in the visualization literature.

The idea behind the discrete LIC formulation is to capture all the information about the transformation of the image by the flow field in one matrix, while the input noise and output image are represented as vectors in $\mathbb{R}^n$. One of the advantages of such representation is that it simplifies the analysis of the net effect of various choices involved in LIC implementation on the input. The components that constitute the LIC matrix are:

- integral curves sampling,
- input noise interpolation weights $w(\mathbf{x})$,
- convolution kernel $k(t)$.

As demonstrated below, this matrix arises naturally in a practical implementation as a result of finite image resolutions.

A discrete version of the LIC Equation 1.7 has already been studied in the literature [128] in the context of algorithm performance optimization, but only the discrete computation of the line integral itself was taken into account. We take the discretization a step further, additionally explicitly specifying discrete input and output images.

First, as a result of integral discretization, we consider a finite number of particle positions, sampled at different distances along the same streamline. Formally, a particle p with initial position $\mathbf{x}_p$ moving along a streamline $\sigma_p(s)$ is sampled at the distances s_i, providing a set of positions $\mathbf{x}_{pi} = \sigma_p(s_i)$.

Next, we assume that the input image is sampled from an in-memory texture, with some common interpolation method (linear, bilinear, spline), as opposed to arbitrary procedural input texture generation (e.g., a non-smooth analytically defined function). In other words, any input function that can be sampled on a grid and then reconstructed with interpolation satisfies this constraint.

Suppose that the input image is given on a grid $\mathbf{h} = \{\mathbf{h}_k\}$ with all the grid nodes enumerated in some sequential order with one-dimensional index k. An input value $n(\mathbf{x}_{pi})$ is then interpolated as in Equation 6.1:

$$n(\mathbf{x}_{pi}) = \sum_k n_k w_k(\mathbf{x}_{pi}) \tag{6.1}$$

We discuss when this is representation is possible in the next passage. Now, using the interpolation formula in Equation 6.1, the discrete version of the integral in Equation 1.7 can be written as in Equation 6.2:

$$u_p = \sum_{i=-L}^{L} k(t_i) \sum_k n_k w_k(\mathbf{x}_{pi}) \tag{6.2}$$

Changing of the order of sums and setting $A_{pk} = \sum_{i=-L}^{L} k(t_i) w_k(\mathbf{x}_{pi})$ results in a basic matrix-vector product representation of LIC in Equation 6.3.

$$u_p = \sum_k A_{pk} n_k \tag{6.3}$$

Further we call the matrix A a **LIC matrix**. Each row of this matrix can be represented as an image of one or more flow integral line segments as in Figure 6.1.

Finally, let us discuss the conditions when the linear input representation of the noise in Equation 6.1 is applicable. In fact for the discrete noise input and discrete interpolation output the resulting transformation is linear even if the interpolation scheme is not linear: given a set of interpolation functions $b_j(\mathbf{x})$, the resulting signal is obtained

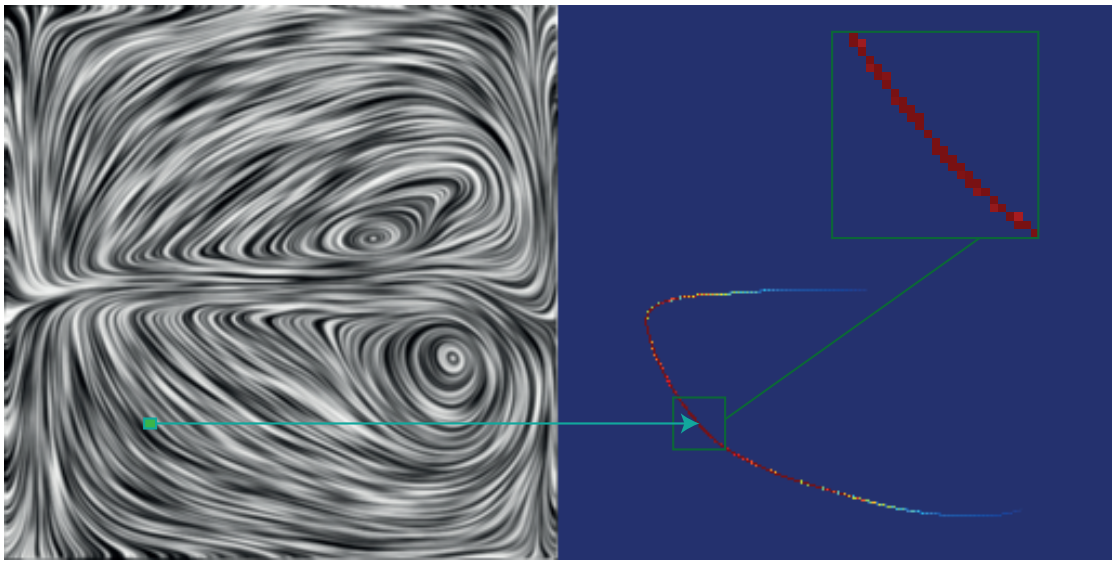

FIGURE 6.1: The left image shows a 2D vector field visualized using LIC, with one pixel marked with a green rectangle. The right image represents one row of the sparse LIC matrix represented as a two-dimensional array with a rainbow color map applied to it. The matrix is computed using nearest neighbor interpolation and Gaussian kernel.

as $n(\mathbf{x}) = \sum_j c_j b_j(\mathbf{x})$ and the vector of interpolation coefficients $\mathbf{c}$ can be computed from Equation 6.4:

$$\begin{aligned} U(\mathbf{h})\mathbf{c} &= \mathbf{n} \\ \mathbf{c} &= U(\mathbf{h})^{-1}\mathbf{n} \end{aligned} \tag{6.4}$$

where $U = U(\mathbf{h})$ is the matrix of interpolation functions evaluated at the data points $\mathbf{h}_k$, such that $U_{kj} = b_j(\mathbf{h}_k)$ and $\mathbf{n} = \mathbf{n}(\mathbf{h})$ is the vector of sampled input signal values at the nodes $\mathbf{h}_k$. The reconstructed signal at point $\mathbf{x}$ is then given by Equation 6.5:

$$\mathbf{n}(\mathbf{x}) = \langle \mathbf{b}(\mathbf{x}), \mathbf{c} \rangle = \mathbf{b}(\mathbf{x})^T U(\mathbf{h})^{-1} \mathbf{n}(\mathbf{h}) = \sum_k n_k w_k(\mathbf{x}_{pi}) \tag{6.5}$$

where vector $\mathbf{b}(\mathbf{x})$ has coordinates $b_j(\mathbf{x})$. That is, in typical implementation scenarios (operating on digital images, the representation in Equation 6.1 is valid.

In summary, we have shown that the LIC operator acting on a discrete texture can be represented by a single linear transformation of the input, captured by matrix A even if a non-linear interpolation is employed for the reconstruction of input and without any assumptions about the integral curves computation and sampling method. In the next sections we will further study and extend this transformation.

6.5.2 Matrix Representation of Spot Noise

This line of thinking can further be extended to other DFV techniques. As a brief note, we provide an example of another canonical DFV method, spot noise, which also fits into this framework. Sampling the spot function h on the grid $\mathbf{x}_j$, one can immediately

update the core spot noise computation formula [61] and obtain Equation 6.6

$$f_j = \sum_i a_i h(\mathbf{x}_i - \mathbf{x}_j, \mathbf{v}(\mathbf{x}_i)). \tag{6.6}$$

It is worth noting that the spot matrix $h_{ij} = h(\mathbf{x}_i - \mathbf{x}_j, v(\mathbf{x}_i))$ appearing in place of the spot function h differs significantly in structure from the matrix arising in LIC. Likewise, the vector space of random scaling factors a_i is not the image space to which f_j belongs and might even have a different dimensionality. In order to keep the focus on the practical application of the approach, we further explore and develop the matrix representation obtained from the LIC formulation and do not investigate other techniques in depth in this chapter.

6.5.3 Some Basic Features of the LIC Matrix Representation

In order to demonstrate the utility of the discrete matrix formulation, we briefly highlight some of its straightforward implications. Some of the properties of the LIC operator can be immediately identified and understood in this form, but are less obvious in the continuous setting. In particular, interpreting the LIC operator as an image filter allows us to apply basic results from image processing to study its effect in different settings.

Filter Sequences and Iteration. One of the techniques for the enhancement of LIC output is the iteration of the LIC kernel, combined with a high-pass filter suggested by Okada and Lane [78].

In the matrix framework, the cumulative effect of a consecutive application of the LIC operator A and another filter B can be represented by the multiplication of the input by one matrix BA. This view allows a transparent combination of LIC with basic image filters representable in the matrix form (e.g., Box, Gaussian, Laplacian).

In particular, the sequence of filters suggested by Okada and Lane can be represented by matrix A^2H where H performs a convolution with some high-pass kernel. Here we do not take into account their final non-linear contrast-enhancement step (histogram equalization), which can be seen as post-processing.

Moreover, n multiplications by the LIC matrix A are clearly equivalent to the multiplication by A^n and it is well known from algebra that $A^n\mathbf{x}$ under mild assumptions on $\mathbf{x}$ converges to the dominant eigenvector $\mathbf{v}$ of A for $n \to \infty$, with $A\mathbf{v} = \lambda\mathbf{v}$ by definition. In other words, the eigenvectors of the image filter matrix can be interpreted as stationary points of its powers iteration.

Maximum and Average Value Preservation. Additionally, two basic conclusions can be made, relating the properties of the input and output images. It is known that the matrix multiplication preserves the maximum norm (as an upper bound) if the sum of the row elements is equal to one, i.e., $\sum_q A_{pq} = 1$. Indeed, consider Equation 6.7:

$$u_p = \sum_k A_{pk} n_k \leq \sum_k A_{pk} \max_k n_k = \max_k n_k \tag{6.7}$$

Equation 6.7 holds when k and w are normalized such that

$$\sum_{i=-L}^{L} k(t_i) = 1 \tag{6.8}$$

and $\sum_k w_k(\mathbf{x}_{pi}) = 1$.

It is also easy to check that the average value of the input $\mathbf{n}$ is preserved under the matrix multiplication if the sum of the matrix column elements is equal to one. That is, given $\sum_p A_{pk} = 1$ the Equation 6.9 holds,

$$\sum_p u_p = \sum_p \sum_k A_{pk} n_k = \sum_k n_k \tag{6.9}$$

This property is not guaranteed by the standard LIC techniques but can be enforced by column normalization. Intuitively, it means that all pixels possess the same amount of gray value they can redistribute among their neighbors. In particular, the value of any pixel in the vicinity of a critical point, hit by multiple streamlines, makes only a very small contribution to the value of other pixels on each of these streamlines. The lack of average value preservation can cause a visual effect of gray value smearing around critical points.

In Section 6.6.2, we will discuss the importance of the column-wise versus row-wise normalization, in the context of probabilistic interpretation of the matrices.

6.6 Probabilistic Approach to Flow Visualization

This section represents a contribution to the toolbox of methods for dense flow visualization. We use the discrete matrix representation derived in the previous section to represent the flow domain and the particle trajectories. Furthermore, we combine this representation with probabilistic modeling to analyze the discrete flow domain in terms of particle transport (mixture probability).

First, we briefly look at the LIC matrix in terms of probability theory and then develop a novel model based on this view.

