UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a doctoral dissertation by

Timothy Matthew Urness

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Victoria Interrante
Name of Faculty Adviser

Signature of Faculty Adviser

Date

GRADUATE SCHOOL

Texture-Based Visualization of Multi-Field Flow Data

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Timothy Matthew Urness

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Victoria Interrante, Adviser

May 2006

Acknowledgements

First and foremost, I express my sincere gratitude to my adviser, Victoria Interrante. Vicki's advice, wisdom, guidance, and leadership have been instrumental to my success throughout my graduate school career at the University of Minnesota. I am particularly appreciative of her encouragement in the development of my teaching ability through teaching assistantships and adjunct professor positions. Vicki has done a wonderful job of guiding my research, education, and experiences while allowing me to develop my passion for education.

There are several individuals who have had a profound impact in my decision to pursue a Ph.D. and to undertake scientific visualization as a research area. Steven Senger asked me to be a part of an undergraduate research project at the University of Wisconsin – La Crosse while I was sophomore. This is where my interest in computer graphics and scientific visualization began, and I am very thankful to Steve for giving me the chance to work alongside him while I was still learning the basics of computer science. My experience at UW-La Crosse paved the way for my application to the summer undergraduate research program at Argonne National Laboratory. This is where I met Lori Freitag, my mentor at Argonne. The experience at Argonne taught me about being a professional computer scientist, and it also resulted in my very first publication. These first few experiences during my undergraduate life were foundational in my decision to pursue a Ph.D. I like to think that my previous research experiences allowed for my application to catch the attention of the admission committee at the University of Minnesota and my future adviser, Vicki.

Several other professors at UW-La Crosse have been instrumental in my education. One in particular, Karen Sutherland, has always been one of my biggest supporters. Her mentoring and advice, both during my undergraduate and graduate school education, has been very influential in my academic career.

In assembling this dissertation, I received invaluable assistance from my committee: Ellen

Longmire, John Carlis, Gary Meyer, and Vicki Interrante. Not only was their expert advice crucial to this volume, but without their support and encouragement, I would never have been able to complete it.

Numerous people have made my experience at the University of Minnesota more enjoyable and productive. Professors Gary Meyer and Baoquan Chen, as well as the graphics group at the University of Minnesota, deserve a tremendous amount of credit. Present and past members of the group who have been particularly involved in my research projects include: Gabriele Gorla, Jason Gott, Hae Young Kim, Sunny Kim, Kirti Kesavarapu, Matt Heinzen, Gary Dahl, Minh X. Nguyen, Lijun Qu, Brian Ries, Amit Shesh, Nathan Gossett, Clement Shimizu, Hui Xu, Xiaoru Yuan, P. Coleman Saunders, K. Evan Nowak, and Haleh Hagh-Shenas. Collaborations with researchers from the department of Aerospace Engineering and Mechanics and the department of Astronomy have resulted in several successful visualizations. I would especially like to thank Ivan Marusic, Ellen Longmire, Bharathram Ganapathisubramani, Nicholas Hutchins, Blayne Field, Sean O'Neill and Thomas W. Jones.

This research is ultimately due to my parents, John and Ellen Urness. I have never been able to adequately express my appreciation for all that they have done to provide for my upbringing. They have held a household that is safe, loving, secure, and that encouraged education, curiosity, and discovery. I would also like to mention the love and support of my two brothers, Mark and Daniel, and the best sister-in-law/editor one could ask for, Heather.

Lastly, I would like to acknowledge my lovely wife Kristin. She has been the inspiration for it all.

This work has been supported by National Science Foundation grants (ACI-9982274, CTS-0324898), the University of Minnesota Digital Technology Center, the University of Minnesota Graduate School Fellowship, and the University of Minnesota Doctoral Dissertation Fellowship.

Abstract

Researchers have long been interested in developing a deeper understanding of the key physical mechanisms in fluid dynamics. Of particular interest are the complex relationships between the multiple scalar and vector quantities used to characterize scientific phenomena within the flow. Efforts to achieve a fundamental understanding of these key physical mechanisms remain limited mainly because of a lack of understanding of the nonlinear interactions that occur among the components of the flow.

The goal through this work is to enable researchers to obtain a succinct, meaningful visual summary of the contents of a dataset that consists of multiple, coincident variables. This is accomplished through providing techniques that allow the creation of an image in which the important features of multiple scalar or vector fields can be understood both independently and in the context of the other fields.

This research offers several new techniques for effectively using color and texture to simultaneously convey information about multiple co-located scalar and vector distributions. Specifically, we introduce:

- *color weaving*, an alternative to traditional color compositing for simultaneously representing multiple distributions by allowing colors to be closely interwoven via the assignment of distinct separate hues to individual streamlines
- applying natural textures to streamlines to create a richly diverse set of possibilities for the visualization of multiple distributions within a flow
- embossing to encode the out-of-plane component of a 3D vector field defined over a 2D domain
- visualizing multiple 2D vector fields using strategies involving layers of disparate textures and overlapping streamlines

These methods ultimately enhance the ability to provide insight into the complicated interactions that occur within multi-field flow data.

Contents

List of Figures

Chapter 1

Introduction

The field of scientific visualization focuses on creating images that convey salient information about underlying data [22]. The visualization of data makes it possible for researchers, analysis, engineers, and the lay audience to obtain insight in these data in an efficient and effective way, thanks to the unique capabilities of the human visual system, which enable us to detect interesting features and patterns in a short time [82].

The way information is displayed has a profound impact on how it is interpreted. Like mathematics, the principles of the design of images is universal, and are independent of any specific language or culture. For these reasons, it is important to analyze the method of information visualization, and create images that accurately reflect the data represented [72, 73, 74].

1.1 Motivation

The goal of scientific visualization is to represent information in a manner that is easy to interpret, accurate, and lends itself to a fundamental understanding of the underlying organization of the information being displayed. Whether consciously considered or

not, the process of evaluating and analyzing scientific visualization images relies upon human visual perception and aesthetics [57]. To produce an image that successfully reflects multi-field data it is necessary not only to be accurate in the representation of individual distributions, but also to portray each specific component in a way that does not interfere with the accurate perception of the other components. This process of successfully combining variables has important applications in analyzing the scientific phenomena represented by multi-field data.

A multi-field dataset is a collection of data in which several related distributions exists at the same location. A dataset that includes pressure and temperature at each sampling point is an example of a multi-field data set. Current scientific experiments and computational models are capable of generating a wealth of information; much of this data includes multiple coincident variables. What has become apparent through the collection of this data is the difficulty involved in extracting important information from these datasets, and the lack of adequate tools that are available to efficiently analyze them. In this dissertation, we discuss several different visualization techniques for this purpose. Our ultimate goal through this work is to enable researchers to obtain a succinct, meaningful visual summary of the contents of a dataset through providing techniques that allow the creation of images in which the important features of multiple distributions can be understood both independently and in the context of multiple other distributions.

1.2 Overview

This dissertation presents a combination of methods that approach the task of multi-field flow visualization.

Chapter 2 provides three different kinds of background information: visualization (section 2.1), fluid dynamics (section 2.2), and astronomy (2.3). This chapter is important because it

provides the foundation of work that the contributions presented in the dissertation are built from. The visualization section reviews the relevant work done in the flow visualization and multi-field visualization communities. A background in the application areas of fluid dynamics and astronomy is included to assist the reader in following the discussion of the images throughout the dissertation. While the algorithms developed are not limited to only applications in astronomy or fluid dynamics, we have applied the techniques to very a very specific set of problems in these fields. We have included these sections to give a background of the problems challenging scientists, establish a working vocabulary, and define the terms used throughout this volume.

Chapter 3 addresses several visualization techniques designed to graphically depict components of flow. We illustrate direct and texture-based flow, explain how the relative velocity of the observer effects flow visualizations, and describe methods to depict direction of flow within a static image. We also discuss the difference between streamlines, streaklines, and pathlines.

Chapter 4 presents different techniques for the visualization of scalar fields within flow images. Effective color use is motivated by the introduction of a novel technique that provides an alternative to traditional color compositing. We also discuss the role of 3D graphics used to depict a scalar value. The perceptual limitations that prevent individual techniques representing scalar values to be effectively combined is illustrated through the combination of contrast and luminance. This leads to a technique, based on embossing, that has been developed to encode the out-of-plane component of a 3D vector field over a 2D domain.

Chapter 5 describes the approach of using textures to provide a consistent and highly-detailed visualization of multi-field flow data. We discuss how textures can be geometrically mapped to streamlines and used to represent flow fields. We demonstrate how texture attributes and multiple textures can be utilized to represent various components

of the data.

Chapter 6 presents strategies for the visual representation of multiple vector fields. We explore the range of effects that can be obtained by combining several existing flow visualization techniques for the purposes of analyzing multiple vector fields. We apply the insights gained by the experiments to three different scientific applications.

We conclude in chapter 7 and discuss future work.

1.3 Contributions

The major contributions of the dissertation can be summarized as:

- Colorweaving (section 4.1.3) [76]. This technique provides an alternative to traditional color compositing in flow visualizations by allowing multiple colors to be closely interwoven via the assignment of distinct separate hues to individual streamlines, rather than blended.
- Contrast and mean luminance analysis (section 4.3) [77]. We rigorously examine the abilities and limitations of using contrast and mean luminance in the representation of scalar values within a flow field.
- Embossing to represent scalar data (section 4.5) [77]. This technique utilizes an embossing algorithm to encode the out-of-plane component of a 3D vector field defined over a 2D domain.
- Flow visualization using natural textures (section 5.1). We present methods that utilize the qualities and attributes of natural textures to visualize multiple scalar distributions and multiple vector fields obtained across a 2D domain in a turbulent boundary layer flow.
- Strategies for the visualization of multiple 2D vector fields (section 6.1) [78]. We

discuss strategies for effectively visualizing co-located 2D vector fields, enabling the key physical structures of one vector field to be clearly understood within the context of a related vector field.

Chapter 2

Background

The material in this chapter serves as a background for the research in the remainder of the dissertation. The primary goal of our research is to produce effective visualization tools for multi-field data.

The techniques introduced in this dissertation have been specifically applied to experimental or computationally-generated data supplied by researchers in the Department of Astronomy and the Department of Aerospace Engineering and Mechanics at the University of Minnesota. This interaction between visualization researchers and domain scientists has proven extremely beneficial to both parties. Through our collaboration we have both advanced our ability to create effective visualizations and made important discoveries that further the development of important theories related to the applications. The application scientists played a critical role in defining the specific needs that the visualization techniques presented here were developed to address. In addition, they provided an objective assessment of the functionality of the methods, in terms of how well they meet their goals of gaining greater insight into their data.

Often times, a challenging component to this research is developing a common vocabulary with the scientists in order to properly communicate *what* are the key physical structures

that need to be visualized in addition to *how* to go about visualizing the data.

In this chapter, we describe the basic fundamentals of fluid dynamics and astronomy necessary to understand the problems inherent with the multi-field data in each of these domains. Furthermore, we examine the background of multi-field visualization and flow visualization techniques.

Section 2.1 provides a background in techniques developed to visualize flow and multi-valued data. We focus on texture-based techniques designed to depict a vector field in addition to other coincident variables. Section 2.2 summarizes the principles of fluid dynamics that are relevant to the collaboration with our colleagues from the Department of Aerospace and Mechanics at the University of Minnesota. We explain the properties of turbulent flow and the mathematically derived fields that are important to characterizing regions of interest within the data. We also describe the experimental setup to collect data and a numeral simulation of turbulent flow. Section 2.3 briefly explains the research currently conducted in the field of computational physics – namely in modeling astrophysical jets. We supply an introduction and explanation of the multi-field data generated by these simulations.

This dissertation builds upon the vast amount of literature in scientific visualization – particularly in the fields of flow visualization and multi-field visualization. In the next section, we review many of the seminal works in these categories.

2.1 Flow Visualization

Flow Visualization, a classic subfield of scientific visualization, is a richly diverse field that has applications in a variety of industries including: aerospace, automotive transportation, chemical processing, weather simulation, climate modeling, and medical visualization. Thus, the variety of solutions for flow visualization applications demands a diverse range

of techniques. In the following, many of the existing flow visualization algorithms are categorized and discussed in general before the literature is reviewed.

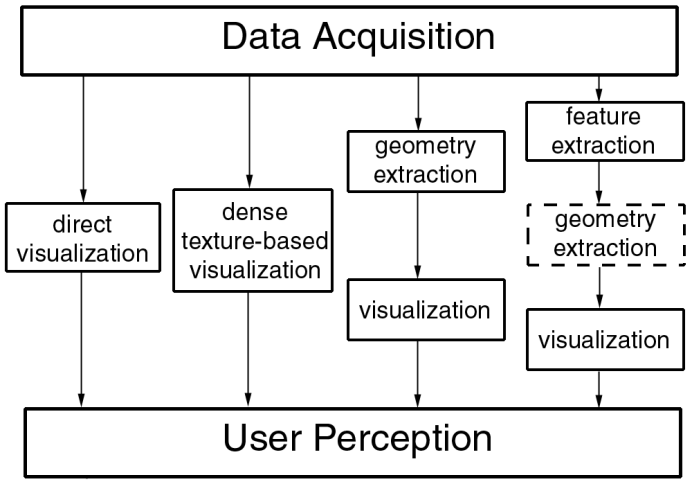

Figure 2.1: Classification of flow visualization techniques based on [46] – from left to right: direct, texture-based, based on geometric objects, and feature-based.

2.1.1 Classification

In a state of the art report on dense and texture-based flow visualization, Laramee et al. [46] classify flow visualization into several different sub-categories depending on the need of the user. These categories include direct flow visualization, texture-based flow visualization, geometric flow visualization, and feature-based flow visualization.

Direct flow visualization

This category uses a technique that directly transfers the data to an image in the simplest manner possible. These techniques provide an intuitive representation of the data and allow immediate investigation of the flow field. Color coding an additional scalar attribute, such as temperature or pressure, would be considered a direct flow visualization technique. Perhaps the most straightforward approach to presenting a vector field is to use a serious of glyphs known as vector plots or hedgehogs [42].

Texture-based flow visualization

This category uses a computed texture to construct a dense representation of the flow. The use of texture allows for a consistent, highly-detailed, and correlated representation of a vector field. In most cases, the vector field is used to determine the resulting texture through the filtering of pixel values.

Geometric flow visualization

This category includes techniques that first identify geometric objects as a basis for the flow. Examples of these objects include streamlines, streaklines, and pathlines. These objects are typically computed by using an integration-based approach and are used to depict behavior induced by the flow dynamics [52].

Feature-based flow visualization

This category utilizes a two-step approach to visualize special or important features of the data. First, important phenomena or topological information is extracted from the data set. Then, the visualization is performed on the extracted information – allowing for compact and time-efficient visualizations of potentially very large data sets [53].

The algorithms that belong to the dense and texture-based flow visualization category comprise a suitable background for the flow visualization component of the dissertation. In the next section, we examine the classic texture-based flow visualization techniques.

2.1.2 Texture-Based Flow Visualization

Textures have traditionally been used to visualize vector fields for the purpose of analyzing the form and behavior of flow consistent with theoretical models and to infer the underlying

behavior of experimentally-generated flow fields. The use of texture allows for a consistent and highly-detailed representation of a vector field allowing an observer to both analyze and better understand the dynamics of fluid flow.

Spot Noise – Spot Noise, developed by Van Wijk [79], was one of the first texture-based algorithms used to display a flow field. In this algorithm, a texture is produced from weighted and randomly positioned spots deformed in accordance with the direction of flow. A spot noise texture is defined by:

$$f(\vec{x}) = \sum a_i h(\vec{x} - \vec{x}_i)$$

where $h(\vec{x})$ is called the intensity function, a_i is a scaling factor, and x_i is a random position.

The original contribution of spot noise did not accurately reflect high, local velocity curvature. Enhanced Spot Noise by De Leeuw and Van Wijk [8] use curved spot primitives to address this problem.

One advantage of the spot noise algorithm is that it allows for the depiction of velocity magnitude through the deformation of the spots. A comparison between spot noise and LIC (the subject of the next review) conducted by de Leeuw and van Liere [7] states that the standard LIC algorithm uses normalized vectors and does not allow for the perception of velocity magnitude; however, LIC does provide a better sense of flow direction than spot noise.

Line Integral Convolution – *Line Integral Convolution* (LIC) was first introduced by Cabral and Leedom [4]. A widely-used and versatile algorithm, the original algorithm takes as inputs a vector field defined on a Cartesian grid and a white noise input texture. Texels are convolved along calculated streamlines, using a filter kernel, to produce an output texture that is highly correlated in the flow direction. Specifically, given a streamline σ, LIC calculates the output texture by calculating the intensity I for a pixel located at $x_0 = \sigma(s_0)$ by

$$I(x_0) = \int_{s_0-L/2}^{s_0+L/2} k(s-s_0)T(\sigma(s))ds$$

where T stands for an input noise texture, k denotes the filter kernel, s is an arc length used to parameterize the streamline curve, and L represents the filter kernel length.

Fast LIC – Stalling and Hege [66] achieved an order of magnitude increase in the efficiency of the LIC algorithm by taking advantage of coherence along streamlines and increased the fidelity of the resulting output image by using a fourth-order Runge-Kutta method for streamline calculations. This results in the computation of the output texture being streamline oriented, not pixel oriented.

LIC with Normal – Scheuermann et al. [62] combined surface deformation with LIC to visualize three-dimensional vector fields defined on a two-dimensional manifold. The normal component is used to deform the manifold and is rendered as a three-dimensional scene. A similar approach presented by Sanna and Montrucchio [59] uses bump mapping to encode an arbitrary additional scalar variable over a vector field.

LIC with Flow Direction – One disadvantage of traditional LIC images is that the direction of movement in a flow is ambiguous. Animation can be used to make that information explicit [4, 33, 66]. Wegenkittl, Groller, and Purgathofer [87] introduced a technique called Oriented Line Integral Convolution (OLIC) that addresses this issue in a single static image. The OLIC algorithm, in essence, uses a sparse texture resembling ink droplets on a page as input and a ramp-like convolution kernel smears the droplets according to the vector field, resulting in a collection of streaks in which intensity increases from tail to head. Computation time for this method was significantly reduced with the introduction of Fast Oriented Line Integral Convolution (FROLIC) [86]. More recently, in another approach similar to OLIC, Sanna et al. [60] propose a Thick Oriented Stream Lines (TOSL) method,

in which the orientation of a flow is depicted by increasing the luminance along calculated streamlines.

Multi-frequency LIC – Kiu and Banks [41] illustrate how using a multi-frequency noise texture can be used to demonstrate components of a flow field. Using multi-frequency input textures along with increased filter kernel lengths allow the resulting image to incorporate indications of velocity magnitude.

Texture Mapping Flow – Verma, Kao, and Pang presented a method for generating LIC-like images through texture-mapped streamlines called PLIC (Psuedo-LIC) [83]. By experimenting with different input textures, both LIC-like images and streamline-like images can be produced. Taponecco and Alexa [68] introduced an algorithm that used a Markov model approach to synthesize a texture according to a flow field. This technique uses a strongly directional texture that is rotated according to the vector field. The resulting image is created in a pixel-by-pixel manner using Markov Random Field texture synthesis. Shen, Li, and Bordoloi [65] utilize texture mapping in addition to analysis of three dimensional geometric properties to volume render three-dimensional unsteady flow fields.

2.1.3 Multi-Field Visualization

Multi-field data and *multi-valued* data both consist of multiple variables that are spatially coincident over the domain. The terms *multi-field* and *multi-valued* are distinguished by how the multiple variables are derived. In *multi-valued data*, multiple variables are derived from a single calculated or sampled variable. In the PIV experiments, for example, the velocity field is experimentally generated and the quantities of vorticity, Reynolds shear stress, and swirl strength are mathematically derived. *Multi-valued* data, however, is a more general term the describes a dataset of multiple coincident variables, including variables that are derived or obtained in a different manner.

Multi-field visualization refers to the techniques developed to analyze multiple variables that exist on the same domain. Classic examples are scalar and vector variables that co-exist, or tensor data visualization. In the following, many of the existing techniques that allow for the visualization of several different co-existing fields are reviewed.

Geometry-based Multi-Field Visualization – As early as 1991, Crawfis and Allison [6] presented a technique for mapping multiple scalar fields in addition to a vector field on a non-planar surface using the visual variables of color, texture, and height. An inherent limitation in visualizing vector data over surfaces in 3D is that it can be difficult for observers to disambiguate the effects of projection – possibly causing false interpretation of the orientation information. Also, the 3D nature of the new surface may occlude other regions of the domain.

Perceptual Textures – Healey and Enns [23] contributed a method that utilizes texture elements on an underlying 3D height field to visualize multi-field data. The texture dimensions of height, density, and regularity can be used to increase the number of attributes that can be simultaneously represented. Ware and Knight [85] proposed the use of Gabor functions to create texture-like images of flow data in which information is encoded along the perceptually significant texture dimensions of scale, orientation and contrast.

Layers of Images – Weigle et al. [89] propose a texture generation technique based on the layering of patches of oriented slivers. Luminance and orientation are used to encode information about multiple overlapping scalar fields. Inspired by brush strokes and layering concepts from painting, Kirby, Marmanis and Laidlaw [40] showed how different sized icons, color, elongated ellipses, and layering could be used to portray multivariate data from 2D compressible flows.

Tensor Visualization Using Ellipsoids – Laidlaw et al. [43] demonstrated how shape, orientation, and color attributes of ellipsoids could be used to represent multivariate components in diffusion tensor images of the mouse spinal cord. In addition, they

demonstrated a method for representing multi-valued data inspired by the brushing and layering techniques used in oil painting. Kindlmann [38] showed that intuitive tensor glyphs, based on superquadric surfaces, can be effective in demonstrating diffusion tensors data of the brain.

2.1.4 Texture Synthesis

Textures provide a subtle richness and wide spectrum of possibilities for visual appearances. The use of different, related textures applied in such a way as to demonstrate the underlying data would produce the ability to represent multiple variables in an intuitive and unique fashion. Several of the seminal algorithms and techniques related to the use of texture in visualization are reviewed below.

Texture synthesis methods [14, 88] allow for an unlimited quantity of texture patterns to be generated that are perceptually equivalent to a sample input texture. These algorithms work by constructing the output image in a pixel-by-pixel scan line order in which a pixel is synthesized by comparing a similarly shaped neighborhood around it with the original sample.

Along these lines, Efros and Freeman [13] introduced a “image quilting” algorithm that produces an effective texture synthesis pattern for a wide variety of input textures. The algorithm works by essentially piecing together small patches of the input texture which allows the output to maintain both continuity and coherence across the entire pattern. Similarly, Ashikhmin [1] presents a technique for synthesizing natural textures based on repeating patterns consisting of small objects of familiar but irregular sizes such as flowers and pebbles.

Khouas, Odet and Friboulet [36] use a 2D autoregressive synthesis method to simulate a 3D fur-like texture in order to represent two dimensional flow fields. This technique

allows control over streamlet orientation, length, and density, and has been used to produce striking visualizations of vector orientation and magnitude.

2.2 Fluid Dynamics

This section serves as a background to the research in fluid dynamics for which we developed the visualization techniques presented in this dissertation. It is designed to supply the reader with a sufficient background and terminology in order to understand the motivating factors which drive the need for multi-field visualization.

The understanding of the properties of fluid dynamics are essential in a range of scientific applications such as engineering, aerodynamics, and weather prediction. Turbulent flow is an active area of research in the field of fluid dynamics. Nobel-prize winning physicist Richard Feynman describes turbulence as "the most important unsolved problem of classical physics."

2.2.1 Turbulent Flow

Turbulent flow is chaotic in nature and characterized by swirling vortex structures, or *eddies*, of various sizes, strength, and orientations. One of the most basic types of turbulent flows is that which occurs when a fluid passes a boundary, such as air over an airplane wing or the flow of oil within a pipeline. The velocity of the flow must match the velocity of the wall at the surface of the boundary. A *boundary layer* or a *turbulent boundary layer* is the zone over which the average fluid velocity decreases from freestream value to zero. A flow is characterized by the ratio of inertial forces to viscous forces. This quantity is referred to as the *Reynolds number*. The Reynolds number of a flow, denoted *Re*, is mathematically defined as

$$Re = \frac{UL}{v}$$

Where U is the characteristic velocity, L is the characteristic length, and v is the kinematic viscosity of the fluid [54]. Turbulence occurs when the inertia term dominates the viscous term. A flow that is not turbulent is called a *laminar* flow.

We next introduce several derived quantities that are derived from the velocity vector field. These variables are extremely useful for the analysis of turbulent flow within a boundary layer.

Vorticity

In addition to the velocity vector field of the flow ($\vec{v}$), several other variables can be numerically calculated and are significant to theories related to turbulent flow. For example, the *vorticity* vector field ($\vec{w} = \nabla \times \vec{v}$) can be numerically calculated and analyzed with the velocity vector field. The vorticity vector field is a measure of the rotation of fluid flow. Analysis of the vorticity vector field is useful as it can provide an indication as to the strength and orientation of the swirling eddies within the flow. Understanding the coincidence of the velocity and vorticity fields is of interest to a number of investigators, particularly in conjunction with the visualization of vortex core locations. These visualizations allow for the analysis of several terms such as ($\vec{v} \times \vec{w}$), called the *Lamb* vector, and ($\vec{v} \cdot \vec{w}$) which is proportional to *helicity*, both of which have been suggested as possible descriptors of vortex tubes [50].

Reynolds Shear Stress

Given the components of the velocity vector field, the streamwise component u, the spanwise component v and the wall-normal component w, the scalar field *Reynolds Shear Stress* can be calculated as $-uw$. Areas in where there exists significant values of Reynolds Shear Stress are interesting to researchers in fluid dynamics as the generation of Reynolds shear stress is believed to be correlated with increased surface drag and the sustenance of turbulence [16]. In turbulent boundary layers, various theories have indicated that hairpin shaped vortices cause drag by producing Reynolds shear stress, and that this process may be enhanced when multiple hairpins travel together with similar speeds as a packet.

Swirl Strength

Once the velocity vector field has been numerical simulated or experimentally-generated, velocity gradients can be calculated and are typically used to understand and identify the subtle components of fluid dynamics. Jeong and Hussain [30] indicated that the second invariant of the velocity gradient tensor, λ_2, is a quantity that measures the dominance of vorticity over strain and proposed that it could be used as a criterion to identify vortices. Zhou et al. [93] suggested the use of the imaginary part of the complex eigenvalue (λ_{ci}) of the velocity gradient tensor to visualize vortices. This value is referred as the local *swirl strength* of the vortex. For the calculations presented in this dissertation, the Zhou et al. definition is used.

A simplified version of swirl strength using a 2D velocity gradient can be computed to identify vortex cores. This quantity is referred to as the 2D swirl scalar field and is limited to identifying vortex cores that are oriented in a position normal to the sampling plane. Dual plane PIV is an extension of single plane PIV in which velocity components are measured in two planes simultaneously. This allows for all 3D velocity gradient measurements to be calculated. The corresponding 3D swirl scalar field is a measure of the existence of a

vortex core in any orientation.

The characteristics of vortex cores, including their orientation given by 2D and 3D swirl, are useful in theories designed to reduce skin-friction drag in turbulent flow.

2.2.2 Particle Image Velocimetry

Particle image velocimetry (PIV) is a technique that can be used to experimentally measure instantaneous components of a velocity field in a plane of a turbulent boundary layer in a moderate to high Reynolds number flow. PIV uses optical recording to measure a full field flow in a manner that is instantaneous and non-intrusive. The experimental set-up of a PIV system includes tracer particles that are added to the flow. A laser illuminates these particles in a plane of the flow at least twice within a short time interval. The light scattered by the particles is recorded and the displacement of the particle images is determined through evaluation of the PIV recordings.

Standard PIV is capable of recording the projection of the velocity vector within the plane. The out-of-plane velocity component can be calculated by making an additional PIV recording from a different viewing axis and reconstructing the 3D geometry using both camera angles. This technique is referred to as *stereoscopic* PIV. The experimental facility for the PIV experiment is shown in figure 2.2.

The particles used in PIV experiments must be small as to non-intrusively follow the flow, scatter the laser light, and illuminate the cameras positioned to capture images. Tiny particles of olive oil (mean diameter of 1 μm) are the most commonly used for the tracer particles as they can be atomized, used in air flows, and are non-toxic.

While the PIV data is sampled on a 2D plane, the resulting footprint of the multi-variate data gives insight into the structure and behavior of the 3D flow. The advantage of examining the PIV experimental data, compared to computationally-generated data, is the

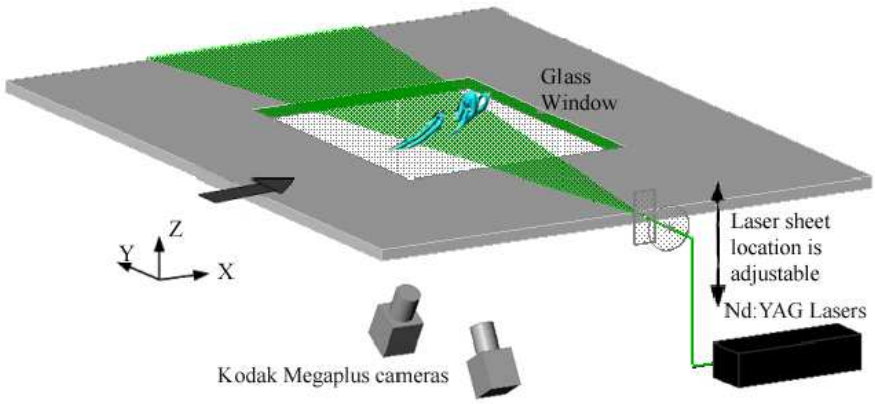

Figure 2.2: Experimental facility for a PIV experiment.

ability to analyze the behavior from experimental results of moderate to high Reynolds number flows.

2.2.3 Numerical Simulation

Apart from experimentally-generated data, we consider wall-bounded turbulent flow data from a direct numerical simulation (DNS) of the full Navier-Stokes and continuity equations [11]. The great value of DNS data is that it is fully resolved and provides full four-dimensional (space and time) data. However, the difficulties with DNS is that it is very expensive and limited in Reynolds number. This is because cost of simulating wall-bounded turbulent flows scales nominally with the Reynolds number Re_τ^3. A simulation involving 2.7 billion grid points ($3072\times2304\times385$), results in a time-step yielding a raw (unprocessed) field of nominally 9 GB in size. Recently, Hoyas and J. Jiménez [26] have reported a new simulation for nominally twice the present Reynolds number ($Re_\tau = 2003$). In this case, the simulation was reported to run 6 million processor-hours on a 2048 processor supercomputer, generating approximately 25 TB of raw data.

The modeling of a turbulent channel flow is of interest to researchers in fluid dynamics because the simulation allows for the analysis of large, energetic features in the flow and the

ability to correlate the flow structures in different layers within the 3D model. Developing a deeper understanding between the different regions within a channel flow results in a better understanding of the formation of vortical structures and the key-physical features that cause skin-friction drag.

The numerical technique requires the integration of the Navier-Stokes equations in the form of evolution problems for the wall normal vorticity and the Laplacian of the wall-normal velocity [11]. For spatial discretization, Chebychev polynomials were used in the wall normal direction, while de-aliased Fourier expansions were utilized in wall-parallel planes. The temporal discretization used was a third-order semi-implicit Runge-Kutta scheme. As a large streamwise extent is needed to capture the largest energy-containing motions of the flow, the simulation employed a computational domain of $8\pi\delta$ units in the streamwise direction and $3\pi\delta$ units in the spanwise direction, where δ is the channel half width. The large streamwise extent is needed to capture the largest energy-containing motions in the flow. The Reynolds number for the simulation was $Re_\tau = 934$.

One aspect of the DNS data is that it allows for the investigation of very large scale energetic features and how these may interact between different regions in the flow.

2.2.4 Regions of Interest within Flow Data

Conventionally, wall turbulence is described in terms of three predominant regions: an inner region, immediately next to the wall, that is dominated by viscous processes; an outer region, far from the wall, where viscous effects are negligible; and an intermediate region, which often is referred to as the overlap of the inner and outer regions. This middle region is also referred to as the log region or the inertial wall region. Most existing theoretical approaches regard the inner region to universally scale with inner-wall viscous variables and to be independent of the log region and beyond. Recent studies have shown evidence that the inner region is influenced by outer region parameters (δ the channel half-width,

and U_1 the channel center-line velocity) [10, 17].

In streamwise-spanwise planes parallel to the boundary layer surface, the packets can be characterized by zones of uniform but low streamwise velocity containing areas of high negative Reynolds shear stress and falling between cores of strong positive and negative vorticity [18].

PIV experimental studies have consistently shown the log region to consist of packets of hairpin vortices, with long streamwise regions of momentum deficit and with high-speed fluid seeming to fill the separation between neighboring motions. However, the PIV fields are somewhat limited in length (approximately 2δ). It has recently been discovered that the structures can exist up to 20δ in length and may have a tendency to meander in the spanwise direction [27]. The tendency of the low-speed region to meander across the flow explains why such long length scales have not been observed previously from single-point measurements in boundary layer flows. Kim and Adrian [37] have noted such long structures for turbulent pipe flows from single-point hot-film measurements, and referred to them as “very large-scale motions” (VLSM).

The discovery of the very long “superstructures” is particularly significant as it indicates an outer-layer scaled phenomena that may have influence all the way down to wall in the inner layer. The DNS data is very useful for investigating this conjecture and some results are shown in section 6.4.3 in which two planes are shown simultaneously from the DNS dataset. The results indicate that near-wall regeneration mechanisms are *not independent* of the slow dynamics associated with structures on the order of the external dimension of the flow, as has always been previously believed.

2.3 Astronomy

An active area of research in the field of computational physics is the modeling of magnetohydrodynamic light supersonic jets in the context of astrophysical galaxy clusters [51]. These high-speed jets propagate distances of over 650,000 light years from their sources, transporting energy and magnetic fields to their surrounding environments. The jet magnetic field is advected along with the flow and is expected to reflect properties of the evolving velocity field. Of particular interest is the extent to which the magnetic field and velocity vector fields are spatially aligned and/or orthogonal to one another and the interplay between magnetic field strength and the corresponding velocity structures. The primary goal of this research is to explore the connection between the large-scale flow dynamics and the small-scale physics underlying the observed emissions from real radio galaxies [70].

2.3.1 Astrophysical Jets

Several observable properties result from the interaction between a high-velocity plasma jet launched within an active galactic nucleus and an ambient environment. In addition to observable lobes of luminous material, characteristic radio emission signals the presence of magnetic fields and relativistic electrons. It is not well understood whether these materials are transported along the jet, introduced in the jet-intergalactic medium encounter, or both [71]. In addition to the velocity field, physicists are concerned with the magnetic vector field advected by these supersonic jets and are interested in analyzing the relationship between both vector fields. Developing a deeper understanding of the relationship between the vector fields' topology, magnetic field strength, and corresponding velocity structures results in a greater understanding about how kinetic and magnetic energy distributions evolve in these systems and contributes to the explanation of radio emissions that can be

physically observed.

2.3.2 Magnetohydrodynamics

Turbulence within electrically conducting fluids is necessarily accompanied by magnetic-field fluctuations [3]. Magnetohydrodynamics is the branch of science dealing with the dynamics of matter moving in an electromagnetic field. Given a magnetic field $\overrightarrow{B}$, and a vector field $\overrightarrow{v}$, changes in magnetic field can be calculated from the ideal magnetic induction equation:

$$\frac{\partial \overrightarrow{B}}{\partial t} = \nabla \times (\overrightarrow{v} \times \overrightarrow{B})$$

This equation describes how motions of a perfectly conducting fluid change the magnetic fields contained therein. By examining the magnitude of this vector, we examine correlations between increases in magnetic field strength and anticipated shear or compression regions in the velocity field. Simultaneous visualization of all components of the velocity field and the rate of change of the magnetic field strength enables us to locate regions of active field enhancement, distinguish newly-magnified fields from those advected along with the plasma, and identify velocity structures that generate enhanced fields.

2.4 Discussion

Because the process of knowledge discovery related to these applications is predicated on the ability to achieve an integrated understanding of the individual contribution of each variable and of how the variables interrelate with each other, developing effective multi-

variate visualization methods is of critical importance to facilitating the understanding and analysis of results from the turbulent flow (PIV) experiments and magnetohydrodynamic jet simulations.

Chapter 3

Illustrating Components of Flow

In this chapter we address several visualization techniques designed to graphically depict components of flow. The techniques highlighted here are mainly the result of an ongoing effort to construct tools to develop a better understanding of the complicated interactions of fluid dynamics.

We begin by discussing the classification of existing flow visualization techniques including the differentiation between texture and direct flow visualization. We continue with an illustration of visualizing different convection velocities. We expand upon a current technique designed to effectively use luminance ramps over dense streamlines to represent the direction of flow. Finally, we define the subtle differences between streamlines, streaklines, and pathlines.

3.1 Direct and Texture-based Flow Visualization

Images similar to figure 3.1 (left) are often used to display flow information through the use of glyphs, such as arrows, to indicate the direction of flow at a discrete set of points. The length of the arrow may be used to represent the velocity magnitude. While effective

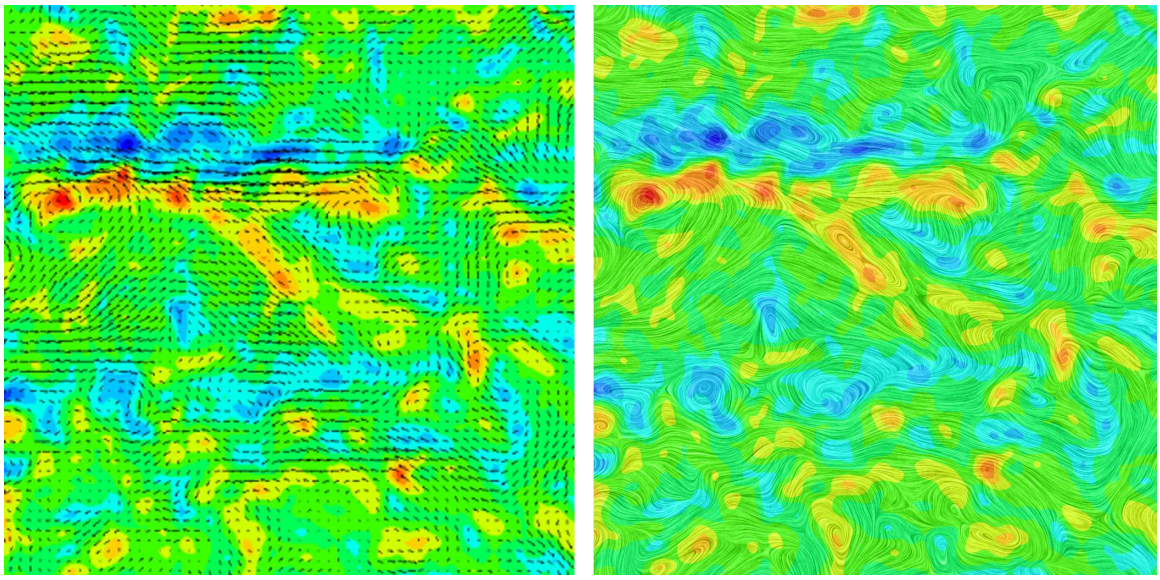

Figure 3.1: Examples of glyph-based visualization (left) and texture-based visualization (right).

and efficient, one limitation is that such simplistic plots can only provide information at relatively sparsely sampled points over a domain, as each glyph will require several pixels to be drawn. Even when the vector field is not down-sampled, the collection of glyphs may not easily lend itself to the perception of global fluid flow as the segments must be perceptually interpolated and connected in order to understand the path of a particle. Selecting samples along a uniform grid may lead to artifacts in which the structure of the grid interferes with correct perception of the direction indicated by the vectors [44].