6.6.1 Probabilistic Interpretation of the LIC Matrix

Within the apparatus of probability theory, the LIC operator can be described as a computation of expected value for each pixel over a set of neighbor pixel values.

Considering a two-dimensional stationary flow domain for simplicity, we introduce a rectangular grid over the domain, consisting of cells corresponding to image pixels. We associate a particle s_i with each cell c_i and observe it for a certain time t before and after it is registered in some chosen position within the cell.

We introduce two random variables: S, taking the values on the set of observed particles and C, taking the value on the set of cells. The particle can visit a number of other cells while moving along the streamline, as illustrated by Figure 6.2a.

For each cell on the particle trajectory, a conditional probability distribution $P(C = c_j | S = s_i)$ is assumed, denoting the probability of a particular cell c_j being visited, given

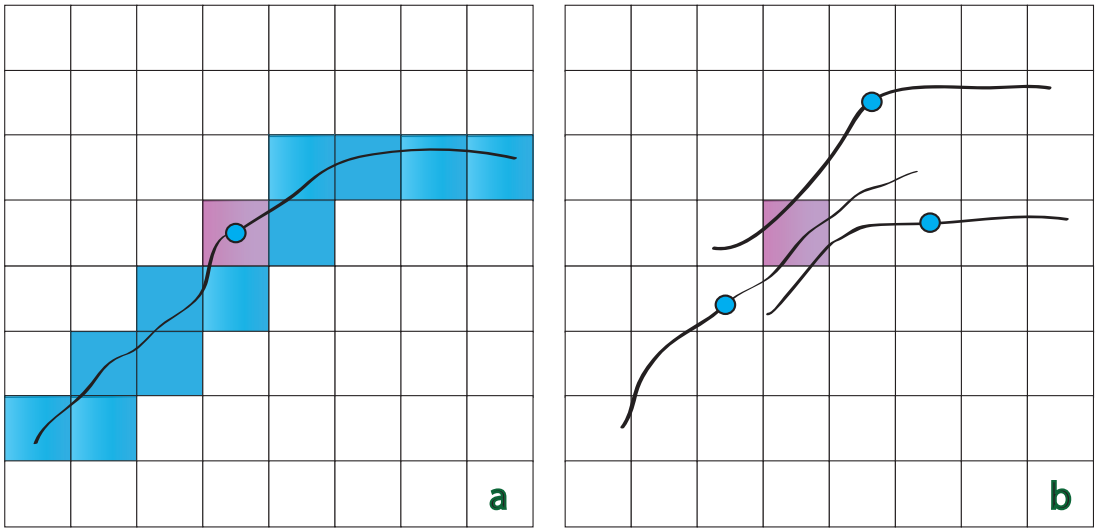

FIGURE 6.2: a) The blue cells on the trace of a particle, contributing to the computation of the image value in the red cell, where the particle is observed. b) Several blue particles, each seeded in its own cell, contribute to the value of one red cell.

we observe a particle s_i. This distribution is expressed by the shape of the LIC kernel and represents the measure of relevance of c_j to the trace of s_i or the uncertainty of the particle position within a time window.

Two common examples are the box kernel and the Gaussian. The box kernel, corresponding to the uniform distribution, suggests that each cell on the particle trajectory equally likely captures the position of the particle. The Gaussian shape expresses an increasing uncertainty about the particle position with increasing distance from the initial point. The described probabilities directly correspond to the entries of the LIC matrix, if the rows are normalized such that : $\sum_j P(C = c_j|S = s_i) = 1$.

From this point of view, given a 2D signal U, the LIC operator computes for each of the cells c_i and the particles s_i associated with them, the conditional expectation in Equation 6.10:

$$E_{C|S}[U] = \sum_j P(C = c_j|S = s_i)U(c_j) \tag{6.10}$$

of the input distribution over a set of cells visited by the particle s_i. As a result, the output values are correlated for two particles, if they produce overlapping traces, that is traces sharing common cells. In particular, the input signal U can represent the initial position probability distribution of a single particle s over all domain cells, as, for instance, in the model suggested for streamline separation by Reich [89]. In this case, the expected value in the cell c_i corresponds to the probability of particle s arriving in cell c_i. This case is intuitively illustrated with Oriented LIC [125], considered within the probabilistic framework. If the sparse input texture represents the distribution of possible initial positions of a single particle and the LIC kernel is asymmetric (backward flow direction only), then for each output image cell one possible trajectory history is sampled and the resulting image represents the probability that the initial particle arrives to this cell.

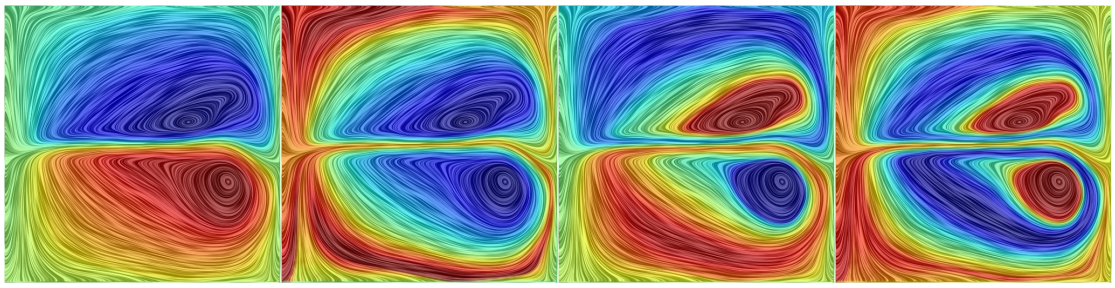

FIGURE 6.3: First four eigenvectors of the Laplacian of the mixture probability matrix H represented as two-dimensional arrays and with rainbow color map applied. A LIC texture of the flow is shown in the background. The eigenvectors are ordered from left to right by the corresponding eigenvalue.

In the next section we demonstrate how, extending the probabilistic interpretation, a novel flow visualization approach can be formulated.

6.6.2 Idea of Particle Mixing Probability

In the above formulation, we interpret the widely used LIC method, as seeding a particle within each image cell and computing the intensity value for this cell based on the trajectory of this particle. It is then natural to switch the focus from particles directly to cells. Such a shift would correspond to the transition from Lagrangian to Eulerian approach, common in the study of fluid dynamics. This viewpoint change allows us to formulate the requirements for the resulting visualization image explicitly, since the cells are directly linked to image pixels.

So, we are interested in the probabilities $P(S = s_i|C = c_j)$. That is, given the cell c_j is observed, we compute the conditional probability of each of the particles s_i (each originating from its own cell c_i) arriving at this cell. This cell-centric view is illustrated by Figure 6.2 b. Applying Bayes' theorem, one gets Equation 6.11

$$P(S = s_i|C = c_j) = \frac{P(C = c_j|S = s_i)P(S = s_i)}{P(C = c_j)} \tag{6.11}$$

where $P(S = s_i)$ is the marginal probability of a particular particle. We assume here that for any s_i $P(S = s_i) = \frac{1}{n}$.

The marginal probability $P(C = c_j)$ that a particular cell is visited by any of the particles being observed is $P(C = c_j) = \frac{1}{n}\sum_i P(C = c_j|S = s_i)$. Note that technically the transition from $P(C = c_j|S = s_i)$ to $P(S = s_i|C = c_j)$ is implemented by the normalization of columns of the original matrix.

The computed conditional probability allows us to answer the following question: **what is the probability that two particles s_i and s_j emitted from cells c_i and c_j will meet in some cell?** By "meeting" here we refer to visiting the same cell, not necessarily at the same moment but within a certain time interval (defined by the kernel length). We use δ_{ij} to denote this probability. It is computed by summing up the probability of these

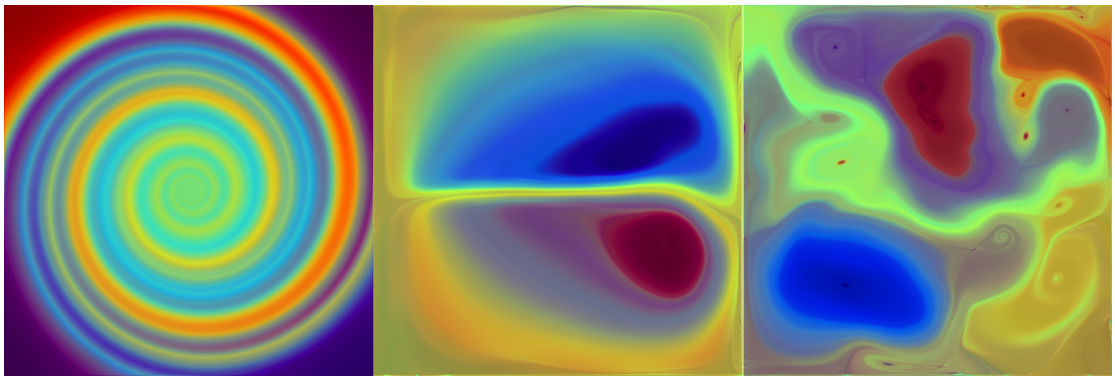

FIGURE 6.4: A visualization of several embeddings for each flow combined with one transfer function. The transfer function effectively blends rainbow-colored embedding images.

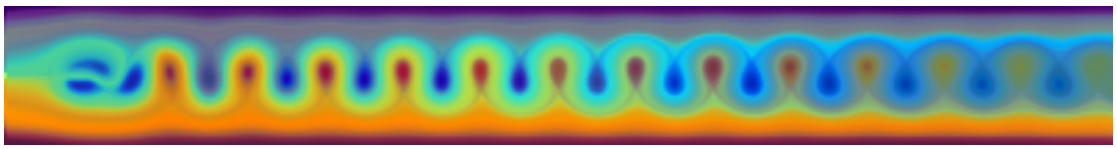

FIGURE 6.5: A visualization of several embeddings for a von Kármán vortex street in the flow behind a cylinder. Embeddings are combined with a transfer function (average of rainbow-colored embedding images).

particles visiting the same cell c_k over all cells.

$$\delta_{ij} = \sum_k P(S = s_i | C = c_k) P(S = s_j | C = c_k) \tag{6.12}$$

Formally, this operation can be described as the computation of a matrix $H = PP^T$, $h_{ij} = \delta_{ij}$ of the conditional probability matrix P such that $P_{ij} = P(S = s_i | C = c_j)$. The main diagonal of H, corresponding to the probability that two particles starting at the same cell meet, is set to 1.

This probability of two particles visiting the same cell provides important information about the flow domain connectivity. It is also important to remember that the computed probabilities are restricted to the particle movement for a certain predefined time period. Further, we refer to this probability as to short-term mixture probability. Clearly, for trajectories that are nowhere closer than one cell size apart, this probability is zero and it is higher for largely overlapping trajectories.

6.6.3 Flow Visualization Using Mixture Probability

The short-term mixture probability introduced above is a relation defined for every pair of cells, that is for n^2 values with n being the number of cells. The direct visualization of this additional amount of data by itself clearly presents a significant challenge. However, one particular advantage of this representation is that it allows us to compute a sequence of uncorrelated domain feature maps that reveal the domain connectivity on different scales.