Texture-based visualization methods produce high-resolution output images that allow structures of the flow to be perceived more easily than with the field-of-arrows technique (figure 3.1 right). Dense textures allow for the direction of flow at all locations in the domain to be salient, whereas the field of arrows only give discrete samples that can be difficult to interpret. Advantages of texture-based visualizations over field-of-arrow techniques include higher resolution and a continuous representation of flow when compared to a field of arrows.

An additional scalar variable may be represented through the use of a color encoded

background underlying the array of vector glyphs or as a color overlay in the texture-based approach. Both images in figure 3.1 use a color overlay to represent vorticity magnitude in an experimentally acquired PIV dataset. Techniques designed to visualize scalar values will be covered in detail in chapter 4.

3.2 Illustrating Different Convection Velocities

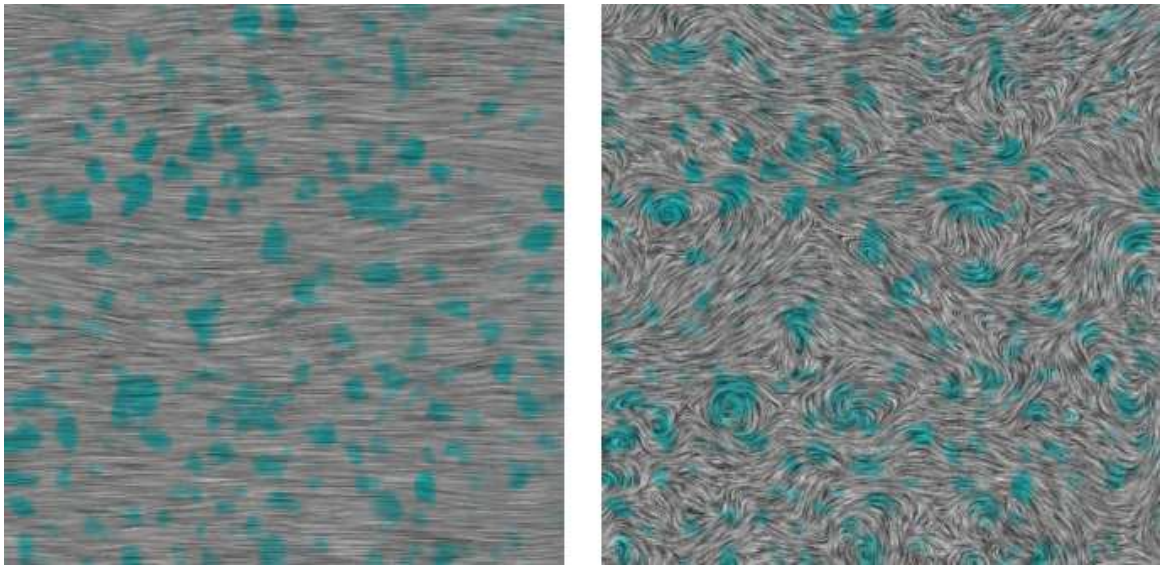

Figure 3.2: In-plane velocity field with swirl strength superimposed in color. The left image is the raw flow data in which the visualization is dominated by the magnitude of the vectors in the streamwise direction. The right image depicts the vector field that results when the average streamwise component is subtracted from each vector.

The visualization of any vector field is dependent on the relative velocity of the observer. For a stationary observer (figure 3.2 left), the streamwise component of the vector field (U) greatly dominates the visualization creating an image that does not lend itself for a further understanding of the dynamics of the flow. Typically, the average value of the streamwise component ($\overline{U}$) is calculated and subtracted from each vector prior to visualization (figure 3.2 right). Subtracting a value of the streamwise component from each vector changes the relative velocity of the observer. The resulting image, critical points, and vector field

features are greatly influenced by the magnitude of the value subtracted.

While subtracting a value of the streamwise component from each vector allows for the visualization to be effective, the resulting flow is specific only to that relative velocity. Different characteristics of the flow may be evident by subtracting a value other than the average streamwise velocity. To illustrate this phenomenon, the scalar quantity swirl strength (λ_{ci}) is superimposed in blue over the images in figures 3.2 and 3.3. Swirl strength λ_{ci} is *Galilean invariant* and does not vary with the magnitude subtracted from the streamwise velocity. Swirling eddies become apparent at regions of high swirl strength only when the velocity of the eddies matches the convection velocity subtracted from each vector.

LIC animations can be used in combination with the swirl strength to determine the convection velocities of various eddies in the flow. Specifically, the swirl strength can be overlaid on LIC animations computed using a range of values $U - U_c$ to determine under what conditions locations with large λ_{ci} coincide with swirling streamlines. The animation contains a series LIC images in which U_c is varied from $0.5\overline{U}$ to $1.5\overline{U}$. Figure 3.3 shows several frames from an animation where the number in the bottom left hand corner denotes the percentage of the average streamwise vector component that was subtracted from each vector. By stepping through the various images in the animation, it becomes apparent that swirling streamlines appear and disappear at different values of the convection velocity. In addition, many other specific features are emphasized at distinct convection velocities.

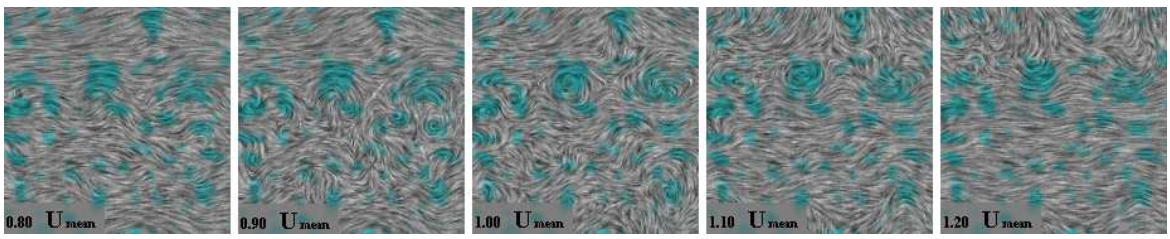

Figure 3.3: In the progression of the images below, the swirling eddies, highlighted by swirl strength, are evident in some frames and not in others only because the frame is convecting at a speed that matches the convection velocity of those eddies.

For example, many of the swirling zones appear to convect at the streamwise local mean velocity ($\overline{U}$).

This process of visualizing different convection velocities can aid scientists in determining under what conditions locations with large swirl strength coincide with different flow dynamics.

3.3 Depicting Flow Direction

Several techniques have been proposed to overcome the problem of ambiguous flow direction that occurs in static LIC images [86, 87]. These techniques typically involve either animation or the use of a monotonically increasing luminance ramp to disambiguate the direction of the flow. Sanna et al. [60] developed a space-filling method called Thick Oriented Stream Lines (TOSL), in which the orientation of a flow is depicted by increasing luminance values along calculated streamlines. An advantage of this technique is that it provides a dense representation of the vector field.

The TOSL technique is particularly advantageous because of its high density output, ability to accurately depict flow direction, relative simplicity, and potential for efficient implementation. Using the original TOSL algorithm as inspiration, we extend the technique to enhance the visual effect and improve perception of the flow field.

The first step in the TOSL method, as in the LIC method, is to numerically calculate streamlines according to the given flow field. The two approaches differ, however, in that the TOSL method does not use an input texture and does not initiate a convolution process. Instead, intensities for pixels along streamlines are incremented according to the local vector magnitude. The initial 8-bit pixel value is randomly set within a range of 30 and 120 and the algorithm continues by stepping along each pixel calculated in the streamline and assigning an increasing intensity value. Local vector magnitude is taken into account,

as the value of each pixel is incremented by an amount that reflects the velocity magnitude at that point. Each vector magnitude is normalized with respect to the maximum velocity on the local streamline. If the vector has a high relative velocity along the streamline, the increment in grey tone is proportionally high at that point in the image. A high density output texture is obtained by creating the image in two passes. The first pass creates a sparse texture by coloring only a percentage of the pixels, using a specific procedure to select randomly spaced seed points. After a user-defined percentage of the image has been filled, the remaining pixels are considered in scan-line order to ensure that the entire image is completed.

Next we describe how we have expanded this technique to more effectively use luminance ramps over dense streamlines to represent direction of flow.

Modifications of the TOSL Algorithm

We have found that starting with an initial intensity value between 30 and 120 can lead to artifacts due to streamlines that have a similar range and start from a similar point (such as the edge of the domain or a singularity). Figure 3.4 shows that this can result as a darkened

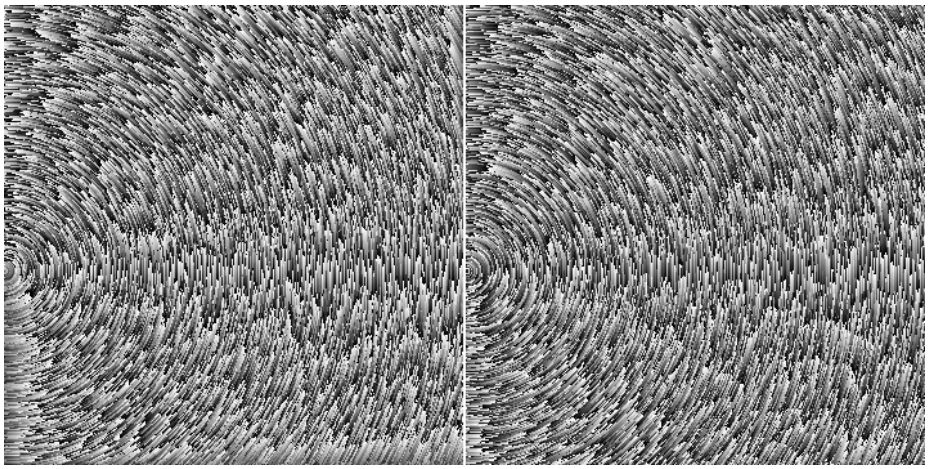

Figure 3.4: Restricting the initial value of the streamline can lead to artifacts at boundary of the image domain, as seen in the lower left and bottom of the leftmost image. This problem can be alleviated by allowing the initial pixel values to span the entire range from 0 to 255.

and uniform artifact along the edge of the domain where the streamlines are introduced. The authors of TOSL suggest this starting range in order to avoid initial dark grey and also to avoid *jumps* in which the values of neighboring pixels would be 255 and 0. It has been our experience that computing long streamlines in which several cycles of pixel values ranging from 0 to 255 occur, including *jumps*, can be advantageous. This allows one streamline to carry several repeated luminance ramps indicating the direction of flow and results in a fluid final image. The problem of edge artifacts can be alleviated by allowing the starting value to be randomly assigned to any value between 0 and 255.

Secondly, we feel that a more accurate representation of the entire vector field could be obtained by using the maximum global vector magnitude to normalize the step size. Incrementing the intensity of pixels with a step size that is directly proportional to the local vector velocity magnitude produces appealing results at the expense of global inconsistencies. With this approach, one cannot compare line lengths in different areas of the image to determine if the velocity magnitude is at a global maximum or simply a local maximum. By adjusting the factor in which the vector magnitudes are normalized, it is possible to provide a more globally consistent portrayal of the scientific phenomenon.

Finally, we find it appropriate to make the step size inversely proportional to the vector velocity magnitude instead of directly related to the velocity magnitude. While an observer of an image may learn to read short lines as representing high speed areas, and long lines as representing slow moving flow, we find this approach counterintuitive. Reversing the mapping results in creating long smooth lines where the flow velocity is at the global maximum. This is evocative of the result that a spot smeared out over a period of time will produce a long streak where the flow is faster. Figure 3.5 illustrates the effects of making these changes. In the topmost image, velocity magnitude is normalized with respect to the values along the local streamline only. This results in short streamlines at places where the velocity is greatest along each individual streamline. In the lower image, velocity

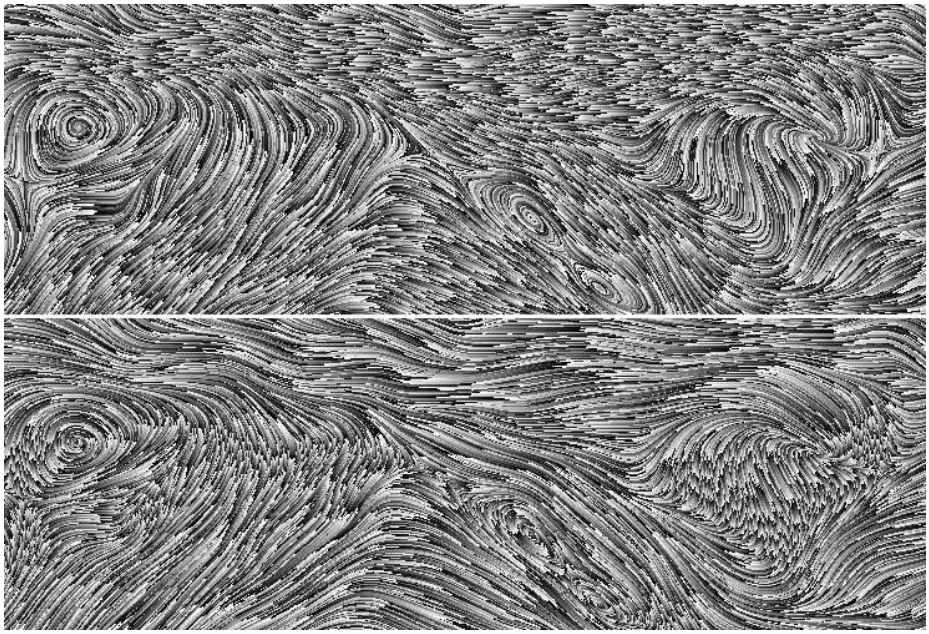

Figure 3.5: Velocity magnitude is normalized with respect to the values along the local streamline only (top). Velocity magnitude is normalized with respect to the global velocity magnitude (bottom).

magnitude is normalized with respect to the global velocity magnitude. The luminance ramp along streamlines is also defined using an inverse relationship between the step size and the vector magnitude, resulting in long streamlines where the velocity is at the largest magnitude globally over the domain

3.4 Streamlines, Streaklines, and Pathlines

Streamlines, streaklines, and pathlines are tools commonly used in the analysis of flows in order to obtain a fundamental understanding of a vector field. They are often confused as the differences between them are subtle. The purpose of this section is to define and differentiate streamlines, streaklines, and pathlines as they are used in the fields of flow visualization and fluid dynamics [5].

A *streamline* is a line that is tangent to the velocity vector at every point [66]. This line

gives the direction and orientation of the flow at each point along the line. In the simplest case, we define a stationary 2D vector field as map $v : \mathbb{R}^2 \to \mathbb{R}^2, x \mapsto v(x)$. A streamline is defined by its *integral curve* or path, $\sigma(u)$, whose tangent vectors coincide with the vector field:

$$\frac{d}{du}\sigma(u) = v(\sigma(u))$$

Notice that time is not included in the above formula. Streamlines are drawn for a specific, static time in the history of the flow. In general, streamlines constitute the outline of the fluid layers of the vector field at a particular instant in time [15].

If we wish to consider how a flow field changes, we need to modify the model to include the dimension of time. A *pathline* is the trajectory of a single fluid particle takes over time. In this system, the vector field must be defined with respect to a given location x and the time t. Correspondingly, the equation that defines pathlines is:

$$\frac{dx}{dt} = V(x,t)$$

A *streakline* is the collection of points that have previously passed through the same point in the flow field. In other words, a streakline is the current location of all fluid particles that have passed through a particular spatial point in the past.

To mathematically define a streakline [5], we consider a function that supplies a location within the domain if it has previously passed through a point, say $\mathbf{y}$. Define a point $\mathbf{a}$ that has pass through point $\mathbf{y}$ at time τ by stating $\mathbf{a} = \mathbf{a}(\mathbf{y}, \tau)$. At time t, the particle has advanced to $\mathbf{x}$ given by $\mathbf{x} = \mathbf{x}(\mathbf{a}, t)$.

Therefore, the streakline z at time t for all points that have passed through point y would be defined as:

$$z = x[a(y,\tau),t] \qquad 0 \leq \tau \leq t$$

In static flow, where the vector field does not change with respect to time, streamlines, pathlines, and streaklines reform to the same collection of points. However, in unsteady motion, the vector field is defined with respect to time and streamlines, pathlines, and streaklines trace out distinctly different paths. A streamline is line that follows the flow given a static snapshot of the vector field. A pathline traces out the motion of a particle released in the flow. A streakline is the collection of points that have all passed through the same point.

3.5 Discussion

One of the challenging aspects of research in the field of visualization is understanding the principles of the scientific phenomenon that are under investigation. This section introduced several important components of fluid dynamics and flow visualization that were uncovered during our research into developing images that accurately represent vector field data. We discussed texture-based visualization versus direct visualization, depicting flow direction with luminance ramps, illustrating convection velocities, and defined the difference between streamlines, streaklines, and pathlines. While this chapter does not provide a comprehensive list of underlying components associated with flow visualization and fluid dynamics, we feel it does provide a informative introduction to the novel contributions presented in the rest of this dissertation.

Chapter 4

Techniques for Visualizing Scalar Values

A scalar is defined to be a single data value associated with each point of a dataset, quantified with a numerical value [63]. The collection of values representing the same quantity over the entire domain is referred to as a scalar *field*. Visualizing scalar fields is often useful in obtaining an understanding a particular behavior of an attribute over a domain. For example, a scalar field such as temperature can quickly indicate which regions are warmer or cooler than others, and at which point the most extreme values exists. While scalar field visualization is relatively straight-forward, the visualization of scalar fields can become complicated when there exists multiple coincident fields.

The images produced in this chapter are an effort to represent multiple scalar fields obtained from stereoscopic *particle image velocimetry* (PIV). As introduced in section 2.2.2, PIV is a technique that can be used to experimentally measure instantaneous component of a velocity field in a plane of turbulent boundary layer. Along with the vector field, scalar fields of vorticity, Reynolds shear stress, and swirl strength can be mathematically derived and are important in characterizing potential "regions of interest." The process of knowledge discovery related to dual-plane PIV experiments is predicated on the ability to achieve an integrated understanding of the individual contributions of each scalar variable

and how the variables inter-relate with each other. Developing effective multi-variate visualization methods is of critical importance to facilitating the understanding and analysis of results from these experiments.

In an effort to better understand how multiple variables interact, we have developed several techniques to assist in the visualization of single and multiple scalar fields.

We begin by discussing the use of color to represent scalar fields. We illustrate how color has traditionally been used for scalar representation and motivate a new technique designed to effectively represent multiple scalar fields within an image depicting turbulent flow. We then discuss how 3D graphics can be used to represent a scalar field by incorporating height in addition to a standard texture map. We continue by discussing the role of luminance and contrast in grayscale images, and the perceptual limitations that prevent individual techniques to be effectively combined to represent multiple scalar fields simultaneously. Additionally, we describe how streamline density can be effectively used for representing a scalar variable. Finally, we present a technique, based on embossing, to encode the out-of-plane component of a 3D vector field over a 2D domain.

4.1 Color

Past and present research in the fields of color theory and information visualization address the issue of effectively choosing colors to represent data [2, 24, 55, 56]. Defining an appropriate mapping that accurately correlates the quantifiable aspects of data into the human perceptual system can be a challenging problem. We discuss the challenges and techniques related to using color to represent scalar fields within flow data.

4.1.1 2D Color Plots

One of the most straight-forward approaches to representing scalar fields is to use a colormap to represent the data. Once a sequence of colors and corresponding scalar values for each color have been defined, pixel values for the image are colored according to the value of the scalar at each point on the domain. Figure 4.1 illustrates this method on three images that depict different variables obtained from a PIV dataset. While effective in representing each component separately, it is difficult to obtain an *integrated* understanding of the coincident scalar fields from the separate images.

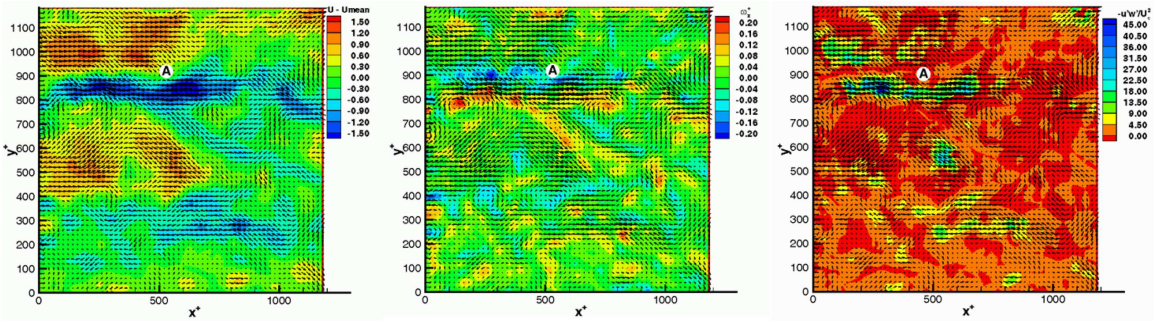

Figure 4.1: 2D color plot images depicting a single scalar value of a PIV dataset: (from left to right) streamwise velocity, vorticity, and Reynolds shear stress.

Figure 4.2 shows an attempt at using multiple colors to represent several variables within a single image designed to visually highlight hairpin vortex packets from an experimentally acquired PIV dataset. In this image, red and blue show regions of positive and negative vorticity, yellow depicts zones of strong Reynolds stress, green shows regions of relatively uniform streamwise velocity. In general, saturated hues indicate larger magnitudes. Two other features are depicted in figure 4.2: black curves mark the spanwise center of each hairpin packet identified and pink regions mark search zones upstream and downstream of each packet. The composite plot is fairly effective at highlighting the regions of interest as defined by the combination of the variables. The type of analysis that this image allows would be nearly impossible from separate color plots as in figure 4.1. However, the composite plot does have shortcomings. It is difficult to view the extent of overlap of

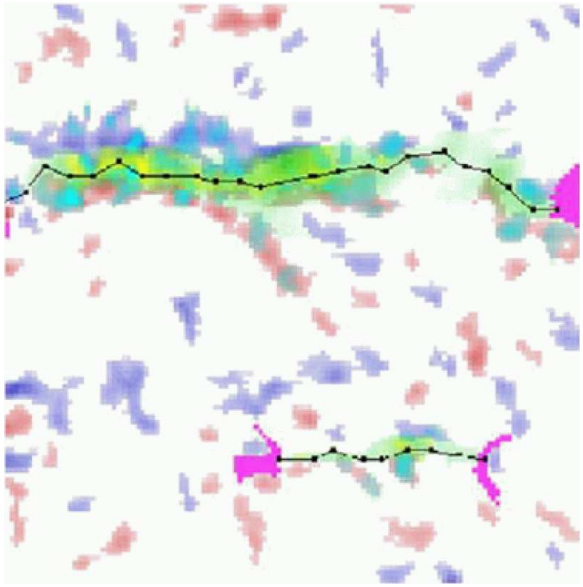

Figure 4.2: Using overlaying colors to represent four different scalar fields in a single image.

the different variables as the mixture of the colors can result in an ambiguous color. This makes it difficult to interpret finer points of the multi-valued data.

Next we present how color can be used, in conjunction with LIC images, to effectively represent multi-valued flow data.

4.1.2 Motivation for Effective Color Use

With few exceptions, the use of color with LIC has traditionally been limited to the simplest of color compositing operations in which a LIC texture image is in effect overlaid with a single continuous semitransparent color wash image, with the resulting effect that blacks are left black and the whites are shifted toward the specified hue at each point. While effective for conveying a single scalar distribution in the context of the flow, this post-process method does not allow for the effective simultaneous representation of multiple scalar fields, due to the perceptual difficulty of and inherent ambiguity in color decomposition.

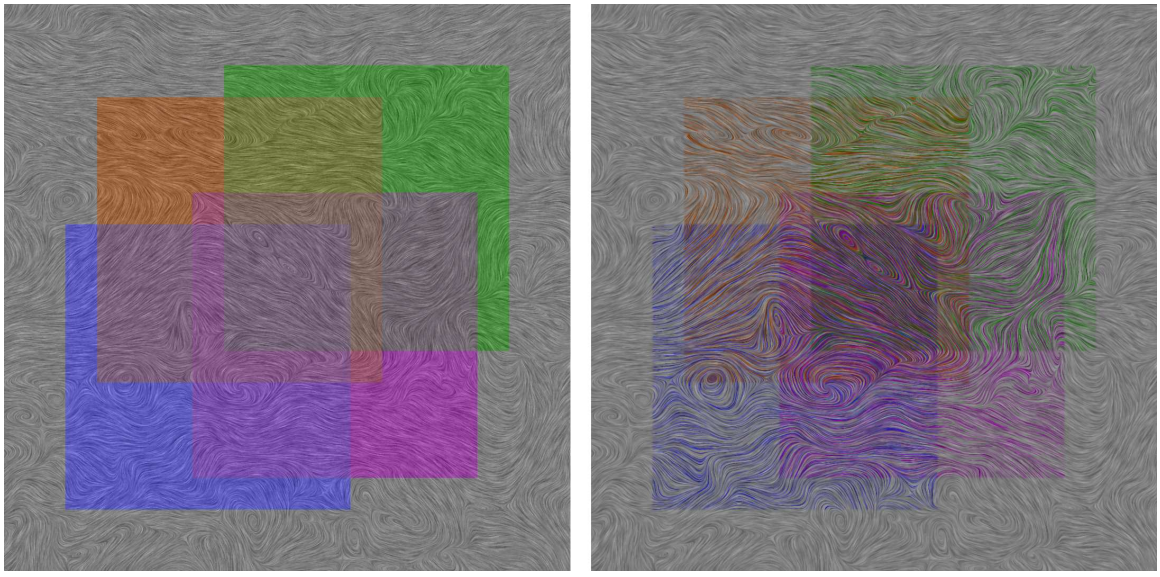

Figure 4.3: Four artificially defined, mutually overlapping regions, overlaid on a LIC image (left). The same four regions, represented across the same LIC image via color weaving (right).

Figure 4.3 (left) displays the inherent problem with overlaying colors to represent multiple fields. The color combinations in the overlap regions are obtained by averaging in RGB colorspace. Averaging multiple colors in this manner results in an ambiguous grey color that is not representative of an area in which multiple colors or significant scalar values are present.

As an alternative, we propose a technique in which multiple colors are allowed to coexist on neighboring streamlines, resulting in multicolored images that resemble a tapestry woven with different colored threads. Figure 4.3 (right) displays four regions, represented across a LIC image via *color weaving*. Note the continuity of color along individual streamlines within each region, and the ability to accurately perceive combinations of component colors

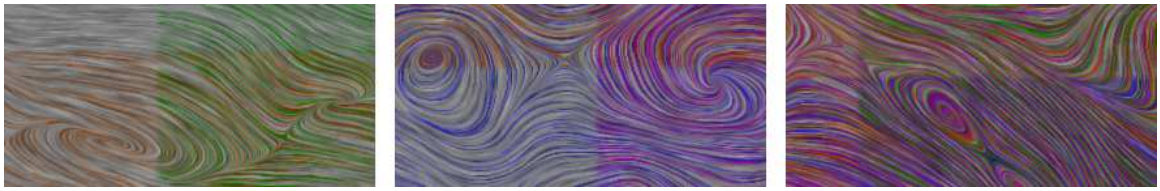

Figure 4.4: Three close-up excerpts from the overlap image shown in figure 4.3 (right).

in the areas of high overlap (characterized by the presence of three or more layers).

We describe the details of the algorithm in the next section.

4.1.3 Color Weaving Algorithm

We begin by selecting several highly saturated and perceptually iso-luminant colors using the technique suggested by Kindlmann [39]. As luminance plays a primary role in how features are perceived [84], selecting base colors that are as perceptually uniform as possible helps to achieve a final image in which similar concentrations are represented with reasonably equivalent prominence across the multiple distributions. Additionally, selecting colors that are relatively equivalently discriminable reduces the potential for ambiguous or misleading representations [24]. We then use each base color to define a corresponding two-dimensional colormap, in which saturation increases along the horizontal axis and value increases in the vertical direction (figure 4.5).

The colormaps are constructed by determining the extreme values for each of the four corners of the colormap. First, a default LIC algorithm is run on a white noise image to produce a grey-scale texture that represents a vector field. This pre-process step is initially required in order to determine typical values for the darkest and lightest shades of grey, and is not necessary to repeat once the colormaps have been determined. The value of the upper left corner of the 2D color map is the darkest grey color obtained by the LIC algorithm; the value of the lower left corner of the color map is the lightest shade of grey in the default grey-scale LIC image. We match the luminance values of highly saturated and moderately saturated values for several distinct hues using the technique introduced in [39]. The value in the upper right corner of the 2D color map is the darker, subtle color of the hue; the value in the lower right corner of the colormap is the highly saturated color. The values for the interior of the 2D colormap are calculated by interpolating the saturation of the colors in the horizontal direction and interpolating the value of the colors in the vertical direction.

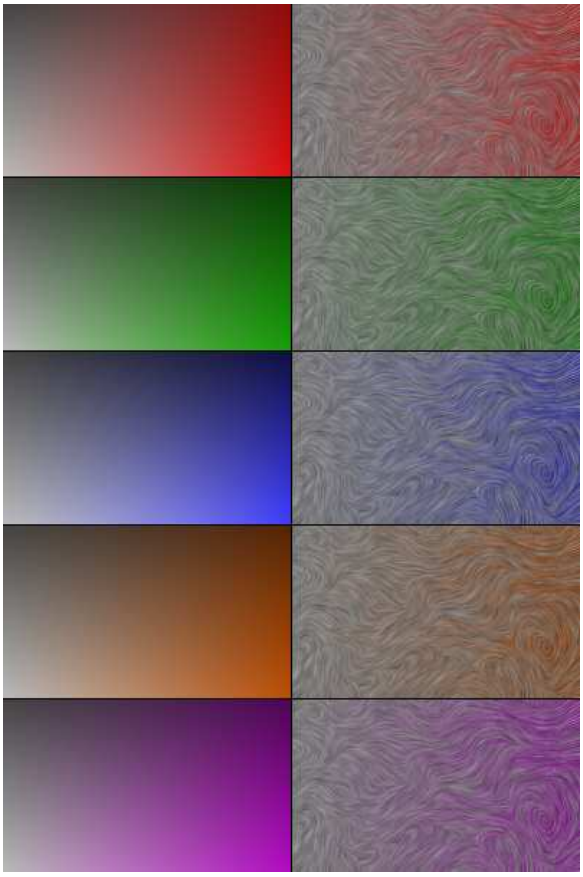

Figure 4.5: A suite of two-dimensional colormaps; the results of using each colormap to represent, over the same LIC texture, a simple scalar distribution that is increasing in value from left to right.

Each scalar distribution is associated with a unique hue and colormap.

We introduce color on a streamline-by-streamline basis during the computation of the LIC image. At each pixel along a streamline we define the final image color using a 3D color table lookup. The first index defines the hue, or the choice of which 2D colormap to be used. Within each 2D colormap, the index for the value component is defined by the grey value obtained from running the LIC computation, and the index for the saturation component is defined by the magnitude of the value of the particular scalar distribution being represented at that point. The desired result is to preserve the luminance pattern established by the LIC while defining the colors to be fully saturated at points where the scalar variables reaches their maxima, fading to the default LIC value as the magnitudes of the scalar variables

decrease.

The streamline-based fast-LIC [66] algorithm is critical to our implementation as the hue index is only incremented when a new streamline is calculated. In our latest implementation, the output image is created in a single pass. Special steps have to be taken to prevent color mixing, while preserving the anti-aliasing effects that are achieved when multiple streamlines are allowed to pass through each pixel. We accomplish this by recording, for each pixel in the output image, the hue index of the first streamline that was used to determine its color. When subsequent streamlines pass through the same pixel, the original hue index is used in the color table lookup, so that only the luminance components from the multiple streamlines are blended when the results are averaged.

In areas characterized by the presence of prominent values in multiple distributions, alternate colors are visible along adjacent streamlines. We use a sparse and a consistent mapping of individual colors to individual streamlines in order to maintain continuity in the representation of each distribution and to ensure that the apparent concentration of each color remains in consistent proportion to the magnitude of the corresponding scalar distribution, regardless of the concentrations of the other scalar distributions at that point.

The success of our method depends on being able to represent multiple different streamlines across each point in the multi-variate distribution being portrayed. To achieve best results, it is generally necessary to up-sample the input data. For the images presented in this chapter, a 1071x1071 input texture was used, and the maximum streamline length was 380.

Figure 4.6 shows the color weaving algorithm applied to an experimentally acquired PIV dataset in which we simultaneously highlight areas of significant positive vorticity (red), negative vorticity (blue), strongly negative Reynolds shear stress (green), and high swirl strength (orange or magenta, depending on the direction of the swirl). Careful analysis of this image allows for an accurate interpretation of the flow structure in the overlapping regions, e.g. correlation of out-of-plane vorticity and velocity or identification of vortex

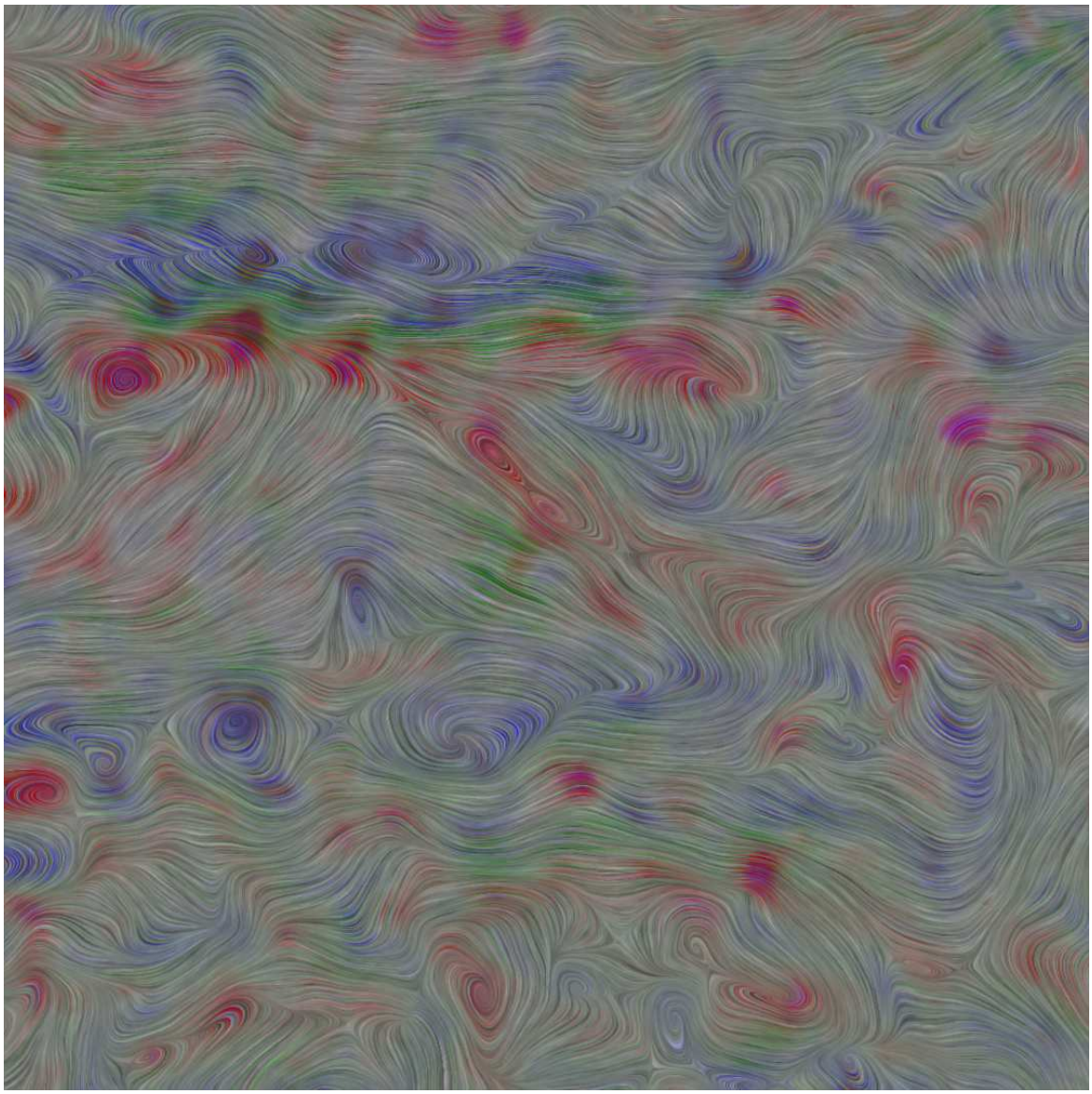

Figure 4.6: A composite *color woven* image of an experimentally acquired PIV dataset.

cores within shearing zones. Figure 4.6 specifically demonstrates evidence of spatially organized packets of hairpin structures in the streamwise-spanwise planes of the PIV dataset.