We define a vector $\mathbf{f} \in R^n$ or a feature map f_i for each cell c_i, as a minimizer of an expected quadratic error function $e_{ij} = (f_i - f_j)^2$, with constant non-zero total energy.

$$\min_{\mathbf{f}} E_f = \sum_{ij} \delta_{ij}(f_i - f_j)^2 \qquad (6.13)$$
$$\sum_i f_i^2 = 1$$

The first, smoothness condition, above ensures that the difference between feature map values for any two cells is penalized in direct proportion to their short-term mixture probability whereas the second, energy condition, restricts the problem to non-trivial solutions.

One optimal value is achieved with a constant feature map $f = v^0$, such that $v_i^0 = \frac{1}{\sqrt{n}}$, which is not useful for the purposes of the flow behavior analysis. Further we show how the complexity of the solution can be increased gradually. The approach taken is well studied in machine learning and is known as the spectral embedding of a graph (induced by the particle mixture relationship between cells).

Equation 6.13 can be rewritten using matrix notation as a minimization of quadratic form $E_f = \mathbf{f}^T L\mathbf{f}$ for $||\mathbf{f}|| = 1$ where L is a Laplacian matrix $L = D - H$ and D is a degree matrix, holding the row sums of H on the main diagonal $D_{ii} = \sum_j h_{ij}$. Two important properties of the matrix L are that it is symmetric and positive semi-definite (the second can be checked as $\mathbf{x}^T L\mathbf{x} \geq 0$ for any $\mathbf{x}$). This observation implies that all of its eigenvectors are orthogonal and eigenvalues are non-negative. Since eigenvectors $\mathbf{v}_i$ of L form a orthogonal basis in R^n, every solution of Equation 6.13 can be represented as their combination in a unique way

$$\mathbf{f} = \sum_{j=0}^{n-1} \mathbf{v}_j \left\langle \mathbf{v}_j, \mathbf{f} \right\rangle \qquad (6.14)$$

The objective E_f can be rewritten as :

$$E[\mathbf{f}] = \sum_{j=0}^{n-1} \left\langle \mathbf{v}_j, \mathbf{f} \right\rangle^2 \lambda_j \qquad (6.15)$$

since $L\mathbf{v}_i = \lambda_i \mathbf{v}_i$ and $||\mathbf{v}_i|| = 1$ by the definition of eigenvectors and $< \mathbf{v}_i, \mathbf{v}_j >= 0$ for $i \neq j$.

The eigenvalues λ_i are non-negative due to the Laplacian positive semi-definiteness, so it is convenient to enumerate them in the ascending order: $\lambda_i \leq \lambda_{i+1}$ Then, the minimum of the $E[f]$ is attained on the first eigenvector v_0, as follows from Equation 6.15:

$$E[\mathbf{f}] \geq \lambda_0 \sum_{j=0}^{n-1} \left\langle \mathbf{v}_j, f \right\rangle^2 = \lambda_0 ||V^T \mathbf{f}||^2 = \lambda_0 = E[\mathbf{v}_0] \qquad (6.16)$$

where V is an orthogonal matrix that contains vectors $\mathbf{v}$. The multiplication by this matrix preserves the unit norm of $\mathbf{f}$ since $\mathbf{x}^T VV^T \mathbf{x} = \mathbf{x}^T \mathbf{x}$.

As mentioned above, the first eigenvector is of little interest, but we can restrict the space of solutions to non-constant vectors, introducing an additional constraint: the new

solution is orthogonal to the constant one. Analogously to the estimate in Equation 6.16, setting $\langle \mathbf{f}, \mathbf{v}_0 \rangle = 0$ in Equation 6.15 results in the optimal value being λ_1 achieved at $\mathbf{v}_1$.

Repeating this step, it is possible to progressively refine the vector space of admitted solutions, constructing increasingly more detailed feature maps, which we call flow spectral embeddings by analogy with graph theory. Requiring the new solution to be orthogonal to the previously computed solution, we ensure that there is no correlation between the new image and any linear combination of previously computed images.

At the same time the smoothness objective controls the discrepancy in the solution cell values according to the cell mixture probability so that cells with a higher probability of mixing maintain a small difference in value for a larger number of steps.

The described algorithm computes a sequence of feature maps, visualizing the short-term cell mixture probability in the domain, increasingly adding small features. We can summarize the steps of the algorithm as follows:

1. choose the integration distance d and the trajectory certainty distribution,
2. trace one or more particles s_i for each cell c_j,
3. sample the particle positions and store the probability $P(c_j|s_i)$ of visiting cell c_j, given the particle s_i is observed,
4. store the $P(c_j|s_i)$ in a matrix P,
5. compute the probability $P(s_i|c_j)$ of particle s_i visiting the cell, given that cell c_j is observed, normalizing the matrix P by rows,
6. compute the short-term mixing probability matrix $H = PP^T$, i.e., the probability that two particles, originating from cells c_i and c_j, meet at some cell,
7. compute the Laplacian matrix $L = D - H$,
8. compute first several eigenvectors of the Laplacian (corresponding to smallest eigenvalues).

For visualization purposes, we explore a few of the first such maps (embeddings), that is, eigenvectors corresponding to the first several smallest eigenvalues of the Laplacian matrix. The obtained sequence of images exhibits multi-scale structure, with the spatial frequency increasing with the number of eigenvectors computed as demonstrated by Figure 6.3. It is important to note that the maps are ordered by the expected error and are orthogonal to each other. In other words, each subsequent map constains a smaller amount of detail compared to the previously constructed set. These detail is not present in the previous maps and has higher spatial frequency (lower smoothness). Consequently, the eigenvectors with large indexes (specific number depending on the flow) look as uniform noise. This is a result of the spatial frequency reaching the Nyquist limit [33] of the chosen image resolution and of the increasing numerical imprecision associated with eigenvalue computation.

The resulting images visualize the connectivity of the domain based on flow transport, since the mixture probability for any two small domain regions, which is defined as the probability of the meeting of particles seeded in these regions, corresponds to the

difference in colors assigned to these regions in the visualization. Such a presentation simplifies the interpretation of the flow mixture patterns even for a non-expert user.

Although the spectral embeddings by themselves already capture the flow structure and to some extent visualize the flow transport, they can be combined into a single image using an n-dimensional transfer function for enhancement of detail on different scales. The examples of such visualization for the 2D case are shown in Figure 6.4 and Figure 6.5. In this case the transfer function is constructed as $TF = \sum_k 2^{-k} r(v_k)$ where r is a rainbow color transfer function and v_k are the embeddings. The practical computation of the embeddings is discussed in Section 6.7. In summary, the numerical scheme requires only a series of sparse matrix-vector multiplications, which are provided by many general-purpose sparse linear algebra software packages, including those that are GPU-accelerated.

6.6.4 Method Extensions

The probabilistic model we described consists of several pluggable components, which can be adjusted to the requirements for a particular visualization task at hand. These settings are not all specific to the presented approach but are largely shared with the Line Integral Convolution method. We interpret the effect of possible modifications in the new context, not aiming to demonstrate all their combinations. Staying within the bounds of expected error minimization framework, we briefly discuss different approaches for the definition of the input probabilities and different domains types.

The trajectory certainty distribution corresponds to the LIC kernel with the restriction that it represents the conditional probability and hence is positive and normalized. There are no assumptions about the shape of the kernel, in particular it does not have to be symmetric. For example, a one-sided kernel (defined on a positive or negative half of the real line) would correspond to either injecting particles in the cell and following their position or registering the arriving particles trajectories in the cell. The mixing probability then can be interpreted as the probability of mixing in the future (reaching a common sink) or in the past (originating from a common source). Since the short-time mixing probability matrix properties are independent of the kernel shape, the other computational steps remain unchanged.

The concept of particle trajectories is generic and is not limited to the physical particle movement. In the case of unsteady flow, the probability of virtual particles mixing in the domain cells can be computed for streamlines as well as streak lines or path lines.

Different objectives can be achieved with two alternative types of trajectory parameterization: by time or by distance traveled by a particle. In the first case the mixing probability corresponds to mixing in a particular time period. In the second case, a mixing in a neighborhood of a particular size in space is considered, effectively ignoring the particle velocity magnitude.

Another option worth noting is that, empirically, tracing multiple particles per cell improves the stability of the solution, because more possible trajectories are considered. This improvement, however, comes at the cost of elevated requirements to the computational power and a potential decrease of the sparsity of the matrix in the regions

with extremely chaotic flow behavior. For the discussion of the influence of the sparsity property on the practical feasibility of the problem, please refer to Section 6.7.

Finally, the presented method does not make any assumptions about the dimensionality of the flow domain and hence is applicable in 3D without modification. The straightforward extension of the method to 3D makes it possible to use common volume rendering utilities to visualize the flow embeddings. We demonstrate several 3D examples in Section 6.8.3.

6.7 Implementation Details

In this section, some technical aspects of the current implementation are highlighted and possible performance issues mentioned. The core of the method is the computation of eigenvectors of the Laplacian matrix L. The computation of the conditional probability matrix P entries is not essentially different from the standard LIC algorithm and the search for eigenvectors relies on well-established algebraic routines, allowing straightforward implementation using existing software packages for scientific computation.

The subsequent steps are simplified by the explicit storage of uncertainties $P(c_j|s_i)$ in a matrix, but it is not necessary to keep this matrix in memory, since the probability $P(s_i|c_j)$ can be computed directly by summing up the contribution of all incoming particles for each cell.

We compute the integral line segments using a basic Euler integration scheme, implemented using NVidia CUDA platform, and store them in memory. Usage of higher order integration schemes is of course possible, but since the input for the model are particle trajectories, rather than the vector field, we leave the discussion of the accuracy of the computed lines out of the scope. The remaining computations are implemented with the means of the SciPy stack [47]. For the sake of reproducibility, the complete implementation of the technique and examples is provided for public access on the web[1].

6.7.1 Sparse Matrix Computations

An important property of the matrix P that makes the eigenvector computations feasible in practice is its sparsity. Let us assume the average number of pixels D on each particle trace is much smaller than the image domain size N (the total number of pixels/voxels), and the interpolation requires K input pixels to compute one output pixel. Under these typical conditions, each LIC matrix row contains $O(KD)$ non-zero values, and the total number of non-zero matrix entries is $O(NKD)$.

Since the matrix size grows as $O(N^2)$ with the size of image N, it quickly becomes challenging to handle on a computer for large image domains in a naive way. Fortunately, several numerical methods are available that exploit the sparsity structure, featuring memory requirements and run times that are linear in the number of non-zero entries.

Indeed, it is not necessary to store the matrices H or PP^T for the purpose of Laplacian eigenvector computation. As a numerical scheme for eigenvector computation, we employ the LOBPCG method [53], which is implemented in terms of matrix-vector multiplications or linear operator application to a vector. Assignment of all ones to the

[1] `http://matvict.github.io/flow_embedding/`

flow	resolution	kernel	time (min)	% non-zero
borromean	128×128×128	80	149	0.019 %
abc	128×128×128	80	118	0.014 %
benard	128×128×128	80	198	0.020 %
spherical drop	128×128×128	80	106	0.013 %
stuart vortex	128×128×128	80	113	0.013 %

TABLE 6.1: Run-time measurements for computation of the embeddings for some of the presented 3D flows.

diagonal of matrix H and the subsequent multiplication $H\mathbf{v}$ can be expressed using P as $H\mathbf{v} = P(P^T\mathbf{v}) - \Lambda\mathbf{v} + \mathbf{v}$, where the subtracted diagonal matrix $\Lambda_{ii} = \sum_j P_{ij}^2$ can be computed from P with $O(N)$ additional memory. The degree matrix D can be computed using a all-ones vector $\phi : \phi_i = 1$ $D = H\phi = (PP^T - \Lambda + I)\phi = (P - \Lambda + I)\phi$. Here we use the fact that due to normalization, $P^T\phi = \phi$. Such explicit degree matrix computation allows us also to use other types of Laplacian for the formulation of expected error functional, for example normalized Laplacian $L = I - D^{-1/2}HD^{-1/2}$.