4.2 3D Graphics

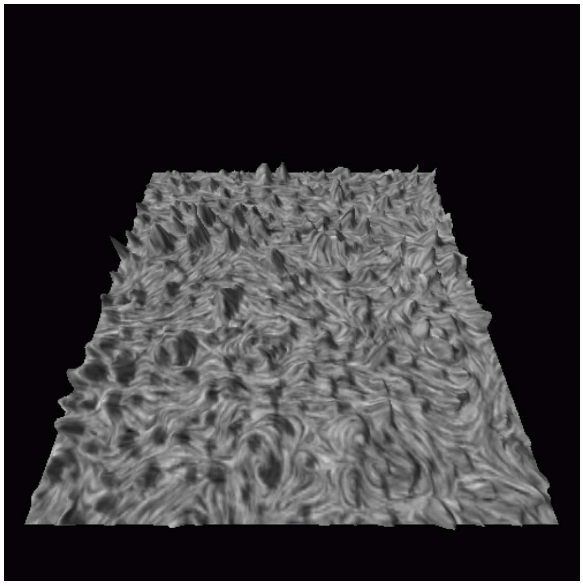

Figure 4.7: 3D graphics in conjunction with a texture map is used to represent a scalar field coincident with a 2D flow.

In addition to color, a third dimension can be employed in order to incorporate additional parameters into a single visualization. A 3D graphical approach can be useful to discriminate spatially discrete zones of a scalar field that may overlap with another. The combination of 3D graphics and texture is illustrated in figure 4.7 to represent components of an experimental PIV dataset.

Swirl strength is a scalar field limited to positive values. Because high swirl strength typically appears in discrete circular areas that lie within zones of significant vorticity, it is a reasonable candidate to be represented with height in a 3D plot. Figure 4.8 shows parameters with color and brightness, along with height, to represent additional

features: shades of green correspond to regions of high Reynolds shear stress, blue and red correspond to negative and positive zones of wall-normal vorticity, brightness is used to emphasize the coherent regions of low streamwise velocity identified by the feature extraction algorithm, and height represents swirl. This image shows that the discrete swirl zones can be embedded within longer continuous zones of strong vorticity. Also, the use of height to represent swirl helps emphasize that these zones can surround streamwise streaks of low momentum fluid with significant Reynolds stress.

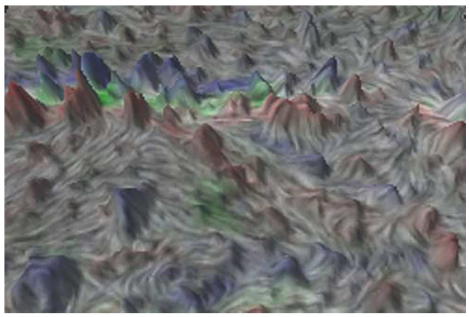

Figure 4.8: The 3D component of height is used to represent the swirl scalar field in a multi-valued PIV dataset.

There are several issues or limitations inherent with using height to represent a scalar field. One issue is that making the domain 3D creates the opportunity for data to be occluded. Interaction through virtual reality devices or rotating the scene can help overcome this problem. Another limitation is that the shading and shadows the are necessary in order to give spatial cues for 3D perception can interfere with the accurate perception of other data fields represent in color or texture.

4.3 Luminance and Contrast Analysis

The role of the luminance component has a prominent effect on how features in an image are perceived [85]. Luminance is defined as the measured amount of light coming from a region of space [84]. In this chapter, we refer to luminance as the perceived reflectance of

a region – a white surface is light while a black surface is dark. This quantity is sometimes also referred to as *lightness* or *brightness* [20].

Contrast is determined by the difference in the color and brightness of the light reflected or emitted by an object within the same field of view. Formally, contrast is calculated as

$$C = \frac{L_{max} - L_{min}}{L_{max} + L_{min}}$$

Where L_{max} is the maximum luminance value and L_{min} is the minimum luminance value within the region.

Manipulations of mean luminance or contrast have the ability to enhance characteristics of an image with the intent of representing a scalar value in addition to the flow data already depicted by a texture. We illustrate these methods on LIC images.

4.3.1 Contrast

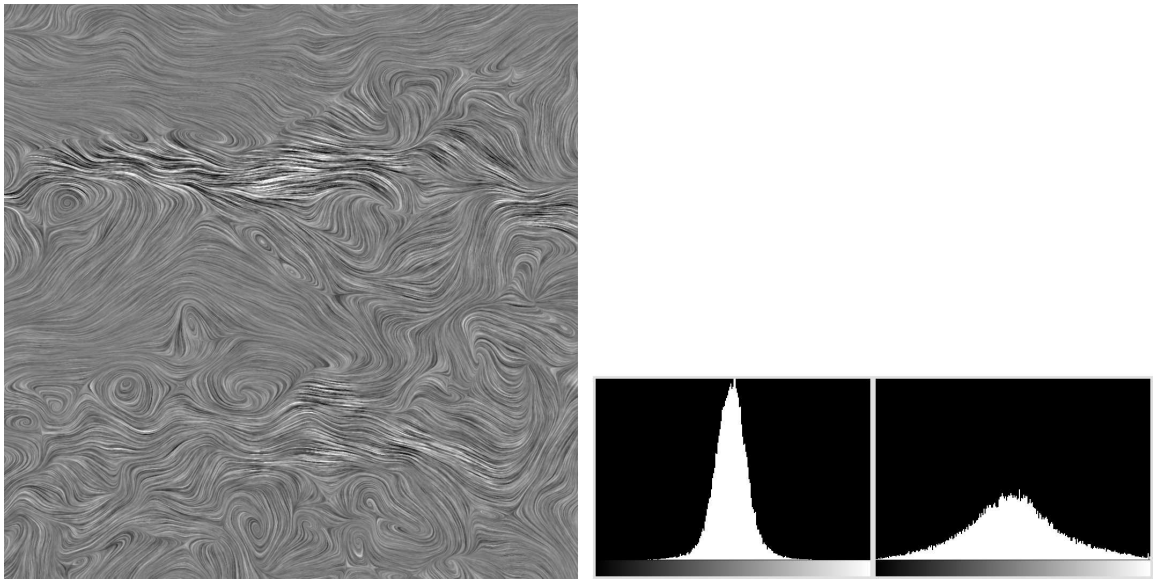

Figure 4.9: Manipulation of contrast is used in this image to represent a scalar distribution (left). Local differences between black and white values are used to portray the calculated quantity of uniform momentum. Two histograms taken from different regions of the contrast image (right).

Differences in the contrast between the blacks and whites can be used to effectively convey

information about a scalar distribution [61]. We describe a method that alters the contrast within the image in accordance with the magnitude of a scalar field.

Once an original LIC image is created to display a flow field, contrast can be manipulated by altering the grey-level values in an image depending on the values in a scalar field that is intended to be displayed. At the locations of prominent scalar values, we reassign pixel values to darker greys or lighter whites than the original. The default values are first analyzed to determine if the pixel value is closer to white or black. The pixel value is then scaled based on the magnitude of the scalar and the previous value of the pixel. A significant scalar value scales a moderately white pixel to an extremely white pixel, and a significant scalar value scales a moderately black pixel to an extremely black pixel. Similarly, if the scalar value is not significant, the algorithm scales the pixel value to be closer to the average intensity. The resulting intensity histogram for the image is different than the original LIC histogram – the contrast within the image at significant scalar value is increased while the contrast within the image at insignificant scalar values is decreased. However, the average pixel value over the new image will remain similar to that of the original. An example is shown in figure 4.9.

Images created in this manner can work to effectively display a scalar distribution because the human visual system is sensitive to different levels of disparity between the blacks and whites in an image. In figure 4.9 (left), local differences between black and white values are used to portray the calculated scalar quantity of uniform momentum. The histograms displayed in figure 4.9 (right) display pixel representations taken from two separate regions of the image. The left histogram depicts the intensity distribution of a low-contrast region. The rightmost histogram depicts a high-contrast region. Both histograms suggest that the average pixel value is in the middle of the range as they are centered and symmetric across the range of pixel values. However, high-contrast regions require the entire range of intensity values while a much more narrow range is utilized in low-contrast regions. Using

the described technique, a continuous scalar distribution can be encoded over a texture image through variations in the internal dynamic range of the pattern.

4.3.2 Mean Luminance

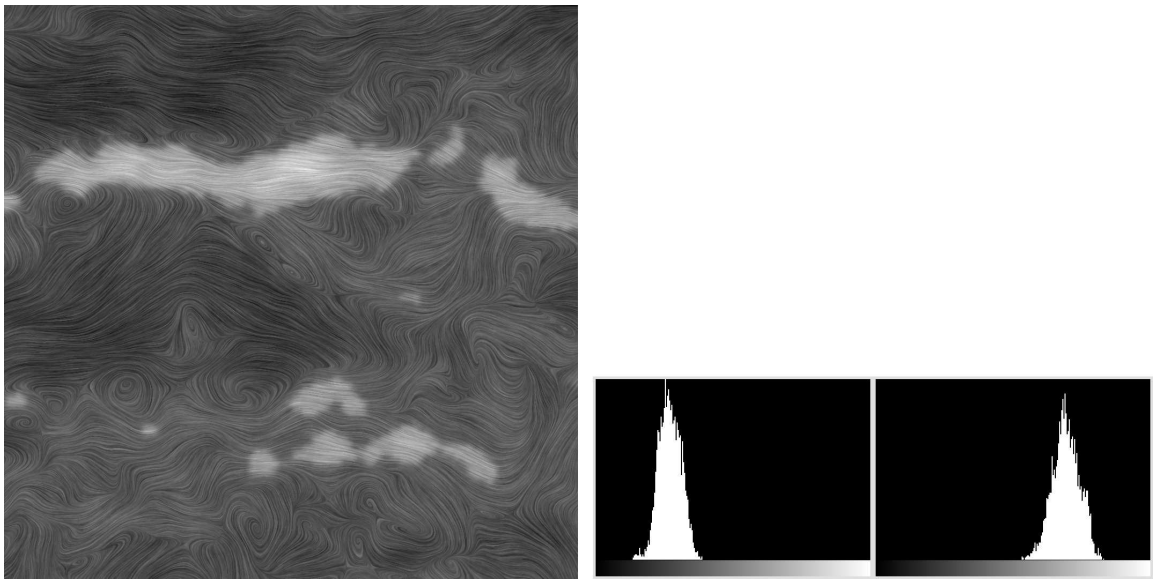

Figure 4.10: (left) Manipulation of mean luminance is used in this image to represent a scalar distribution. (right) Two histograms taken from different regions of the mean luminance image showing that the shapes of the histograms remain similar, and only the average luminance value is changed.

Mean luminance refers to the average intensity value of the pixels in a given region. This quantity is different than the contrast within the image as contrast is a measure of the *differences* between the light and dark values of the image. Mean luminance is a measure of the overall brightness in a region. We describe a method that alters the mean luminance within the image in accordance with the magnitude of a scalar field.

Once an original LIC image is created to display a flow field, the mean luminance of the image can be manipulated by altering the grey-level values in an image depending on the values in a scalar field that is intended to be displayed. The average luminance values of a grey-scale image can easily be altered by adding or subtracting a nominal amount from

each of the pixels. The default 8-bit grey-level values of a LIC image generated from a random white-noise input texture typically have an average value close to 127. The average luminance of the image is changed by adding or subtracting an amount proportional to the scalar distribution. We ensure the luminance value does not go out of range by applying the following formula where α is an arbitrary fixed value determined by the user.

$$I_{new}(x,y) = \alpha * scalar(x,y) + (1-\alpha) * I_{old}(x,y)$$

The α value serves as a balance between the representation of the scalar value and the image in which the scalar value is superimposed. An α value close to 1.0 will emphasize the scalar value at the cost of under-representing the underlying LIC image. An α value close to 0.0 will essentially reproduce the LIC image without much influence from the scalar field.

Applying this formula essentially shifts the histogram with respect to the scalar value while maintaining the overall shape. Using this technique, the contrast between the black and white values of lines remains the same as in the original image and the average luminance value encodes the scalar distribution. Luminance values in an original LIC image are shifted according to the values in an auxiliary scalar distribution. Figure 4.10 illustrates the effects of using mean luminance to represent the scalar quantity of uniform momentum. The histograms displayed in figure 4.10 (right) display pixel representations taken from two separate regions of the image. The left histogram depicts the intensity distribution of a darker region. The rightmost histogram depicts a lighter region. Of note is that the shape of both histograms are similar, and only the average luminance value is changed. Using the described technique, a continuous scalar distribution can be encoded over a texture image through variations in the mean luminance value of the pattern.

Care must also be taken to consider the nonlinearity of intensity perception when using this or any other intensity remapping [48].

4.3.3 Combining Mean Luminance and Contrast

A problem that is inherent with representing multi-valued data is creating an image that accurately depicts multiple distributions in such a way that displaying one distribution does not interfere with the perception of another distribution. This process is further limited by the resolution of the image as there exists a finite number of pixels that can be colored to create the desired visual effect.

In the case of contrast and mean luminance, the two quantities appear to be closely related. Efforts to create a unified image in which two different scalar quantities are differentiated as represented with mean luminance and contrast have resulted in an unsuccessful and ambiguous image. One explanation for this results from examining the respective histograms of the images. As previously described, a scalar field can be represented using the internal contrast by altering the dynamic range of the image while maintaining the same average luminance value. Similarly, a scalar field can be represented using the mean luminance by adding or subtracting an amount respective to the scalar values to the pixel value in the image. In order to combine the two techniques, however, both contrast and mean luminance need to be individually applied to the same image. The underlying problem is that a region that utilizes a high-dynamic range has little opportunity to change the average luminance values as it requires the entire range of values to depict the range of contrast. Thus, it would not be possible for a region of high internal contrast to portray areas of differing mean luminance, and combining the two techniques would not result in an effective representation of both scalar fields.

We next focus our efforts on visualization techniques that allow for multiple fields to be represented and perceived simultaneously.

4.4 Streamline Density

Controlling streamline density facilitates several effective methods of visualizing 2D vector fields. Here we discuss a few variations on previous methods [32, 75] that present information over vector fields by controlling the density of the placement of streamlines. Our methods allow further techniques to visualize additional distributions on these images.

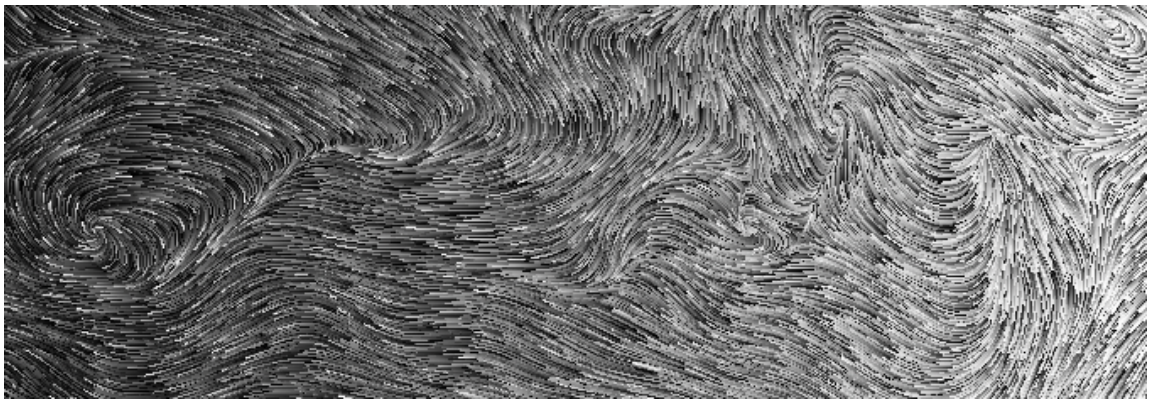

Figure 4.11: Streamline density is increased from left to right to illustrate how a scalar field can be depicted.

A scalar field can be visualized in conjunction with a vector field by allowing the transition within an image from sparse streamlines to dense streamlines. By first creating a sparse and a dense representation of the vector field, the final image can be composited by alpha blending the images together based on the values of a scalar field. Figure 4.11 shows, from left to right, an image of sparse streamlines transitioning to dense streamlines. The image consisting of sparse streamline will inherently have a much lower average luminance value than the dense streamline image. Thus, the final effect of blending the images is essentially manipulating mean luminance to depict the scalar field.

A method to create a sparse texture that accurately reflects a vector field is to begin by using a random distribution to select seed values for streamline calculation. Once a starting point has been selected, the streamline is calculated in both the positive and negative direction. Following [32], the streamline is traced in each direction until one of the following occurs:

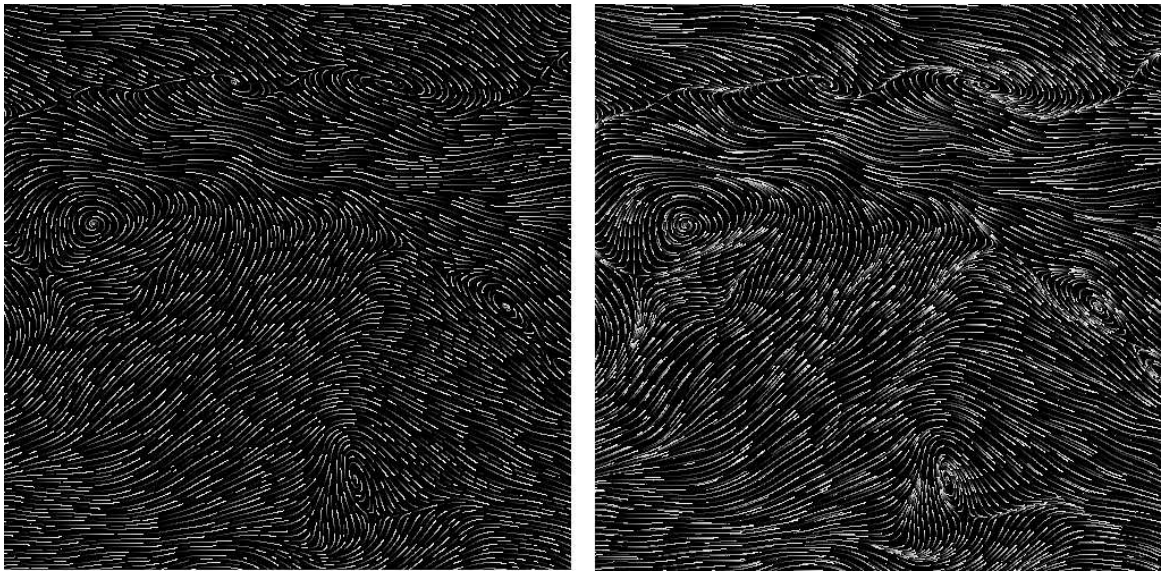

Figure 4.12: A sparse texture of evenly distributed streamlets (left). A sparse texture in which streamlines are not terminated as a function of the proximity to other streamlines (right). This enhances bifurcation lines in the flow.

a singularity is reached, the streamline reaches the edge of the domain, or the streamline comes within some user-defined distance of another streamline that has already been calculated. Intensity values are assigned beginning at the negative end of the streamline. The starting intensity is randomly selected within the range of 0 to 255 and subsequent pixels along the calculated streamline in the direction of flow are assigned monotonically increasing intensity values, wrapping around from 255 to 0, until the end of the streamline is encountered. Streamlines whose total length is less than a user-specified minimum are not colored in. A result of this technique is shown in figure 4.12 (left).

If the restriction of proximity is not enforced, a remarkably different image results. The initial pixel selected at random is checked to determine if another streamline has been computed within a defined proximity. Once a streamline has been initialized, the entire streamline gets defined and colored in regardless of whether it comes too close to another streamline that has already been computed. The effect is that areas where streamlines converge are indicated by a much more dense coverage of streamlines than in the previous technique. As seen in figure 4.12 (right), this increased density highlights the interface

between converging flow regions and can be interpreted as a bifurcation line.