6.7.2 Performance

The current implementation is prototypical and requires significant preprocessing times (up to 20 minutes per volume) for later interactive flow visualization. The run time of the method for 2D images is dominated by the traditional LIC matrix computation, whereas the visualization image itself is formed on an order of seconds. The eigenvector computation, however, is the current bottleneck in 3D, and can last from several minutes to several hours. It is important to notice that this step involves only sparse matrix-vector multiplications and can be parallelized effectively on multi/many-core hardware using, for example, the several available libraries for sparse matrix computations on GPU, such as cuSPARSE. From the theoretical perspective, the memory requirements of the technique and asymptotic complexity of the involved algorithms are both linear in the number of non-zero entries of the LIC matrix.

Actual performance measurements for the computation of 4 first Laplacian eigenvectors for some of the presented 3D flows are provided in Table 6.1. Generally, the computation time can be affected by many factors, including

- output image/volume resolution,
- kernel length,
- number of computed eigenvectors,
- interpolation scheme,
- sparse matrix storage scheme.

6.8 Discussion of Results

6.8.1 Evaluation of Flow Information in the Embeddings

In this section, we investigate how much information about the original flow is captured by the computed embeddings. The purpose of the evaluation is to provide a quantitative comparison of the amount of information about the original flow compared to traditional visualization methods. In other words, we try to verify how much useful data is lost (in comparison to LIC, or streamlines). Our basic assumption here is: **a representation A preserving more information about the original flow than representation B allows to perform at least the same flow analysis as B.**

We approach this problem by conducting statistical tests, largely relying on the framework for information-theoretic analysis of visualization suggested by Chen and Jänicke [12]. For several random flow fields, we compute normalized mutual information between flow features and visualization features.

First, we estimate the uncertainty in the flow direction contained within the embedding visualization gradient, comparing it to the corresponding LIC image. Then, we demonstrate that on average embedding value contains a significant amount of information about flow streamline segments.

We use the notion of mutual information as a quantitative measure, which is defined for two random variables X and Y as in Equation 6.17.

$$I(X;Y) = \int_X \int_Y p(x,y) \log(\frac{p(x,y)}{p(x)p(y)}) dxdy \tag{6.17}$$

where $p(x)$ and $p(y)$ stand for the marginal probability density functions of X and Y and $p(x,y)$ stands for their joint probability density. Numerically these probabilities are estimated with histograms.

Gradient Information. In this experiment, we compare the flow direction with the direction of the gradient of the visualization. As suggested by the previous work on flow evaluation [69], the direction, orthogonal to the image gradient in the dense flow visualization, is important for the characterization of the original flow.

We proceed as follows: randomly generate 100 2D vector flows and quantize the flow orientation $\alpha(x)$ into 128 bins, such that vectors with the same direction but opposite orientation belong to the same bin. Then, we apply this operation to the corresponding visualizations gradients $\gamma(x)$ (LIC) and $\beta(x)$ (embedding, corresponding to the first eigenvector). In the next step, we compute the mutual information between the original flow and the embedding image $I(\alpha,\beta)$ and between the original flow and the LIC image $I(\alpha,\gamma)$.

Finally, we compute the uncertainty coefficients [87] for both visualizations, i.e., the mutual information normalized by the entropy of the source data: $C_{emb} = \frac{I(\alpha,\beta)}{H(\alpha)}$ $C_{lic} = \frac{I(\alpha,\gamma)}{H(\alpha)}$. The uncertainty coefficients represent the fraction of the original direction information present in the visualization gradient.

We conducted the experiments for different streamline integration lengths, each time keeping this parameter the same for LIC and for our method. Based on the experiments, it can be stated that the 99% confidence intervals for the mean value of

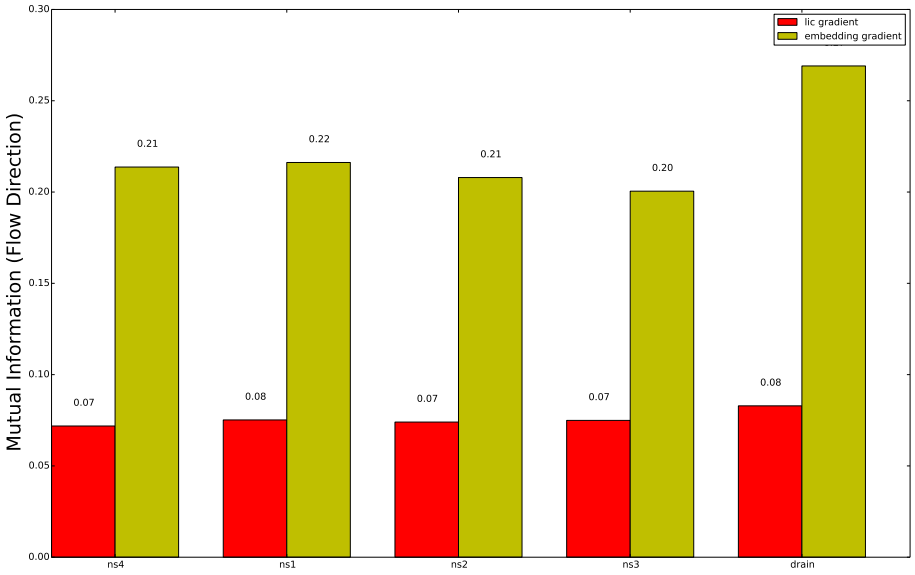

FIGURE 6.6: The normalized mutual information (uncertainty coefficient) for the visualization gradient and the flow direction for flows in Figure 6.4.

the estimated parameters. For an integration length corresponding to 40 output pixels: $C_{lic} = 0.075 \pm 0.001$ and $C_{emb} = 0.216 \pm 0.007$. For an integration length corresponding to 160 output pixels: $C_{lic} = 0.0975 \pm 0.002$ and $C_{emb} = 0.297 \pm 0.019$.

The obtained statistics demonstrate about three times average increase in the mutual information between the visualization gradient and the direction of the original flow. The results for several flow examples are presented in Figure 6.6.

Importantly, our findings concern only the amount of information content in the embedding, but not the accessibility of this information for a human. That is, perception of a particular streamline might be easier from a LIC image than from an unprocessed embedding image. The improvement in this aspect can be achieved by employing common contour-enhancement and edge detection techniques.

Value Information. One common feature of LIC-based flow visualization methods important for the visual tracing of the flow lines is the coherence of the gray value along each line. Thus, different flow line streaks can be distinguished by their average value. However, for LIC this consistency of value if of a local nature. That is, within some neighborhood different streaks are likely to have a large difference in value, but it is not guaranteed within the whole image.

In the presented visualization images, on the contrary, the global value distribution tends to attenuate the local differences between flow lines, while highlighting the variability between regions with different flow behavior. The unique global value consistency property makes the discussed method particularly attractive for 3D rendering, since domain regions of interest can be localized by the selection of a proper value range, thus avoiding occlusion. The local contrast can be enhanced if necessary, with standard image processing methods, such as high-pass filters.

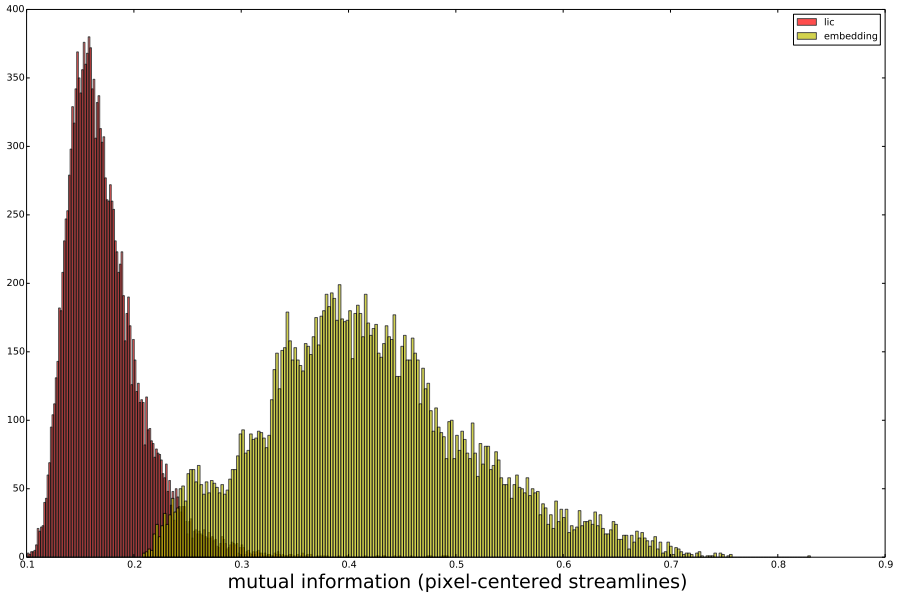

FIGURE 6.7: Yellow: distribution of the uncertainty coefficients between between streamline images and embedding image value over all pixels. Red: distribution of the uncertainty coefficients between streamline images and LIC image value for the same flow (see the corresponding flow visualization in Figure 6.1).

In the following experiment, we measure how much information the image isolines (i.e., lines of a constant value) contain about the flow lines, compared to the corresponding LIC image. For the sake of feasibility, as ground truth we use discrete streamline images, as in Figure 6.1 at the right. Since these images are used in the form of matrix rows as the only input for both LIC and spectral embedding computation, they capture all the available information about the flow. The total number of such images is equal to the number of pixels in each image. We then estimate the mean normalized mutual information between a streamline image and two types of visualization.

Given the set of streamline images α_i, the LIC visualization γ, obtained from the same streamline images, and the embedding visualization (first eigenvector of the Laplacian) β, we compute the following uncertainty coefficients: $C_i^{emb} = \frac{I(\alpha_i,\beta)}{H(\alpha_i)}$ and $C_i^{lic} = \frac{I(\alpha_i,\gamma)}{H(\alpha_i)}$. As a result, we obtain bell-shaped distributions of mutual information over all pixels/streamline images as in Figure 6.7. Sampling the mean of the distributions for 100 random flows allows us to report the following values of the estimated parameters within 99% confidence intervals: $C^{lic} = 0.101 \pm 0.005$ and $C^{emb} = 0.374 \pm 0.01$.

Thus, our statistical tests show with high confidence levels that the value distribution in the spectral embeddings, based on short-term mixture probability, contains on average higher mutual information about input distinct streamline images than the LIC visualization using the same input.

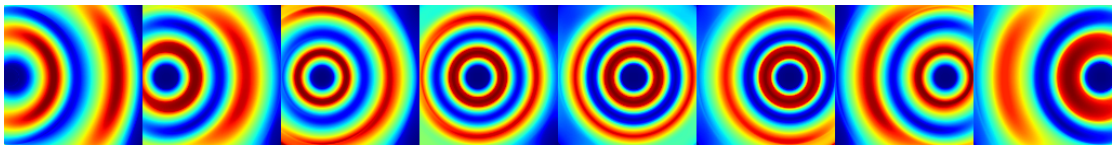

FIGURE 6.8: Visualizations of a circular flow with the center being moved from left to right. For each frame, one spectral embedding (3rd Laplacian eigenvector) is shown using a rainbow color map.