We next discuss a technique that uses the properties of luminance, contrast, and a 3D lighting equation to encode the out-of-plane component of a 3D vector field over a 2D domain.

4.5 Embossing

Applying 3D shading or lighting effects, such as bump mapping or embossing, to 2D images can be an effective method for producing the perception of three dimensional shape.

For the images presented in this section, the additional distribution that we have chosen to visualize is the out-of-plane vector component *w*. The characteristics of this field is significant to various theories and other derived quantities that are valuable for analysis of the turbulent flow data. However, this quantity is often ignored when producing 2D images of 3D vector fields because it is more convenient to simply portray only the in-plane components. With embossing, we can represent the vector field component *w* in a manner that everywhere reflects its depth distance (both positive and negative) from the base plane.

4.5.1 Light Direction

Light direction plays an important role with regard to the perception of depth. Embossing algorithms simulate a standard lighting equation with a single point light source. While this method is highly effective in many situations, artifacts can arise when attempting to emboss vector fields, particularly when the vectors are oriented in the same direction as the light source. To sidestep this problem, we implemented the embossing using two different lights in the plane: one from a source directly above the image, and one from a source

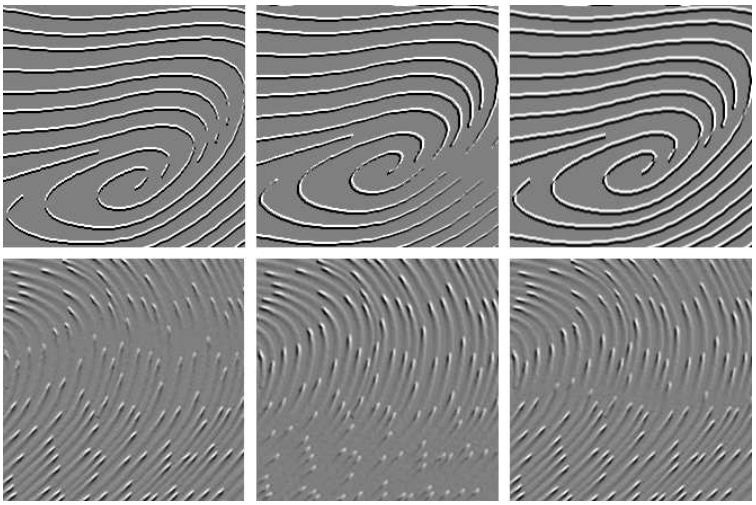

Figure 4.13: Embossing with different light source angles on streamlines (top row) and glyphs (bottom row). Left: Light source from directly above (90 degrees). Vectors vertically oriented aren't illuminated well. Middle: Light source from 45 degrees. In this case, the representation of the diagonally oriented vectors suffers. Right: Both light sources combined.

that is above and to the right of the image, i.e. at a 45 degree angle to the image. The combination of these directions gives the impression of a broad light coming from "above." This produces the visual effect of the embossed image being raised from the surface. We create the composited image on a pixel-by-pixel basis by sampling from the appropriate input image, depending on the orientation of the flow at each point (taking care to always sample from an image that was created using a light source direction that is not aligned with the vector field). Regions of transition are alpha blended between the two images (figure 4.13).

An embossed image that represents a scalar distribution that contains both positive and negative values can be created by combining two images in the following manner. First, an image is created by applying the embossing algorithm with a light source from above, and then a second image is created with a light source from below. The final image is produced by selecting pixel values from the image lit from above wherever the quantity is positive and pixel values from the region lit from below wherever the quantity is negative. The desired effect is that positive values appear to be raised and negative values appear to be

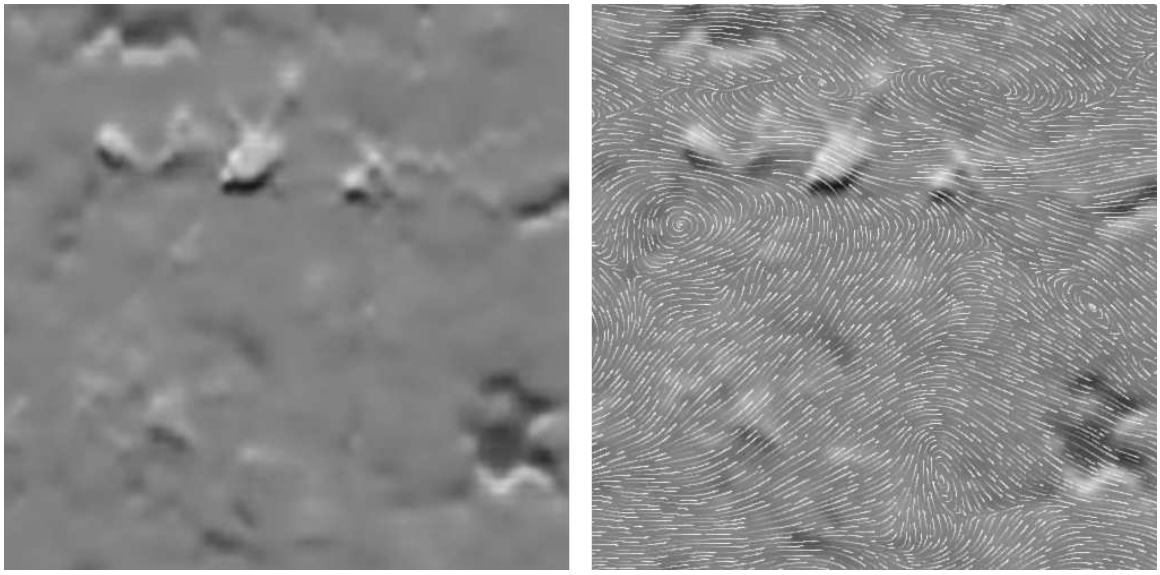

Figure 4.14: (left) An embossed and depth-shaded image representing the gain-adjusted magnitude of the out-of-plane component of a vector field. (right) Overlying a sparse streamline texture on an embossed representation of the out-of-plane vector component.

sunken (figure 4.14 left).

The first method we present to depict the 2D vector field and additional out-of-plane component is displayed in figure 4.14 (right). This image is created by overlaying a sparse texture image and the embossed image of figure 4.14 (left) in such a way that only the values in the former that are lighter than the values in the latter get written to the resulting image. While simple, this technique allows us to effectively visualize both the in-plane and out-of-plane velocity components together in a single image.

4.5.2 Representing Values with Embossed Streamlines

To represent a scalar distribution through the use of embossed streamlines, the magnitude of the scalar value must be encoded in the depth of the embossing. The process is started by creating a discrete number of embossings of the image at different *depth levels*.

If the distribution desired to be displayed has positive and negative components, then two images are created for each level lit in opposing directions – one from above and one

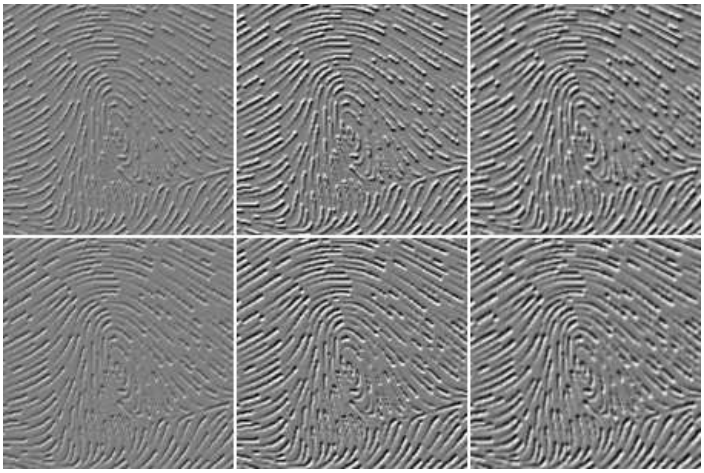

Figure 4.15: Different levels of embossing applied to streamlines to represent the magnitude of an auxiliary scalar distribution. Top row: light from above. Bottom row: light from below.

from below (figure 4.15). We found a linear interpolation between only a few levels to be sufficient to create an effective visualization of the data.

The final image representing the flow field and scalar distribution is created on a pixel-by-pixel basis depending on the magnitude of the scalar component at each point. The value of the scalar field is queried at each point and a linear combination of the appropriate levels of embossing for the respective direction is recorded. This process is continued until all pixels are covered (figure 4.16).

4.5.3 Combining Embossing with Streamlines

A second and more sophisticated approach to representing the flow field in conjunction with the scalar distribution begins with the creation of an embossed streamline image of the vector field. The embossed streamline image is added to the scalar representation once again in a pixel-by-pixel fashion. Once the two pixel values are added, the background color from the embossed streamline image is subtracted. This results in an embossing of the original embossed scalar field as values that were below the average are subtracted from the original image and values above the average are added to the original image (figure

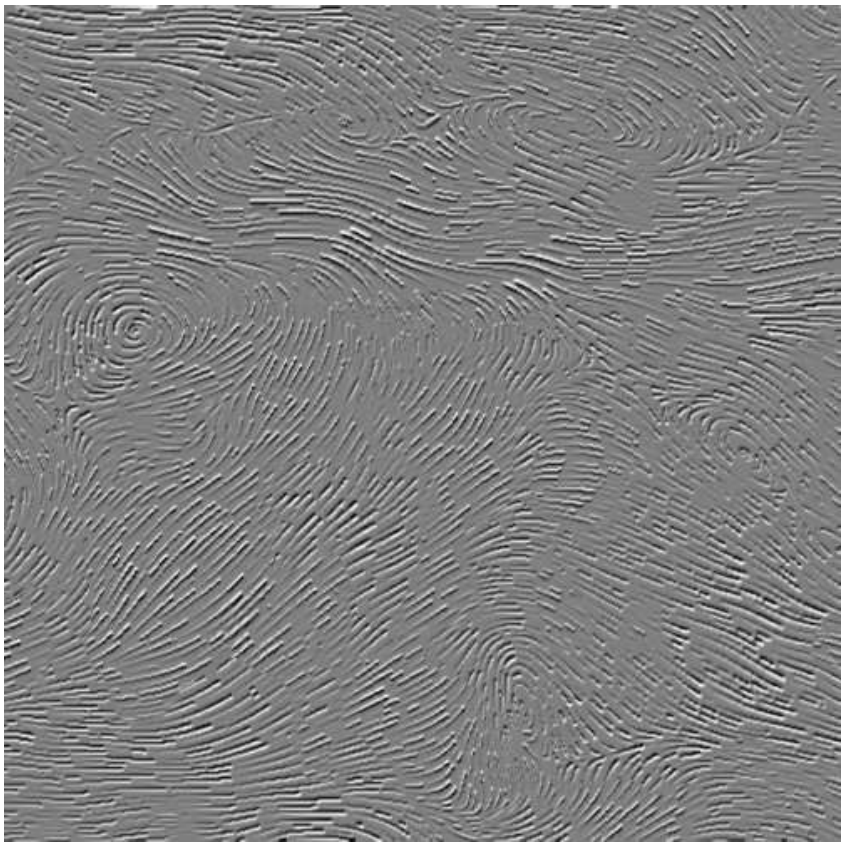

Figure 4.16: Embossed streamlines.

4.17). Any values outside the range of 0 or 255 are effectively clamped.

4.5.4 Limitations

Representing a scalar distribution using this embossing technique is largely impressionistic and contains a few limitations. Namely, the ability to perceive numerous discretized levels of a scalar variable is limited. While all images that reflect scalar values are limited in this manner to a degree, the embossing technique utilizes a number of surrounding pixels and different shades of grey to produce a depth effect. The combination of space and limited number of different levels of grey that can be effectively used to produce the depth effect significantly limit the number of perceptually distinguishable levels using this technique. Additionally, this technique is best employed when the data is relatively continuous and

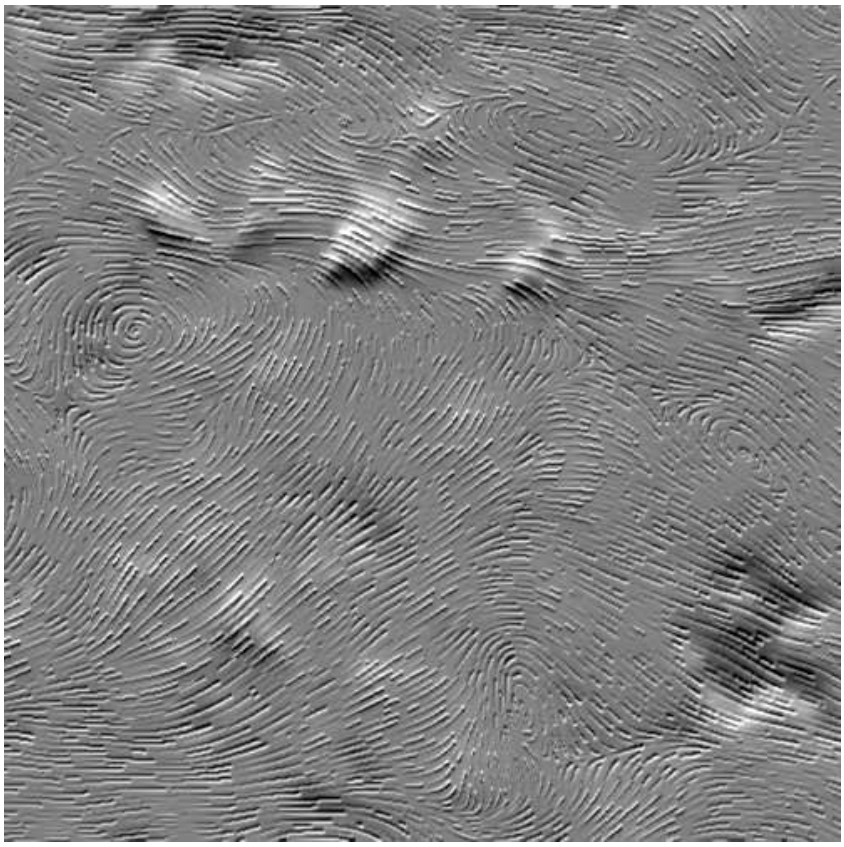

Figure 4.17: Embossed streamlines on an embossed representation of the out-of-plane vector component

does not contain adjacent sporadic positive and negative jumps. This would cause the embossed streamlines to appear segmented in a manner that does not accurately reflect the underlying flow field.

4.6 Discussion

In this chapter we addressed several methods in which a scalar fields may be visualized in a multi-valued dataset.

We have showed that color can be used to represent multiple scalar fields effectively. However, the number of colors that can be overlapped without ambiguity is limited. Color

weaving provides an alternative to traditional color compositing by allowing multiple colors to be closely interwoven via the assignment of distinct separate hues to individual streamlines, rather than blended. The *number* of different colors that are distinguishable when in proximity is an interesting question and a possible direction for future research.

How the visualization is interpreted by the human perceptual system is a significant component of creating a "successful" visualization. We have shown how luminance and contrast can be manipulated within an image to reflect a scalar field, and have also presented a suggestion of why the two techniques are mutually exclusive for representing multi-field data. The human perceptual system, however, is not always a limitation to an effective visualization. By using the 3D perceptual effects of an embossing algorithm, we have introduced a method that can represent a scalar field within a 2D flow field. Effective and accurate visualizations often result through a fundamental understanding of how the human perceptual system interprets visual data.

Chapter 5

Textures

Textures have traditionally been used to visualize vector fields for the purpose of analyzing the form and behavior of flow consistent with theoretical models and to infer the underlying behavior of experimentally-generated flow fields. The use of textures allows for a consistent and highly-detailed representation of a vector field, allowing an observer to both analyze and better understand the dynamics of fluid flow.

Flow textures have traditionally been limited to synthesized renderings where additional attributes are displayed with color, differences in spatial frequency, or contrast enhancements and are added in an artificial manner. In this chapter, we present methods to utilize the qualities and attributes of textures to visualize scalar distributions and vector fields related to a planar velocity field.

A goal of this research is to better understand how the properties of textures can effectively be used to represent various components of flow data. Natural textures have the ability to provide a richly diverse set of possibilities that allow various aspects of the underlying flow field to be visualized. We introduce textures to convey this information in a way that preserves the integrity of the vector field while also taking advantage of the many perceptual dimensions that textures can contribute such as regularity, directionality, contrast, and

spatial frequency.

5.1 Texture Mapping Streamlines

With few exceptions, LIC textures are predominately used in flow visualizations. While effective, LIC textures lack the richly diverse set of possibilities that can be obtained through the use of natural textures. In this section, we illustrate the capabilities of textures as they are applied to streamlines.

5.1.1 Geometry

Applying a texture to a streamline requires that the streamline be extended to include width as textures are inherently 2D and do not project well to streamlines or single pixels [9, 83]. We construct a *thick streamline* by first calculating a 1D streamline given the vector field that defines the flow to be visualized. The streamline is then given width by considering the normal component to the streamline at each point. A user-specified width is multiplied

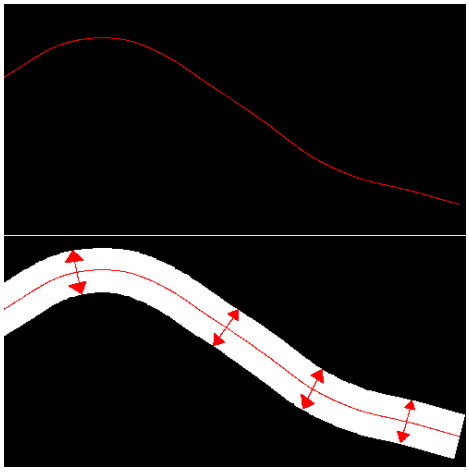

Figure 5.1: A thick streamline is constructed by first calculating a 1D streamline (top). The normal component at each point in the streamline is multiplied by a user-defined width to calculate the coordinates for the thick streamline (bottom).

by the normal component to give the location for each point of the thick streamline. This amount is added to both sides of every pixel in the streamline resulting in a thick, 2D streamline (figure 5.1).

The coordinates of the thick streamline are used to construct polygons in which a texture can be easily applied using standard texture-mapping techniques. Segmenting the thick streamline into polygons allows a texture-mapped streamline to effectively bend and curve in any direction.

An adaptive step size is used during the streamline integration computation to construct polygons that can effectively represent the streamline around areas of high curvature [66]. Using the fourth-order Runge-Kutta formula and given a user-defined error tolerance, an adaptive step size approach chooses a large enough step size to define each polygon while observing the tolerance specified by the user. The effect of this approach is that smaller polygons are generated in areas of high curvature (figure 5.2).

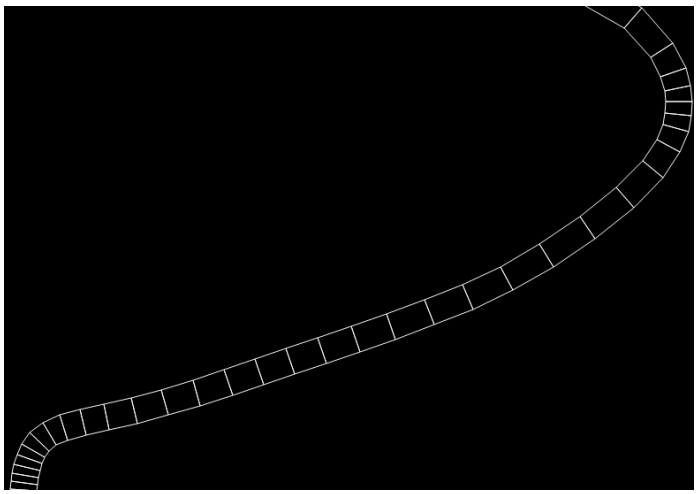

Figure 5.2: Polygons are generated according to an adaptive step-size algorithm that allows for smaller polygons to be generated around areas of high curvature.