6.8.2 Robustness of the Embeddings Under Flow Modification

One practical question regarding the applicability of the spectral embeddings to visualization or flow feature description is how stable they are under a small change of the underlying vector field. It is especially important in animation: when the flow evolves continuously between time frames, the visualization frames should also be coherent. An empirical observation is that the structure of the Laplacian eigenvectors is not prone to large variations in common visualization scenarios. The theoretical consideration of this effect requires, of course, a proper definition of the "small changes" concept, which might depend on the task at hand.

There is one minor change necessary to enforce coherence between subsequent animation frames. Since the computed eigenvectors are defined only up to a scalar factor, it can happen that they flip sign. That is, $\hat{\mathbf{e}}_i \simeq -\mathbf{e}_i$ where $\mathbf{e}_i$ is the eigenvector of the current flow matrix and $\hat{\mathbf{e}}_i$ is the corresponding eigenvector of the modified flow matrix. In this case, since there is freedom involved in the choice of $\hat{\mathbf{e}}_i$, we can set

$$\hat{\mathbf{e}}_i \leftarrow \operatorname{sign}(\langle \hat{\mathbf{e}}_i, \mathbf{e}_i \rangle)\hat{\mathbf{e}}_i \tag{6.18}$$

That is, the sign of the new frame eigenvector can be flipped if its dot product with the corresponding eigenvector for the previous frame is negative. An example of the visualization of a sequence of flow modifications can be seen in Figure 6.8. Note the consistent coloring of the image cells that have higher probability of mixture (that is, belong to the same streamline or closely connected streamlines).

6.8.3 Embeddings in 3D

The flow spectral embeddings are themselves scalar fields, intuitively featuring the separation of the domain into regions with low intermixing between them. Since these scalar fields are computed for each cell of the entire (discrete) domain, they are volumes rather than surfaces and can be visualized using known volume rendering methods and techniques. Unlike traditional dense flow visualization techniques, there is no dense noise present and certain flow regions can be highlighted with a modification of basic transfer function properties such as transparency and color. In other words, what makes this particular visualization method attractive in 3D is that a local region of interest in the flow corresponds to a certain bounded continuous range in the embedding scalar field values, which makes it easy to select and filter. Some examples are presented in Figure 6.9 and in the supplementary video.

Figure 6.11 demonstrates some of the structures in the embedding volume for the ABC flow, enhanced with user-defined color mapping. Remarkably, the contours of

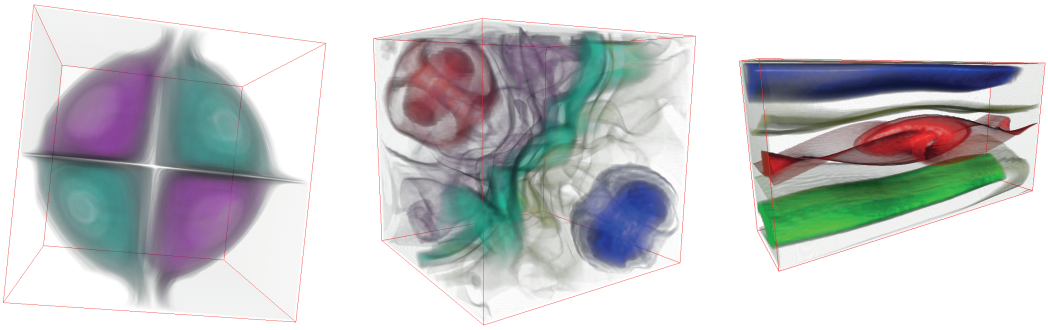

FIGURE 6.9: A visualization of different flows using direct volume rendering of the flow graph embeddings. The datasets from left to right are flow in a spherical drop, Borromean magnetic field, flow around a Stuart vortex. The embedding volumes were computed at $128 \times 128 \times 128$ resolution with a Gaussian kernel of half-length 80 (voxels).

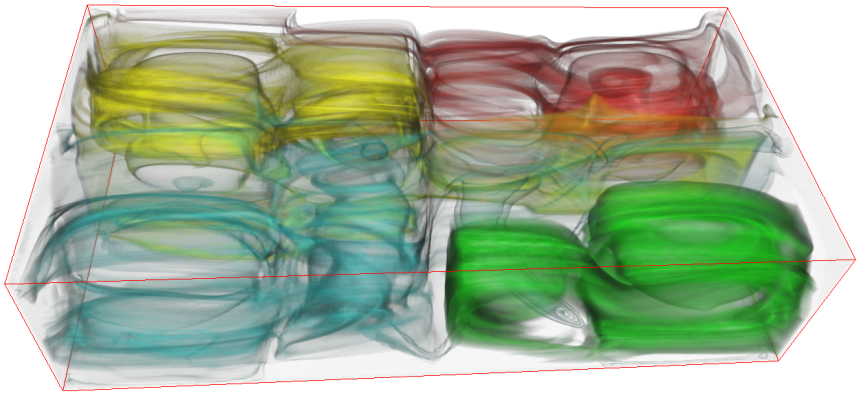

FIGURE 6.10: Direct volume rendering of first two spectral embeddings combined with a color transfer function TF for Benard flow.

the structures are similar to the ridges of the Lyapunov exponential map, shown on the reproduced visualization of Haller [34].

Figure 6.10 reveals four compartments of the Benard flow, separated in the color space with the help of a transfer function applied to the first two embeddings of the flow. The symmetry of the dataset is well captured.

6.8.4 Application To Streamline Clustering

An interesting application of the spectral embeddings is spectral clustering of the cells of the flow domain. Since the computed eigenvectors in 3D are dense volumes, the domain can be segmented based on their value. We consider this technique for streamline segmentation.

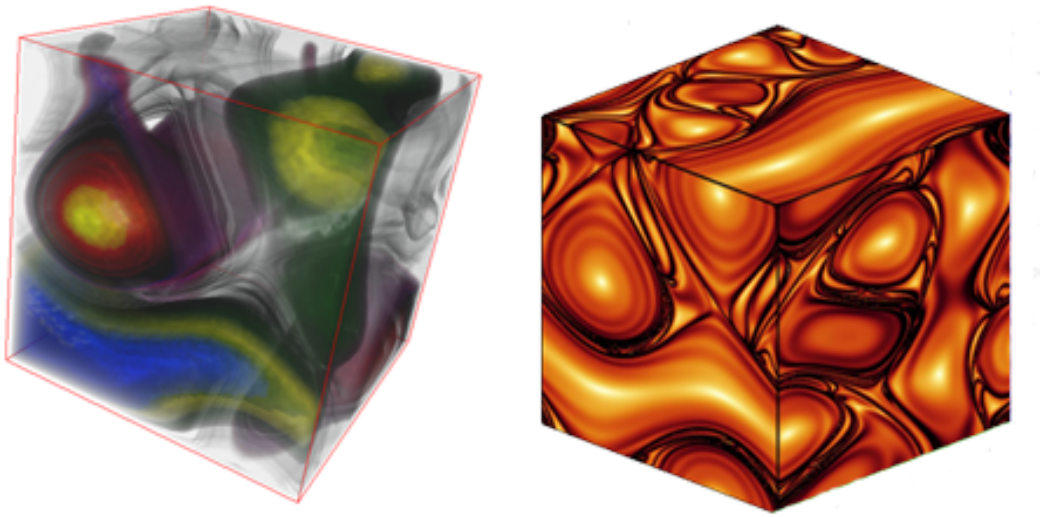

FIGURE 6.11: Direct volume rendering of the first spectral embedding for ABC flow (left), compared to the Lyapunov exponential, reproducing previous visualization of Haller [34] for the same flow.

FIGURE 6.12: Streamline visualizations colored using spectral clustering of the flow domain. The datasets from left to right: ABC flow, Bénard flow, Borromean magnetic field.

Analyzing 3D flows using streamlines is often hindered by occlusion. A possible solution is to reduce the opacity of obscuring streamlines to reveal more interesting patterns behind it. Our goal is to split the flow domain into regions with high probability of mixing. This allows a reduction of visual complexity, by color coding, filtering or grouping streamlines, which might simplify the analysis of the underlying transport processes. Using basic uniform quantization of the eigenvector components we obtain clusters (volume regions) that can be used for grouping streamlines and as opacity and color masks for other traditional visualization techniques.

Figures 6.12 and 6.13 show streamline visualizations of several flows, where the streamlines have been colored according to the segmentation of the domain using uniform quantization of the eigenvectors. Note how distinct the streamlines in one cluster are when compared to the streamlines from other clusters. This clearly shows that the segmentation is aware of the flow features and segments them accordingly. This is especially apparent for the Borromean flow in Figure 6.12: note how the two dominant rings have been clearly separated from the rest of the domain, where an amorphous

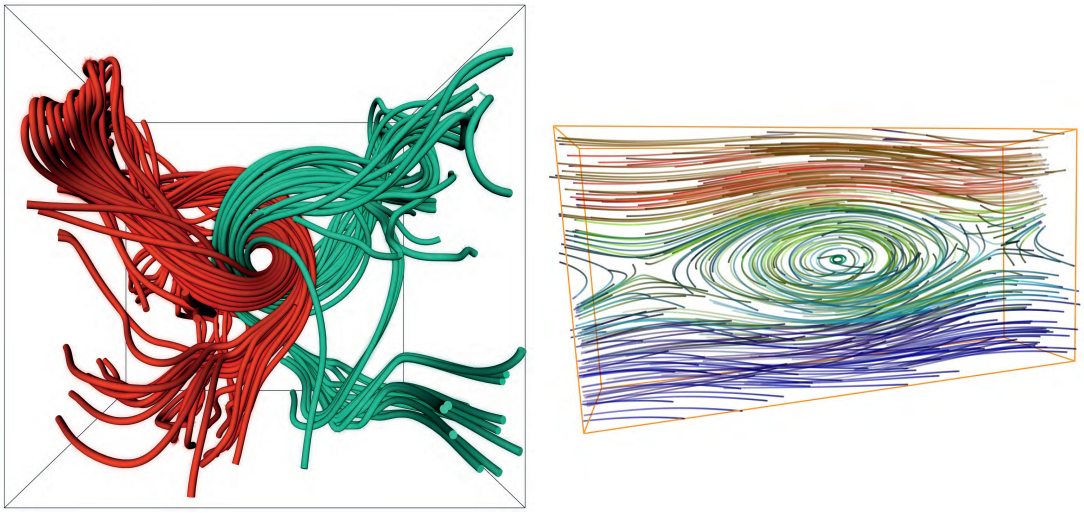

FIGURE 6.13: Streamline visualization of Trefoil magnetic field (left) and Stuart vortex (right) colored with spectral clustering.

behavior prevails.

Figure 6.13 shows a rare, but possible clustering scenario for the Trefoil flow. The two clusters (red and green) are interlocked. This accurately reflects the behavior of the streamlines in this field, where the streamlines form two intertwining rings. Our segmentation is able to capture this correctly.