5.1.2 Flow Fields

Controlling streamline density allows an entire field of thick streamlines to be created and equally spaced so that applied textures can be perceived. Turk and Banks [75] first address the issue of streamline density. We employ the algorithm developed by Jobard and Lefer [32] for ease of implementation and efficiency purposes.

Once an initial seed point in the 2D domain is selected randomly, a streamline is calculated in both the positive and negative direction. The streamline is traced in each direction until one of the following occurs: a singularity is reached, the streamline reaches the edge of the domain, or the streamline comes within some user-defined distance of another streamline that has already been calculated. A new seed point is generated by randomly selecting a point on the defined streamline and moving it a distance greater than the width of the thick streamline in the direction normal to the flow at this point. Controlling the distance of a new seed point from the previous streamline allows flexibility in the density of the streamlines of the resulting image. To generate an image of dense streamlines, the seed point should be at a distance that is approximately the thick streamline width from the previously defined streamline. A distance that is greater than the streamline width will create more space between streamlines and result in a sparse final image. The algorithm continues by placing seed points in this manner until no more valid seed points can be found.

We have found it beneficial to construct streamlines with maximum possible length when creating the final images presented. Streamlines that do not have a sufficient length are not displayed and another seed point is calculated.

Several artifacts can occur when texture-mapping streamlines. To avoid artifacts that may occur with a repeated texture on the same streamline, a sufficiently large texture sample is used. To avoid artifacts that may occur with repeated texture being applied at the same interval on neighboring streamlines, a random texture offset is used when constructing the first texture-mapped polygon of the thick streamline. These artifacts could also be

avoided by synthesizing the texture separately or along each streamline. Additionally, where portions of streamlines overlap, pixels are assigned an opacity value of zero, giving priority to streamlines already defined. The result is the ability for streamlines to effectively "merge" due to convergence or divergence of the flow but not to obstruct a previously placed streamline (figure 5.3).

Figure 5.3: Using texture-mapped thick streamlines to visualize a flow field.

5.1.3 Texture Outline

Texture-mapping a natural texture to a field of thick streamlines may not create an effective visualization if the orientation of the applied texture is not obvious. Figure 5.4 (top) shows the result of applying an anisotropic texture to streamlines and the ambiguous orientation of streamlines that results. The orientation of the streamlines can be specified by combining the texture with an outline of the calculated streamlines. The outline of the streamlines is constructed by mapping an *outlining texture* to the calculated streamlines defined by the vector field. The outlining texture consists of a luminance ramp, from black to white,

emanating from each side of the texture. The intention is to mimic a diffuse lighting effect that would be created if the thick streamline were three dimensional and tubular in shape. The effect of applying this outlining texture to streamlines is displayed in figure 5.4 (middle). Finally, the two images can be overlaid allowing the orientation of the flow field to be displayed (figure 5.4 bottom).

It is important to limit the proximity between streamlines in creating the outlined streamlines texture. If the outlines of thick streamlines are allowed to overlap, areas of high overlap produce distracting regions when the luminance ramp of the streamline is abruptly halted where one streamline overlaps another. To avoid this artifact, the computation of a streamline is stopped if it approaches another streamline within one half the width of the thick streamline.

Figure 5.4: An illustration of texture outlining used to disambiguate streamline orientation. Top: a birdseed texture applied to streamlines. Middle: the outlining texture applied to streamlines. Bottom: combination of the above two images.

5.1.4 Texture Attributes

We have the ability to show a scalar field in addition to the flow field by changing how the texture is mapped to the streamline. The ability to choose texture parameters freely independent of polygonal geometry provides a great amount of flexibility in depicting scalar quantities.

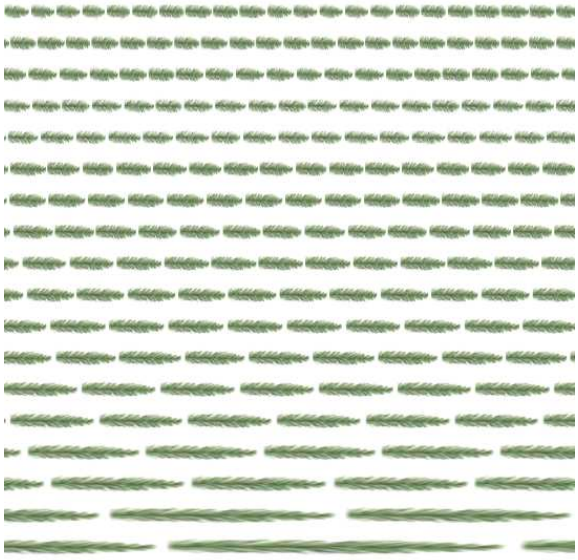

Figure 5.5: Texture parameters can be adjusted to display a scalar distribution in addition to the vector field. The *u* component of the texture is mapped according to an arbitrary scalar component that increases from the top of the image to the bottom.

Starting at the beginning of the streamline, the geometric coordinates for the thick streamline are calculated and vertices for the first polygon are defined. The length of the texture, *u*, or the width of the texture, *v*, can be scaled according to the scalar value and applied to the polygon. Texture continuity between polygons is preserved by ensuring that each polygon starts with the texture coordinates most recently used by an adjacent polygon. Figure 5.5 shows a pine texture representing a scalar distribution, ranging from low to high from the top of the image to the bottom, by scaling the *u* component of the texture while the *v* component remains constant.

Proportionally varying both the *u* and *v* components of the texture influences the relative

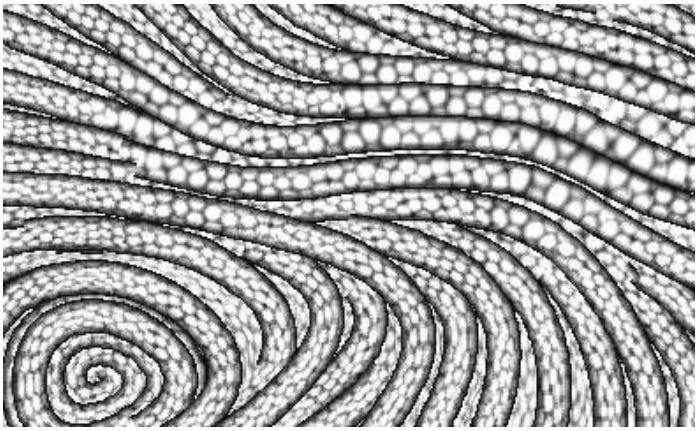

Figure 5.6: An illustration of using texture attributes to represent a scalar distribution. The scale of the texture is directly related to Reynolds shear stress – a scalar field used to characterize regions where drag is generated in turbulent boundary layers.

scale of the texture. This technique creates a difference in the spatial frequency of the texture, reflecting the magnitude of a scalar distribution. Figure 5.6 shows how the scale of a texture can be used to display the magnitude of Reynolds shear stress – a scalar field used to characterize regions where drag is generated in turbulent boundary layers.

5.1.5 Outline Width

In addition to scaling the texture to demonstrate a scalar value, the scaling can be enhanced by adjusting the width of the streamlines according to a scalar quantity [67]. In figure 5.7, the width of the streamlines and scale of the texture correlates to velocity magnitude. In the case of varying streamline widths, the scalar magnitude determines the streamline width and must be taken into account when determining the distance between two streamlines.

A difficulty introduced when streamline width fluctuates is how to handle the discontinuities when thick streamlines come within the proximity of another calculated streamline. The method previously presented suggests that the streamline be terminated. Unfortunately, this leads to an unnatural termination of the streamline and a visual artifact. For this reason, we clamp the thickness of the streamlines to a constant thickness and utilize the texture

Figure 5.7: Varying the streamline width in addition to scaling the texture can accentuate an underlying scalar value.

scale to represent various quantities. An additional limitation is that allowing the outlining streamlines to increase and decrease in size determined by scalar values accentuates the attributes of the texture but reduces the ability to perceive the underlying flow field in combination with the scalar field.

5.2 Texture Stitching

Standard texture-based techniques typically use filter kernels to convolve an input texture and do not utilize streamlines as final part of the image. The next section illustrates a common problem that occurs when an associated scalar variable is visualized along with the vector field using such convolution techniques. The *texture stitching* algorithm presented is designed to encode a scalar variable using spatial frequency while maintaining the fidelity of region boundaries defined by the data.

Figure 5.8 illustrates the classical problem with attempting to apply a color wash to an input texture, before running LIC, to indicate the distribution of values in a scalar field associated

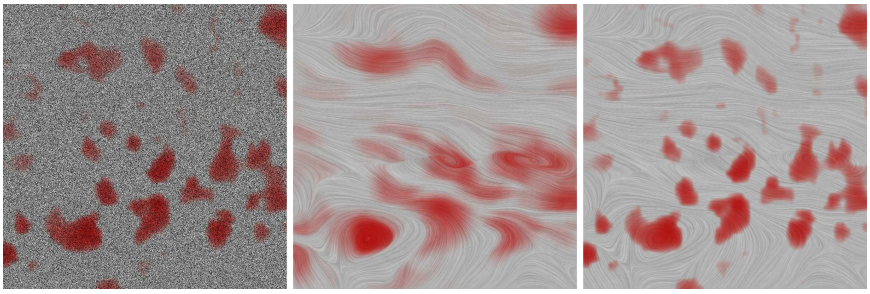

Figure 5.8: An illustration of the problem with trying to use color to indicate regions of interest pre-LIC. Left: color wash applied to the input texture. Middle: results after running LIC – the region definition is not well preserved. Right: results of applying the color wash post-LIC. The goal of texture stitching is to achieve the latter effect with multi-frequency texture patterns.

with the vector data. The effect of the LIC is to smear out the colors, distorting the appearance of the scalar distribution in the final image and impeding efforts to accurately interpret the value of the distribution from the value of the color at any particular point. For this reason, color encoding is universally applied post-LIC, unless it is explicitly desired to use the color to demonstrate the effects of advection [64]. Being aware of these issues with respect to the use of color, and wishing to use spatial frequency to encode the presence of discrete regions of interest in our data, we sought to develop *texture stitching* – a post-LIC variant of the pre-LIC multi-frequency method proposed by Kiu and Banks [41] in which it would be possible to preserve the fidelity of region boundaries implicitly indicated by spatial frequency differences in the texture pattern in the final image, while avoiding the introduction of unnecessary discontinuity artifacts.

Following Kiu and Banks [41], the first step in our approach is to construct a set of correlated noise texture images by low pass filtering an initial high frequency noise pattern and equalizing the intensity histograms of the results to the intensity histogram of the original. For our application we were primarily interested in using spatial frequency to indicate the locations of computed *regions of interest* within a larger surrounding flow field. Thus, it is necessary to generate two noise texture patterns (high and low). To create the images in this section, we applied a Gaussian filter of width 20 and standard deviation

2.0 to the white noise shown in figure 5.9 (left) to achieve figure 5.9 (right).

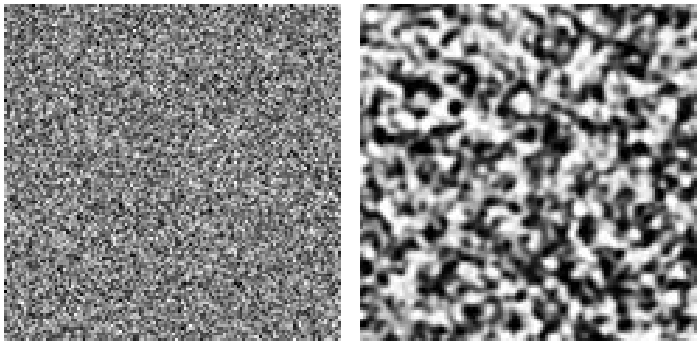

Figure 5.9: Samples from input textures used in our texture stitching technique. Left: the high frequency noise input texture. Right: the low frequency pattern achieved after Gaussian blurring and histogram equalization.

There is a direct correlation between the size differences of the spots in the two input textures and the filter kernel length differences that are required to achieve output textures that will appear to differ by only a uniform (isotropic) scaling factor. Although it is not our intention to use the filter kernel length to encode any information about the flow, a larger filter kernel length with the low frequency input texture is necessary in order to make the lines in the low frequency output texture appear to have the same length-to-width ratio as the lines in the corresponding high frequency pattern. Since the lower frequency lines are less effective at conveying details of the flow orientation, we decided to use the low frequency texture to demarcate the regions of interest, which are characterized by uniform momentum and low velocity.

We proceed by using the high and low frequency noise input textures to create two separate LIC images. We also create a binary mask corresponding to the results of our trial region detection algorithm [18] – one of the goals of the visualization effort is to determine the suitability of the results produced by our region detection method and possibly to provide insight into how it might be refined to achieve greater effectiveness. We use the binary mask to composite the results post LIC. The results of our method are shown in figure 5.10, second from the left . Results obtained using the original multifrequency LIC method are

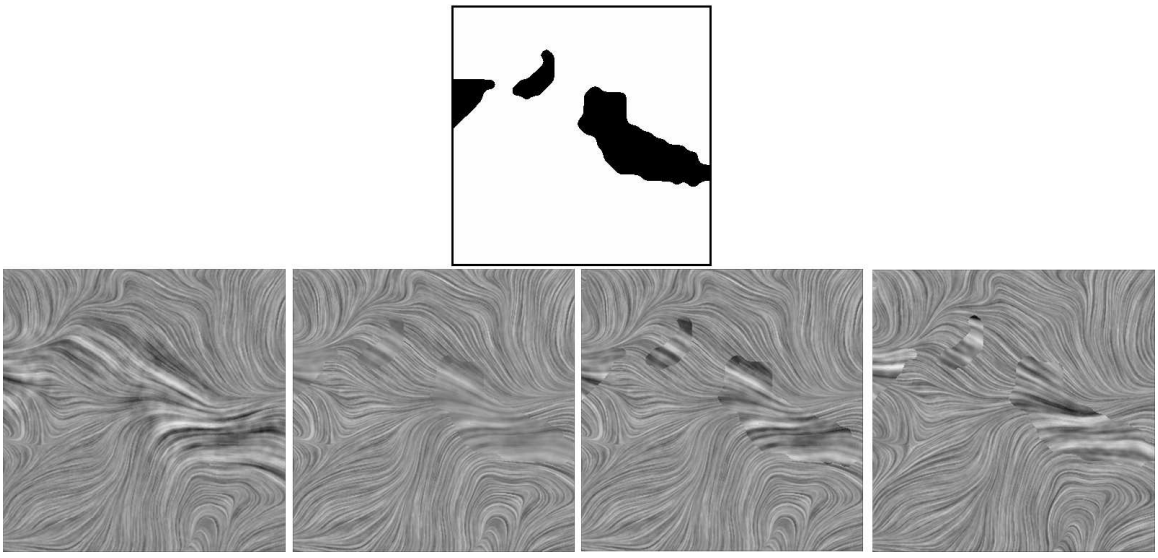

Figure 5.10: Top: the region of interest mask. Bottom: images are obtained, from left to right, using: the Kiu/Banks algorithm; texture stitching with histogram equalization; texture stitching plus contrast enhancement of low frequency regions; texture stitching using unrelated input patterns.

also shown in figure 5.10 (far left) for comparison.

One issue of interest whether it might be desirable, or not, to minimize the incidence of contrast differences between the low frequency and high frequency texture regions. Contrast will inevitably be lower for LIC images obtained from higher frequency input patterns, unless there is a huge reduction in filter kernel lengths, because more different grey values will be averaged together, bringing the result closer to the mean than in the case of the low frequency pattern. Retaining the ability to equalize contrast, which can easily be done in the texture stitching approach, reserves the potential to use contrast differences to encode a different scalar distribution. Figure 5.10, third from left, shows the results of performing texture stitching without contrast equalization. Here the region differentiation becomes more prominent. However, the general continuity of light and dark patterning remains consistent between the regions, which would not be the case if unrelated input textures were used (figure 5.10, far right).

The main characteristic of texture stitching is that it allows region boundaries to be

noticeable in the final image. When compared to the multi-frequency LIC approach taken by Kiu and Banks, the texture stitching approach may not be suitable for applications in which one hopes to approximate a continuous series by a finite set of different spatial frequency patterns, which was the target application for Kiu and Banks.

5.3 Multiple Textures

Texture-mapping streamlines gives great flexibility in the number of different appearances that a vector field representation can have. Figure 5.11 shows a sample of how natural textures are capable of many different appearances when applied to a circular flow. We can use this flexibility and diversity of appearance to represent multiple quantities within the same image. In this section, we present techniques to use multiple textures within an image to represent multiple vector fields and to delineate regions within a flow image.

5.3.1 Texture Splicing

Allowing different appearances within an image can be useful in preserving the fidelity of region boundaries while avoiding the introduction of discontinuity artifacts. The goal of *texture splicing* is to produce an image that achieves a precise depiction of the spatial extents of discrete regions while not detracting from the perception of flow. In the examples illustrated here, the visualization is designed to delineate boundaries of significant structures or *regions of interest* within a 2D image of turbulent flow.

The process of creating the composited image can be conceptualized in three separate steps. First, two textures are selected and the same vector field is used to create two separate flow images. Next, when necessary, an outlined texture of the same computed streamlines is created and overlaid on both images to clarify streamline orientation. In order to avoid discontinuities of the flow, it is important to have both texture images and the streamline

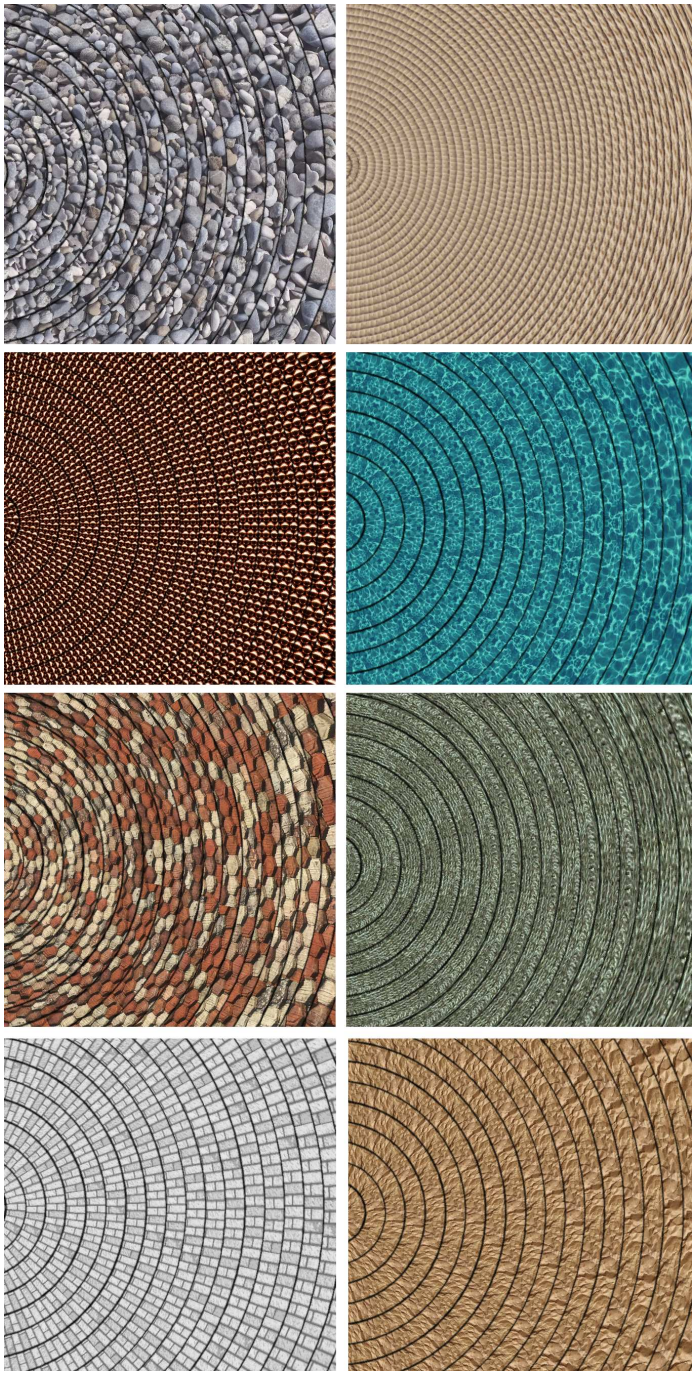

Figure 5.11: Examples of the diversity of natural textures that can be applied to a vector field. A circular flow is used demonstrate each example.

outlines generated from the same computed streamlines. Lastly, the images are composited or “spliced” on a pixel-by-pixel basis, as each pixel in the final image is sampled from the respective source texture determined by the location of the pixel with respect to the region to be delineated. For implementation purposes, it is advantageous for all three conceptual steps to be combined into an efficient one-pass algorithm in which the textures and texture outline are combined then applied along the streamline only in the appropriate region.