6.9 Conclusion

In this chapter, we have considered a novel formal approach to well-known methods in dense flow visualization, based on matrix and probability theory. We studied the consequences of this interpretation, applied to LIC, leading to a novel dense flow visualization technique: flow spectral embeddings visualization. We consider the demonstrated results of this method as valuable for visualization practitioners and encouraging for further development of the underlying ideas.

At present, the discussed techniques are verified with a proof-of-concept prototypical implementation, which could reach production software standards with further optimization efforts. A further research direction could be the refinement of technical aspects of the method as well as its extension to other DFV problems. In particular the focus on the multi-scale aspect of the eigenvector represetnation of the flow domain is of practical interest. On the technical side, a straightforward parallel implementation of the eigenvector computation on a GPU can be adopted, in order to decrease the run times. The systematic theoretical treatment of this process might further speed up the computation by means of effective preconditioners for the LOBPCG algorithm.

Chapter 7

Conclusion

This thesis summarizes the author's efforts on exploration of image-synthesis methods for dense flow visualization. Within its scope there were suggested a number of models and techniques for analysis and improvement of the state-of-the-art approaches in this domain. The discussed techniques incorporate ideas from various areas of science and technology: image and signal processing, computer graphics, perception studies, probability and graph theory. The presented models were evaluated by several approaches, using different automated metrics in conjunction with statistical testing and user surveys. The implementations of the discussed methods utilized heavily the commodity hardware for massively parallel processing to cope with sometimes inconveniently large sizes of processed data structures and to achieve acceptable run times.

The presented theoretical framework for discrete modeling of texture-based visualization methods allows to formulate the constraints to the resulting visualization using the apparatus of linear operators and approach the computational problems using well-established algorithms in this domain. This formal view allowed for a intuitive construction of the presented novel method for visualization of 3D vector fields using spectral embeddings.

We started by addressing the question of quality attributes of a flow visualization, making several formal assumptions about the resulting images. We then captured this assumptions within an integral image score and verified its coherence with human perception of the typical images by the means of an extensive expert survey. Based on the the key points underlying the quality score, mainly image spectral characteristics, we developed an approach for direct construction of the visualization with desired properties.

This approach resulted in several novel techniques, each being a refinement or an extension of the predecessor. Firstly, we constructed wave functions with varying frequencies applicable for texture-based visualization of gradient (tangential) vector fields. This idea of explicit frequency control was further adopted for steady and unsteady 2D flows. First, based on the heuristic streamline distance function, and then further refining the concept with the means of perceptually-motivated Gabor filters.

Finally, we generalized the described techniques as a formal probabilistic framework of discrete image filtering. Looking at the problem from the perspective of discrete filters, we demonstrated that many popular dense visualization methods can be described by simple linear operators acting on input images and constructed using the underlying vector fields. A natural development of this idea allowed us to formulate a novel way for flow visualization in 3D using spectral embedding of the resulting graph.

In summary the following techniques were presented:

- gradient-based integral measurement for texture-based flow visualization quality

- contour visualization for gradient/tangential vector field
- wave interference model for flow visualization
- iterated Gabor filters approach to spatial frequency control in texture-based flow visualization
- spectral embedding of flow mixture graph approach for 3D flow visualization

Computationally these techniques were implemented using a diverse set of software development tools, most of them employing GPGPU computing in order to make processing of practical datasets feasible. In particular, there was developed an interactive 2D flow visualization tool for Chapter 2, Chapter 5 and Chapter 4, utilizing OpenGL shaders rendering. For Chapter 2 there was also developed a web-survey system for texture quality metric validation. Tangential flow visualization was implemented as an extension to ImageVis3D[57] software package and as a WebGL-powered web-application [1]. The spectral embeddings computations for Chapter 6 were automated by Python, powered by SciPy stack and accelerated with NVIDIA CUDA platform modules for time-critical computations.

The main contribution of the thesis is the theoretical framework embracing the above techniques as well a number of existing methods. Its foundational ideas are incorporated within the presented visualization methods and can be summarized as:

- discrete formulation of the visualization texture computation
- explicit modeling of constraints on the visualization image
- control of the directional spatial frequency
- relation of the information content of the visualization to the input data

The proposed texture-based visualization techniques cover a variety of possible domains from stationary 2D to time-dependent and 3D flows. The applicability and limitations each individual implementation are discussed in the corresponding chapters. As a directions of future work the author envisions an incorporation of the discussed methods in a visualization system in order to bring their visual data analysis capabilities to the advantage of an end-user. On the theoretical side, although it has been demonstrated that a the discrete probabilistic representation directly allows for the new insights and explanation of existing methods, investigating the question deeper and establishing further connections to the linear operator theory also seems to be fruitful research direction.

This work has been the source for the following publications prepared for major visualization conferences and journals:

- Dense Flow Visualization With Wave Interference (IEEE PACIFICVIS 2012).
- A Metric for the Evaluation of Dense Vector Field Visualizations (IEEE Transactions on Visualization and Computer Graphics, 2013)

[1] (available at `http://matvict.github.io/isocontouring/index.html`)

- Dense Isocontour Imaging (ACM SIGGRAPH Asia 2013)
- Line Integral Convolution With Adaptive Frequency Management (SciVis at IEEE VIS 2015)

Bibliography

[1] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. “The contour spectrum”. In: *Proceedings of the 8th conference on Visualization ’97*. VIS ’97. IEEE Computer Society Press, 1997, 167–ff. ISBN: 1-58113-011-2.

[2] Laurent Balmelli et al. “Volume Warping for Adaptive Isosurface Extraction”. In: *In Proceedings of the conference on Visualization ’02*. IEEE Computer Society, 2002, pp. 467–474.

[3] Henrik Battke, Detlev Stalling, and Hans-Christian Hege. *Fast Line Integral Convolution for Arbitrary Surfaces in 3D*. eng. Tech. rep. SC-96-59. Takustr.7, 14195 Berlin: ZIB, 1997.

[4] B Boashash. “Estimating and interpreting the instantaneous frequency of a signal. I. Fundamentals”. In: *Proceedings of the IEEE* 80.4 (1992), pp. 520–538.

[5] R.P. Botchen, D. Weiskopf, and T. Ertl. “Texture-based visualization of uncertainty in flow fields”. In: *Visualization, 2005. VIS 05. IEEE*. Oct. 2005, pp. 647–654.

[6] Colin Braley et al. “GPU accelerated isosurface volume rendering using depth-based coherence”. In: *ACM SIGGRAPH ASIA 2009 Posters*. SIGGRAPH ASIA ’09. New York, NY, USA: ACM, 2009, 42:1–42:1.

[7] Matthew Brand and Kun Huang. *A Unifying Theorem for Spectral Embedding and Clustering*. 2003.

[8] Luc Buatois, Guillaume Caumon, and Bruno Lévy. “GPU accelerated isosurface extraction on tetrahedral grids”. In: *Proceedings of the Second international conference on Advances in Visual Computing - Volume Part I*. ISVC’06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 383–392. ISBN: 3-540-48628-3, 978-3-540-48628-2.

[9] Brian Cabral and Leith Casey Leedom. “Imaging vector fields using line integral convolution”. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’93. New York, NY, USA: ACM, 1993, pp. 263–270. ISBN: 0-89791-601-8.

[10] Hamish Carr and Jack Snoeyink. “Path seeds and flexible isosurfaces using topology for exploratory visualization”. In: *Proceedings of the symposium on Data visualisation 2003*. VISSYM ’03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 49–58. ISBN: 1-58113-698-6.

[11] Chaomei Chen. “Measuring the quality of network visualization”. In: *Digital Libraries, 2005. JCDL ’05. Proceedings of the 5th ACM/IEEE-CS Joint Conference on*. 2005, p. 405.

[12] Min Chen and Heike Jänicke. “An Information-theoretic Framework for Visualization”. In: *IEEE Trans. Vis. Comput. Graph.* 16.6 (2010), pp. 1206–1215.

[13] Yuan Chen, J.D. Cohen, and J.H. Krolik. “Similarity-Guided Streamline Placement with Error Evaluation”. In: *Visualization and Computer Graphics, IEEE Transactions on* 13.6 (Nov. 2007), pp. 1448–1455. ISSN: 1077-2626.

[14] Yuan Chen, Jonathan Cohen, and Julian Krolik. “Similarity-Guided Streamline Placement with Error Evaluation”. In: *IEEE Transactions on Visualization and Computer Graphics* 13 (6 Nov. 2007), pp. 1448–1455. ISSN: 1077-2626.

[15] Yi-Jen Chiang and Claudio T. Silva. *I/O Optimal Isosurface Extraction*. 1997.

[16] C Chua. “The processing of spatial frequency and orientation information”. In: *Percept. Psychophysics 47, 1 (Jan.)* 1990, pp. 79 –86.

[17] Paolo Cignoni et al. “Speeding up isosurface extraction using interval trees”. In: *IEEE Transactions on Visualization and Computer Graphics* 3 (1997), pp. 158–170.

[18] *Class::DBI*. `http://cpansearch.perl.org/src/TMTM/Class-DBI-v3.0.17/README`. 2011.

[19] *Comprehensive Perl Archive Network*. `http://www.cpan.org/`. 2011.

[20] Robert L. Cook and Tony DeRose. “Wavelet noise”. In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH ’05. New York, NY, USA: ACM, 2005, pp. 803–811.

[21] Patricia Crossno and Edward Angel. “Isosurface Extraction Using Particle Systems”. In: *IEEE Visualization*. 1997, pp. 495–498.

[22] Qingguang Cui et al. “Measuring Data Abstraction Quality in Multiresolution Visualizations”. In: *IEEE Transactions on Visualization and Computer Graphics* 12 (2006), pp. 709–716. ISSN: 1077-2626.

[23] John G. Daugman. “Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters”. In: *J. Opt. Soc. Am. A* 2.7 (July 1985), pp. 1160–1169.

[24] Jeffrey M. DiCarlo and Brian A. Wandell. “Rendering high dynamic range images”. In: *Proc. SPIE 3965, Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications* (2000), pp. 392–401.

[25] Tiago Etiene et al. “Verifiable Visualization for Isosurface Extraction.” In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1227–1234.

[26] Daniel Filonik and Dominikus Baur. “Measuring Aesthetics for Information Visualization”. In: *Information Visualisation, International Conference on* 0 (2009), pp. 579–584.

[27] Andrew Forsberg, Jian Chen, and David Laidlaw. “Comparing 3D Vector Field Visualization Methods: A User Study”. In: *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), pp. 1219–1226. ISSN: 1077-2626.

[28] Lisa K. Forssell. “Visualizing flow over curvilinear grid surfaces using line integral convolution”. In: *Proceedings of the conference on Visualization ’94*. VIS ’94. Los Alamitos, CA, USA: IEEE Computer Society Press, 1994, pp. 240–247. ISBN: 0-7803-2521-4.

[29] Jinzhu Gao and Han-Wei Shen. “Parallel view-dependent isosurface extraction using multi-pass occlusion culling”. In: *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*. PVG ’01. Piscataway, NJ, USA: IEEE Press, 2001, pp. 67–74. ISBN: 0-7803-7223-9.

[30] Harald Garcke et al. “A continuous clustering method for vector fields.” In: *IEEE Visualization*. Mar. 13, 2002, pp. 351–358.

[31] Allen Van Gelder and Jane Wilhelms. “Topological Considerations in Isosurface Generation”. In: *ACM Transactions on Graphics* 13 (1994), pp. 337–375.

[32] Benjamin F. Gregorski et al. “Interactive View-Dependent Rendering Of Large Isosurfaces”. In: *Proceedings of the IEEE Visualization 2002*. IEEE. IEEE, Oct. 2002.

[33] U. Grenander. *Probability and Statistics: The Harald Cramer Volume*. Wiley Publications in Statistics. Alqvist and Wiksell, 1959. URL: `https://books.google.de/books?id=UPc0AAAAMAAJ`.

[34] G. Haller. “Distinguished material surfaces and coherent structures in three-dimensional fluid flows”. In: *Phys. D* (2001), pp. 248–277.

[35] Charles D. Hansen and Paul Hinker. “Massively parallel isosurface extraction”. In: *Proceedings of the 3rd conference on Visualization ’92*. VIS ’92. Los Alamitos, CA, USA: IEEE Computer Society Press, 1992, pp. 77–83. ISBN: 0-8186-2896-0.

[36] B. Heckel et al. “Construction of Vector Field Hierarchies”. In: *Proc. IEEE Visualization ’99*. Ed. by D. Ebert, M. Gross, and B. Hamann. Los Alamitos, 1999, pp. 19–26.

[37] J. Helman and L. Hesselink. “Representation and Display of Vector Field Topology in Fluid Flow Data Sets”. In: *IEEE Computer* 22.8 (Aug. 1989), pp. 27–36.

[38] Marcel Hlawatsch, Filip Sadlo, and Daniel Weiskopf. “Hierarchical Line Integration”. In: *IEEE Transactions on Visualization and Computer Graphics* 99 (2010). ISSN: 1077-2626.

[39] Berthold K. P. Horn and Brian G. Schunck. “Determining Optical Flow”. In: *ARTIFICAL INTELLIGENCE* 17 (1981), pp. 185–203.

[40] Kirsten A. Hoving. *Joseph Cornell and Astronomy: a Case for the Stars*. illustrated. Princeton University Press, 2009. ISBN: 0691134987, 9780691134987.

[41] Victoria Interrante and Chester Grosch. “Strategies for effectively visualizing 3D flow with volume LIC”. In: *Proceedings of the 8th conference on Visualization ’97*. VIS ’97. Los Alamitos, CA, USA: IEEE Computer Society Press, 1997, 421–ff. ISBN: 1-58113-011-2.

[42] Heike Jänicke and Min Chen. “A Salience-based Quality Metric for Visualization”. In: *Computer Graphics Forum* 29.3 (2010), pp. 1183–1192.

[43] Heike Jänicke et al. “Visual Reconstructability as a Quality Metric for Flow Visualization”. In: *Computer Graphics Forum* 30.3 (2011), pp. 781–790. ISSN: 1467-8659.

[44] Jinhee Jeong and Fazle Hussain. “On the identification of a vortex”. In: *Journal of Fluid Mechanics* 285 (Feb. 1995), pp. 69–94. ISSN: 1469-7645.

[45] Bruno Jobard and Wilfrid Lefer. “Creating Evenly-Spaced Streamlines of Arbitrary Density”. In: 1997, pp. 43–56.

[46] Bruno Jobard and Wilfrid Lefer. “Unsteady Flow Visualization by Animating Evenly-Spaced Streamlines”. In: *Computer Graphics Forum*. Vol. 19. 3. Wiley Online Library. 2000, pp. 31–39.

[47] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001.

[48] J Jones and L Palmer. “An evaluation of the two-dimensional Gabor filter model of simple receptive fields in the cat striate cortex”. In: *Journal of Neurophysiology, Vol. 58*. 1987, pp. 1233 –1258.

[49] *JQuery home page*. `http://jquery.com/`. 2011.

[50] Marc Khoury and Rephael Wenger. “On the Fractal Dimension of Isosurfaces”. In: *IEEE Transactions on Visualization and Computer Graphics* 16 (2010), pp. 1198–1205. ISSN: 1077-2626.

[51] Ming-Hoe Kiu and David C. Banks. “Multi-frequency noise for LIC”. In: *Proceedings of the 7th conference on Visualization ’96*. VIS ’96. Los Alamitos, CA, USA: IEEE Computer Society Press, 1996, pp. 121–126.

[52] Joe Kniss, Gordon Kindlmann, and Charles Hansen. “Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets”. In: *Proceedings of the conference on Visualization ’01*. VIS ’01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 255–262. ISBN: 0-7803-7200-X.

[53] A. Knyazev. “Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method”. In: *SIAM Journal on Scientific Computing* 23.2 (2001), pp. 517–541.

[54] Ron Kohavi. “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection”. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1995, pp. 1137–1143.

[55] Martin Kraus. “Scale-invariant volume rendering”. In: *In Proc. of IEEE Visualization*. Academic Press, 2005, pp. 295–302.

[56] Marc Van Kreveld et al. *Contour Trees and Small Seed Sets for Isosurface Traversal*. 1997.

[57] Jens Krüger and Rüdiger Westermann. “Linear algebra operators for GPU implementation of numerical algorithms”. In: *ACM Transactions on Graphics (TOG), Proceedings of SIGGRAPH* 22.3 (2003), pp. 908–916. ISSN: 0730-0301.

[58] Ares Lagae et al. “Procedural Noise using Sparse Gabor Convolution”. In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2009)* 28.3 (2009), 54:1–54:10.

[59] David H. Laidlaw et al. “Comparing 2D Vector Field Visualization Methods: A User Study”. In: *IEEE Transactions on Visualization and Computer Graphics* 11 (1 Jan. 2005), pp. 59–70. ISSN: 1077-2626.

[60] David H. Laidlaw et al. “Quantitative comparative evaluation of 2D vector field visualization methods”. In: *Proceedings of the conference on Visualization ’01*. VIS ’01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 143–150. ISBN: 0-7803-7200-X.

[61] Robert S. Laramee et al. “The state of the art in flow visualization: Dense and texture-based techniques”. In: *Computer Graphics Forum* 23 (2004), pp. 203–221.

[62] Marc Levoy. “Display of Surfaces from Volume Data”. In: *IEEE Comput. Graph. Appl.* 8.3 (May 1988), pp. 29–37. ISSN: 0272-1716.

[63] Zhanping Liu, Robert Moorhead, and Joe Groner. “An Advanced Evenly-Spaced Streamline Placement Algorithm”. In: *IEEE Transactions on Visualization and Computer Graphics* 12 (5 Sept. 2006), pp. 965–972. ISSN: 1077-2626.

[64] Zhiyan Liu, Adam Finkelstein, and Kai Li. “Progressive View-Dependent Isosurface Propagation”. In: *IEEE TCVG Symposium on Visualization (VisSym 2001)*. 2001.

[65] William E. Lorensen and Harvey E. Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *SIGGRAPH Comput. Graph.* 21.4 (Aug. 1987), pp. 163–169. ISSN: 0097-8930.

[66] Ulrike Luxburg. “A Tutorial on Spectral Clustering”. In: *Statistics and Computing* 17.4 (Dec. 2007), pp. 395–416. ISSN: 0960-3174.

[67] S Marcelja. “Mathematical descriptions of the responses of simple cortical cells”. In: *Journal of the Optical Society of America 11 (Nov.)* 70 (1980), pp. 1297 –1300.

[68] Stephen R. Marschner and Richard J. Lobb. “An evaluation of reconstruction filters for volume rendering”. In: *Proceedings of the conference on Visualization ’94*. VIS ’94. Los Alamitos, CA, USA: IEEE Computer Society Press, 1994, pp. 100–107. ISBN: 0-7803-2521-4.

[69] Victor Matvienko and Jens Krüger. “A Metric for the Evaluation of Dense Vector Field Visualizations”. In: *IEEE Transactions on Visualization and Computer Graphics,* 19.7 (July 2013), pp. 1122–1132.

[70] Victor Matvienko and Jens Krüger. “Dense Flow Visualization using Wave Interference”. In: *IEEE Pacific Visusalization*. 2012.

[71] Marina Meila and Jianbo Shi. “Learning Segmentation by Random Walks”. In: *In Advances in Neural Information Processing Systems*. MIT Press, 2001, pp. 873–879.

[72] Paulius Micikevicius. *Optimizing CUDA*. `http://mc.stanford.edu/cgi-bin/images/0/0a/M02_4.pdf`. 2008.

[73] Microsoft. *Parallel LINQ (PLINQ)*. `http://msdn.microsoft.com/en-us/library/dd460688.aspx`. 2011.

[74] Microsoft. *Windows Forms Data Binding*. `http://msdn.microsoft.com/en-us/library/ef2xyb33.aspx`. 2011.

[75] NASA. *NASA Langley Research Center—Multimedia Repository*. `http://lisar.larc.nasa.gov/`.

[76] Rudolf Netzel et al. "Spectral Analysis of Higher-Order and BFECC Texture Advection". In: *Vision, Modeling and Visualization*. Ed. by Michael Goesele et al. The Eurographics Association, 2012. ISBN: 978-3-905673-95-1.

[77] H. C. Nothdurft. "Different Effects From Spatial-frequency Masking in Texture Segregation and Texton Detection Tasks". eng. In: *Vision Research* 31 (2 1991), p. 299. ISSN: 0042-6989.

[78] Arthur Okada and David Lane. "Enhanced Line Integral Convolution with Flow Feature Detection". In: *In SPIE Vol. 3017 Visual Data Exploration and Analysis IV*. 1997, pp. 206–217.

[79] Rodolfo S. Allendes Osorio and Ken W. Brodlie. "Uncertain Flow Visualization using LIC". In: *TPCG*. Ed. by Wen Tang and John P. Collomosse. Eurographics Association, 2009, pp. 215–222. ISBN: 978-3-905673-71-5.

[80] Sung W. Park et al. "Structure-accentuating Dense Flow Visualization." In: *EuroVis*. Ed. by Beatriz Sousa Santos, Thomas Ertl, and Kenneth I. Joy. Eurographics Association, Nov. 5, 2007, pp. 163–170. ISBN: 3-905673-31-2.

[81] Vladimir Pekar, Rafael Wiemker, and Daniel Hempel. "Fast detection of meaningful isosurfaces for volume data visualization". In: *Proceedings of the conference on Visualization '01*. VIS '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 223–230. ISBN: 0-7803-7200-X.

[82] Daniel Pineo and Colin Ware. "Neural modeling of flow rendering effectiveness". In: *Proceedings of the 5th symposium on Applied perception in graphics and visualization*. APGV '08. New York, NY, USA: ACM, 2008, pp. 171–178. ISBN: 978-1-59593-981-4.

[83] Andreas Pommert, Ulf Tiede, and Karl Heinz Höhne. "On the Accuracy of Isosurfaces in Tomographic Volume Visualization". In: *Proceedings of the 5th International Conference on Medical Image Computing and Computer-Assisted Intervention-Part II*. MICCAI '02. London, UK, UK: Springer-Verlag, 2002, pp. 623–630. ISBN: 3-540-44225-1.

[84] M. Porat and Y.Y. Zeevi. "Localized texture processing in vision: analysis and synthesis in the Gaborian space". In: *Biomedical Engineering, IEEE Transactions on* 36.1 (Jan. 1989), pp. 115–129. ISSN: 0018-9294.

[85] Frits H. Post and Theo Van Walsum. "Fluid Flow Visualization". In: *Focus on Scientific Visualization*. Springer, 1993, pp. 1–40.

[86] Frits H. Post et al. "The State of the Art in Flow Visualisation: Feature Extraction and Tracking". In: *Computer Graphics Forum* 22.4 (2003), pp. 775–792. ISSN: 1467-8659.

[87] William H. Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. New York, NY, USA: Cambridge University Press, 2007. ISBN: 0521880688, 9780521880688.

[88] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Second Edition. Texts in Applied Mathematics. Berlin, Heidelberg: Springer, 2007. ISBN: 3-540-34658-9.

[89] Wieland Reich and Gerik Scheuermann. "Analysis of Streamline Separation at Infinity Using Time-Discrete Markov Chains". In: *IEEE Transactions on Visualization and Computer Graphics* 18 (2012), p. 9.

[90] J. Reininghaus and I. Hotz. "Combinatorial 2D Vector Field Topology Extraction and Simplification". In: *Topological Methods in Data Analysis and Visualization*. Ed. by V. Pascucci et al. Mathematics and Visualization. Springer, 2011, pp. 103–114. ISBN: 978-3-642-15013-5.

[91] C. Rezk-Salama et al. "Interactive exploration of volume line integral convolution based on 3D-texture mapping". In: *Proceedings of the conference on Visualization '99: celebrating ten years*. VIS '99. Los Alamitos, CA, USA: IEEE Computer Society Press, 1999, pp. 233–240. ISBN: 0-7803-5897-X.

[92] J.G. Robson. "Spatial and Temporal Contrast Sensitivity Functions of The Visual Systems". In: *Journal of the Optical Society of America* 56 (1966), pp. 1141–1142.

[93] C. Rössl and H. Theisel. "Streamline Embedding for 3D Vector Field Exploration". In: *IEEE Transactions on Visualiztion and Computer Graphics* 18-3 (2012). fast track TVCG from IEEE Visualization 2010, pp. 407–420.

[94] Tobias Salzbrunn and Gerik Scheuermann. "Streamline Predicates". In: *IEEE Transactions on Visualization and Computer Graphics* 12.6 (2006), pp. 1601–1612. ISSN: 1077-2626.

[95] Tobias Salzbrunn et al. "The State of the Art in Flow Visualization: Partition-Based Techniques". In: *SimVis*. 2008, pp. 75–92.

[96] Tobias Salzbrunn et al. "The state of the art in flow visualization: Partition-based techniques". In: *In Simulation and Visualization 2008 Proceedings*. 2008.

[97] A. Sanna et al. "Adding a scalar value to texture-based vector field representations by local contrast analysis". In: *Proceedings of the symposium on Data Visualisation 2002*. VISSYM '02. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2002, pp. 35–41. ISBN: 1-58113-536-X.

[98] Peter Selinger. *Potrace: a polygon-based tracing algorithm*. 2003.

[99] Raj Shekhar et al. "Octree-based decimation of marching cubes surfaces". In: *Proceedings of the 7th conference on Visualization '96*. VIS '96. Los Alamitos, CA, USA: IEEE Computer Society Press, 1996, 335–ff. ISBN: 0-89791-864-9.

[100] H.W. Shen, C.R. Johnson, and K.L. Ma. "Visualizing vector fields using line integral convolution and dye advection". In: *Volume Visualization, 1996. Proceedings., 1996 Symposium on*. IEEE. 1996, pp. 63–70.

[101] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (1997), pp. 888–905.

[102] *SQLite documentation*. `http://sqlite.org/`. 2011.

[103] D. Stalling and H. Hege. “Fast and Resolution Independent Line Integral Convolution”. In: *Proceedings Siggraph ’95* (1995). Los Angeles, pp. 249–256.

[104] Detlev Stalling. “Fast Texture-Based Algorithms for Vector Field Visualization”. PhD thesis. Zuse Institute Berlin, 1998.

[105] Detlev Stalling and Hans-Christian Hege. “Fast and resolution independent line integral convolution”. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’95. New York, NY, USA: ACM, 1995, pp. 249–256. ISBN: 0-89791-701-4.

[106] Detlev Stalling and Hans-Christian Hege. “LIC on Surfaces”. In: *Texture Synthesis with Line Integral Convolution, Siggraph 97, 24. Int. Conf. Computer Graphics and Interactive Techniques*. 1997, pp. 51 –64.

[107] A. Sundquist. “Dynamic line integral convolution for visualizing streamline evolution”. In: *Visualization and Computer Graphics, IEEE Transactions on* 9.3 (July 2003), pp. 273–282. ISSN: 1077-2626.

[108] Andrzej Szymczak and Levente Sipeki. “Visualization of Morse Connection Graphs for Topologically Rich 2D Vector Fields”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2763–2772.

[109] Shigeo Takahashi, Yuriko Takeshima, and Issei Fujishiro. “Topological volume skeletonization and its application to transfer function design”. In: *Graph. Models* 66.1 (Jan. 2004), pp. 24–49. ISSN: 1524-0703.

[110] Shigeo Takahashi et al. “Emphasizing Isosurface Embeddings in Direct Volume Rendering”. In: *Scientific Visualization: The Visual Extraction of Knowledge from Data*. Ed. by Georges-Pierre Bonneau et al. Mathematics and Visualization. Springer Berlin Heidelberg, 2006, pp. 185–206.

[111] Francesca Taponecco and Marc Alexa. “Vector field visualization using Markov Random Field texture synthesis”. In: *Proceedings of the symposium on Data visualisation 2003*. VISSYM ’03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 195–202. ISBN: 1-58113-698-6.

[112] Natalya Tatarchuk, Jeremy Shopf, and Christopher DeCoro. “Real-Time Isosurface Extraction Using the GPU Programmable Geometry Pipeline”. In: *ACM SIGGRAPH 2007 courses*. SIGGRAPH ’07. New York, NY, USA: ACM, 2007, pp. 122–137. ISBN: 978-1-4503-1823-5.

[113] A. Telea and J.J. van Wijk. “Simplified representation of Vector Fields”. In: *Proc. IEEE Visualization ’99*. Ed. by D. Ebert, M. Gross, and B. Hamann. Los Alamitos, 1999, pp. 35–42.

[114] *Template Toolkit home page*. `http://template-toolkit.org/`. 2011.

[115] Shivaraj Tenginakai, Jinho Lee, and Raghu Machiraju. “Salient iso-surface detection with model-independent statistical signatures”. In: *Proceedings of the conference on Visualization ’01*. VIS ’01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 231–238. ISBN: 0-7803-7200-X.

[116] *Thrust: Code at the Speed of Light*. `http://code.google.com/p/thrust/`. 2011.

[117] G. M. Treece, R. W. Prager, and A. H. Gee. “Regularised marching tetrahedra: improved iso-surface extraction”. In: *Computers and Graphics* 23 (1999), pp. 583–598.

[118] X. Tricoche, G. Scheuermann, and H. Hagen. “Continuous topology simplification of planar vector fields”. In: *Proc. Visualization*. 2001, pp. 159 –166.

[119] Greg Turk and David Banks. *Image-Guided Streamline Placement*. 1996.

[120] Timothy Urness et al. “Effectively Visualizing Multi-Valued Flow Data Using Color and Texture”. In: *Proceedings of the 14th IEEE Visualization 2003 (VIS’03)*. VIS ’03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 16–. ISBN: 0-7695-2030-8.

[121] Vivek Verma, David Kao, and Alex Pang. “A Flow-guided Streamline Seeding Strategy”. In: *In Proceedings IEEE Visualization 2000*. 2000, pp. 163–170.

[122] Vivek Verma, David Kao, and Alex Pang. “PLIC: Bridging the Gap Between Streamlines and LIC”. In: *In Proceedings of Visualization ?99*. IEEE Computer Society Press, 1999, pp. 341–348.

[123] Colin Ware and William Knight. “Using Visual Texture for Information Display”. In: *ACM Trans. Graph.* 14.1 (1995), pp. 3–20.

[124] Gunther H. Weber et al. “Exploring scalar fields using critical isovalues”. In: *Proceedings of the conference on Visualization ’02*. VIS ’02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 171–178. ISBN: 0-7803-7498-3.

[125] R. Wegenkittl, H. Löffelmann, and E. Gröller. “Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet”. In: *Proc. IEEE Visualization ’97*. Ed. by R. Yagel and H. Hagen. 1997, pp. 309–316.

[126] T. Weinkauf et al. “Extracting Higher Order Critical Points and Topological Simplification of 3D Vector Fields”. In: *Proc. IEEE Visualization 2005*. 2005, pp. 559–566.

[127] Daniel Weiskopf. *GPU-Based Interactive Visualization Techniques (Mathematics and Visualization)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 3540332626.

[128] Daniel Weiskopf. “Iterative Twofold Line Integral Convolution for Texture-Based Vector Field Visualization”. In: *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Mathematics and Visualization. Springer Berlin Heidelberg, 2009, pp. 191–211. ISBN: 978-3-540-49926-8.

[129] Daniel Weiskopf, Gordon Erlebacher, and Thomas Ertl. “A Texture-Based Framework for Spacetime-Coherent Visualization of Time-Dependent Vector Fields”. In: *Proceedings of the 14th IEEE Visualization 2003 (VIS’03)*. VIS ’03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 15–. ISBN: 0-7695-2030-8.

[130] Jarke J. van Wijk. “Image based flow visualization”. In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’02. New York, NY, USA: ACM, 2002, pp. 745–754. ISBN: 1-58113-521-1.

[131] Jarke J. van Wijk. “Spot noise texture synthesis for data visualization”. In: *SIGGRAPH Comput. Graph.* 25 (4 July 1991), pp. 309–318. ISSN: 0097-8930.

[132] Jarke J. van Wijk. “The value of visualization”. In: *Visualization, 2005. VIS 05. IEEE*. 2005, pp. 79 –86.

[133] Jarke J. van Wijk. “Image Based Flow Visualization for Curved Surfaces”. In: *Proceedings of the 14th IEEE Visualization 2003 (VIS’03)*. VIS ’03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 17–. ISBN: 0-7695-2030-8.

[134] Wikipedia. *Occam’s Razor*. `http://en.wikipedia.org/wiki/Occam's_razor`. 2011.

[135] Keqin Wu et al. “Topology-Aware Evenly Spaced Streamline Placement”. In: *IEEE Transactions on Visualization and Computer Graphics* 16 (5 Sept. 2010), pp. 791–801. ISSN: 1077-2626.

[136] Chih-Kuo Yeh et al. “Animating streamlines with orthogonal advancing waves”. In: *Information Visualization* 12.3-4 (2013), pp. 257–272.

[137] Qizhi Yu et al. “Lagrangian Texture Advection: Preserving both Spectrum and Velocity Field”. In: *IEEE Transactions on Visualization and Computer Graphics* (2010).

[138] Huijuan Zhang and Timothy S. Newman. “Efficient Parallel Out-of-core Isosurface Extraction”. In: *Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*. PVG ’03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 3–. ISBN: 0-7695-2091-X.

[139] Jianlong Zhou and Masahiro Takatsuka. “Automatic Transfer Function Generation Using Contour Tree Controlled Residue Flow Model and Color Harmonics”. In: *IEEE Transactions on Visualization and Computer Graphics* 15 (2009), pp. 1481–1488. ISSN: 1077-2626.