Figure 5.12 shows how subtle differences in greyscale textures allow for a specific region to be highlighted. In this case, a region of significant swirl strength is constructed using a shell texture. A texture comprised of coins, similar in size to the shell texture but different in luminance properties, is used for the other texture which allows for the swirl center to be visualized clearly. The spiraling streamlines around the area of swirl strength denote that the relative speed of the observer matches the convection velocity of this particular swirl center.

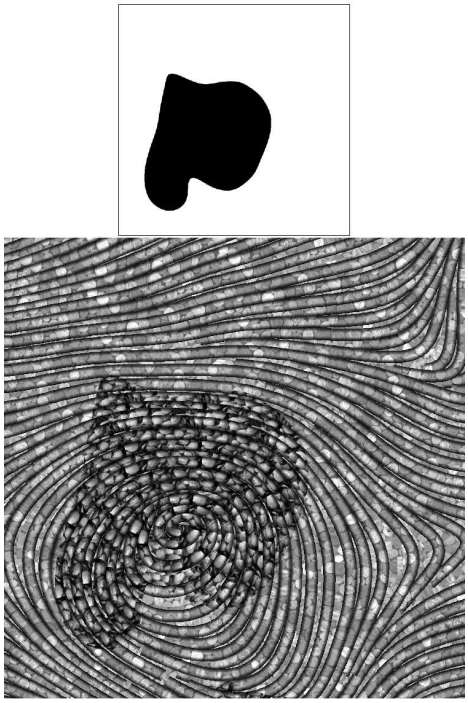

Figure 5.12: Top: Binary region of high swirl strength – a quantity used to identify coherent vortices. Bottom: A greyscale coin texture and shell texture are spliced to delineate the region of high swirl strength.

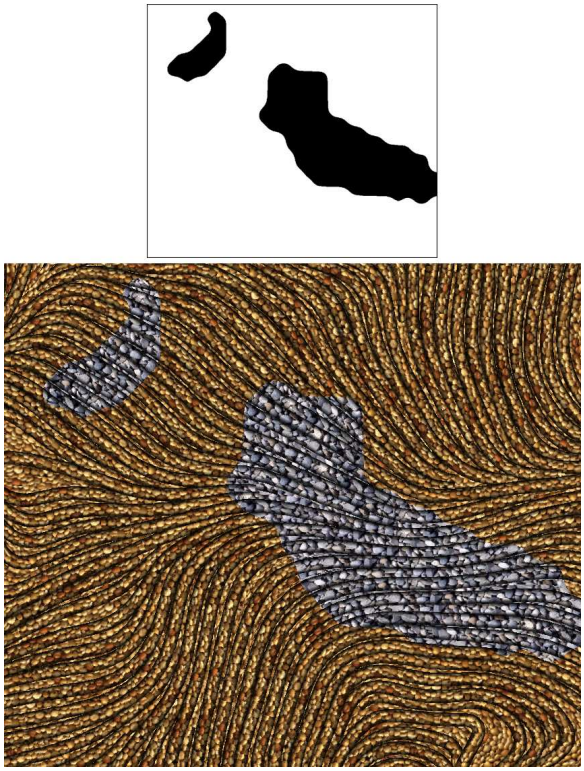

Figure 5.13: Top: Boundary regions of potential vortex packets defined by a feature extraction algorithm. Bottom: A birdseed texture and a stone texture are spliced to specify the boundaries.

There is great flexibility for image appearances using different natural textures. Figure 5.13 illustrates an example using two unrelated textures, a birdseed texture and a stone texture. The region delineated by the stone texture represents boundaries of a potential vortex packet identified with a specific feature extraction algorithm [18]. The size of the components and luminance of each texture chosen are similar so they are not prominent factors in the perception of the difference in texture. While the two textures are most differentiable by color, the regions are still noticeable when the image is viewed as a greyscale image. The natural qualities allow each texture to be distinguishable while the flow of the streamlines remains easily perceivable.

5.4 Discussion

With few exceptions, LIC textures are predominately used in flow visualizations. While effective, LIC textures lack the richly diverse set of possibilities that can be obtained through the use of natural textures. In this chapter, we have shown the flexibility of textures as they can be mapped in a variety of ways to show properties of flow field.

Mapping textures to calculated thick streamlines provides a computationally efficient method for creating a texture-based flow image. The user is given a wide range of flexibility to create a visually rich image using this technique. We have also introduced a method to depict flow direction in the case of anisotropic textures.

Texture stitching and texture splicing allows for a precise depiction of the spatial extents of a discrete region without detracting from the perception of flow. Developing techniques using more sophisticated methods for combining textures within flow visualization to represent multi-field data is a potential direction for future research.

Chapter 6

Multiple Vector Fields

This chapter presents strategies for developing effective methods for the visual representation of multiple co-located vector fields. One of the most challenging aspects to visualizing multi-field data is creating an image that accurately reflects the key physical properties of all of the fields in a way that does not allow for a bias towards one distribution. These methods are designed to allow each field to be understood and analyzed both individually and in the context of the other.

We begin by exploring a variety of different techniques that allow for visual representations to be combined in order to represent two vector fields simultaneously. Several existing techniques for visualizing single vector fields are examined and combined to demonstrate the effectiveness of simply compositing representations to produce an image representing multiple fields. We also consider how elements from vector fields can be distinguished or grouped preferentially through techniques of embossing or using different textures.

We examine the method of using streamlines to represent flow fields and how streamlines from different vector fields can be combined to effectively represent multiple vector fields. We develop new methods, based on methods created for a single vector field visualization, to effectively use streamlines in a manner that allows both vectors fields to be viewed

simultaneously.

Finally, many scientific applications involve data in which the most challenging aspect of interpreting the results is determining how separate fields are related. The modeling of 3D magnetohydrodynamic light supersonic jets and the study of coherent structures in turbulent boundary layers are two such applications. Illustrations from these applications are later presented in this chapter.

6.1 Integrated, Multiple Vector Fields

The visualization of multiple co-located data fields can be done in several simplistic ways. One method is to display the individual distributions in a side-by-side manner, so that each field can be seen clearly and independently. However, spatial correspondences between distributions are not easily revealed using this technique. Another approach is to show multiple fields sequentially in the same space and allow the user to quickly alternate between the images. With this approach, however, it is difficult to obtain a reliable, *integrated* understanding of the multiple fields. This research investigates new methods for the simultaneous representation of multiple fields in a manner that leads to a further understanding of correspondences or interrelations between two vector fields.

6.1.1 Classification of Single Vector Field Techniques

Mining the knowledge base of previous visualization research yields important findings and insight to assist the research of our specific applications. Using this insight, we are able to re-construct, manipulate, and expand upon the existing state-of-the-art in order to further the process of knowledge discovery related to specific tasks and conditions.

Many different techniques have been developed for the visualization of a 2D flow. Various

visual techniques have characteristics that allow for different components of the flow to be more visually salient [45]. We classify existing 2D vector field flow visualization techniques into one of three visually distinct categories: texture-based, line-based, or glyph-based.

Texture-based techniques are characterized by a consistent, highly-detailed, and dense representation of a vector field. As presented in section 2.1, many texture-based techniques have been developed for the visualization of 2D flow as it allows for the fine details of a vector field to be easily displayed and analyzed [4, 31, 79].

Line-based techniques involve the display of elongated *streamlines* – segments that are everywhere tangent to the vector field. These images are fundamentally different than the images created with texture-based techniques as streamlines can be visually traced over a long distance, and line-based images are more sparse than textures. While streamlines give a global sense of flow by depicting paths along the vector field, flow direction is not always apparent. Additionally, locating an advantageous small number of critical lines representing the flow can be a challenging problem.

Glyph-based techniques are characterized by the use of repeatable icons that can express various types of information about the flow, including flow orientation. In a sense, line-based techniques can be thought of as a special case of glyph-based techniques. The most basic glyph technique is the use of hedgehogs (sometimes referred to as vector plots) – short line segments or arrows aligned with the flow direction at regularly or randomly distributed locations. Glyphs provide the ability to depict flow direction by using the glyph shape or a luminance ramp. We also include in this category techniques that involve placing glyphs along calculated streamlines.

6.1.2 Combining Multiple Images

Transparency or layering is one of the most effective methods to portray relationships between overlaying components. Kirby et al. introduced a method inspired by the layering techniques used in oil paintings to visualize components of 2D flow data [40]. Interrante has used transparent stroked textures on 3D surfaces superimposed over underling opaque structures [28]. Weigle et al. demonstrated a texture generation technique, based on the layering of oriented slivers, using orientation and luminance to encode multiple overlapping scalar fields [89]. Hotz et al. vary the input texture density and spot size in addition to kernel length and overlay sparse LIC images to visualize features of a tensor field [25]. Wong et al. developed a technique that combines elements of alpha channel manipulation, filigreed graphics and elevation terrain mapping, along with colormap enhancement to visualize a multi-variate climate dataset [91]. Our research is inspired by these techniques, and seeks to expand the applications of layering in the visualization of multiple related vector and scalar fields.

Considering one sample from each of the three visually distinct methods for vector field visualization (texture, line and glyph) we explore the range of effects that can be achieved using layered combinations of these approaches to visualize two different, co-located vector fields (figure 6.1). For the texture-based sample, we use LIC – the most widely used texture-based approach. For the glyph-based sample, we create an image by repeatedly texture mapping a comet-like glyph along thick, equally spaced streamlines. This results in a sequence of glyphs that is continuous in the direction of the flow. The line-based sample is created using the source code supplied by the authors of an equally-spaced streamline technique [75].

One of the primary goals in each of the applications is to depict the key physical structures of one vector field in the context of the key physical structures in another. Therefore, it is highly desirable to be able to easily differentiate between the visual representations of

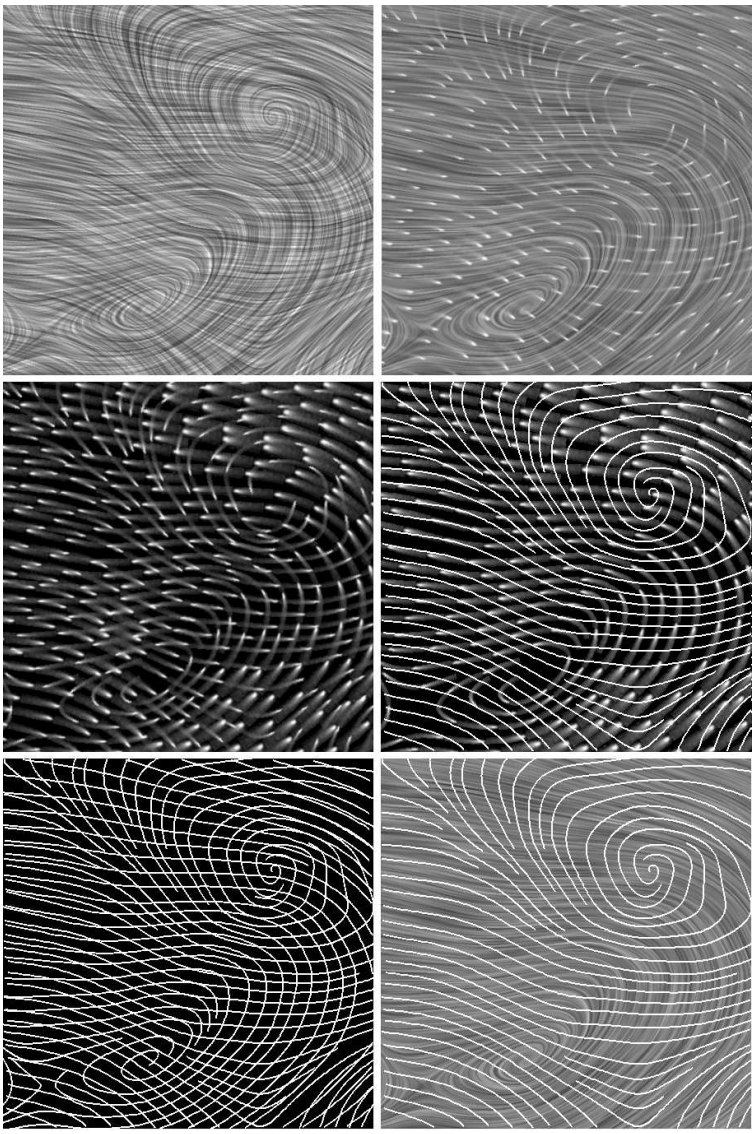

Figure 6.1: Different 2D vector visualization techniques applied to the visualization of two co-located vector fields.

each field and to easily identify each distribution. This process becomes more complicated when the same representational technique (e.g. line, texture or glyph) is applied to each field. While a supplemental visual variable, such as color, could be used to distinguish the two representations, we would ideally prefer to be able to reserve the use of color for the communication of related, co-located scalar quantities.

The problem of differentiating between each vector field is illustrated on the left hand side of figure 6.1. The combination of two texture-based techniques (upper left) provides a

rich representation in which both vector fields are visible, particularly in regions where the vector field orientations are not aligned. However, it becomes challenging to distinguish the two datasets in regions where the orientations are aligned as it is difficult to determine which field is being represented by which texture at any given location. Both the glyph and line techniques utilize an inherently sparse streamline spacing that leaves empty space in the image. Combining two sparse techniques creates a large amount of negative space which can result in a lost opportunity to represent fine details of the vector fields. When two glyph-based techniques are overlaid (center left image of figure 6.1), maintaining good visual continuity or "good continuation" in the direction of the flow in each field becomes a challenge, particularly when the glyphs are separated by a nontrivial amount of empty space. When two line-based techniques are overlaid (lower left image of figure 6.1), the continuity of streamlines is maintained; however, the accidental patterns of intersections between streamlines in different vector fields can lead to visual artifacts that may cause an inaccurate perception of the data (figure 6.2).

When a different representational approach is used for each field, the primary challenge

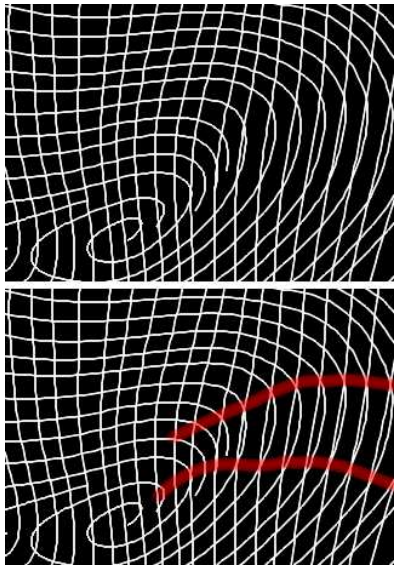

Figure 6.2: Combining streamline representations can lead to visual artifacts. Top: Two streamline images are combined. Bottom: the artifact caused by intersecting streamlines is highlighted

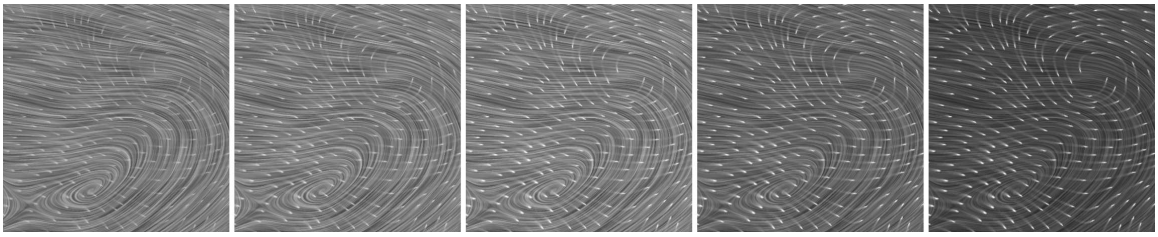

Figure 6.3: Different representations can be emphasized by altering luminance properties of individual representations prior to image compositing. In the sequence from left to right of images above, the glyphs become brighter while the texture becomes darker. When the glyphs are very subtle (left image), the vector field represented by the texture is more prominent. When the contrast between the white glyphs and the darkened texture is more obvious (right image), the vector field represented by the glyphs is more prominent.

is to appropriately balance the visual prominence of each representation, so that neither overwhelms the other in the combined presentation. Figure 6.3 illustrates how different representations can be made more or less prominent by altering the contrast and luminance values prior to layering the two images. Similarly, differences in line thickness can be used to make one vector field more prominent when displayed with another (figure 6.4).

In our experience, particularly good results can be achieved by layering a relatively sparser, higher contrast, glyph or line-based representation of one field over a relatively denser, lower contrast, texture-based representation of the other field. The high contrast between

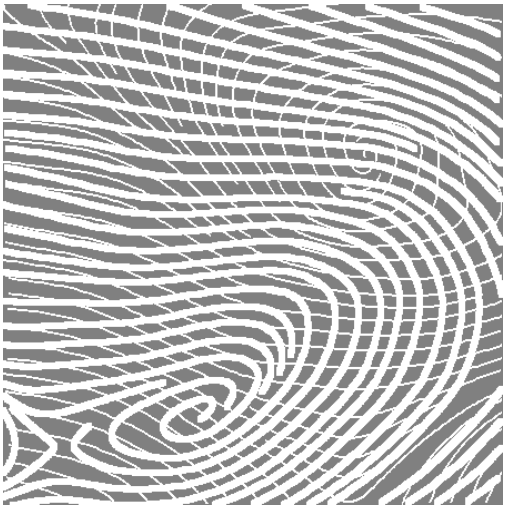

Figure 6.4: Differences in line thickness cause one vector field to appear more prominently than the other.

the glyph or line elements and the background or empty space enables them to maintain their visibility when superimposed over the textured background; the relatively sparse distribution of the glyph or line elements in the top-layer flow is necessary to enable the simultaneous, effortless appreciation of the flow that is portrayed on the underlying layer.

Next we describe the methods that we implemented to allow for two different representations to be combined in such a way that each representation remains distinct and visually separable.

6.2 Methods for Combining Images

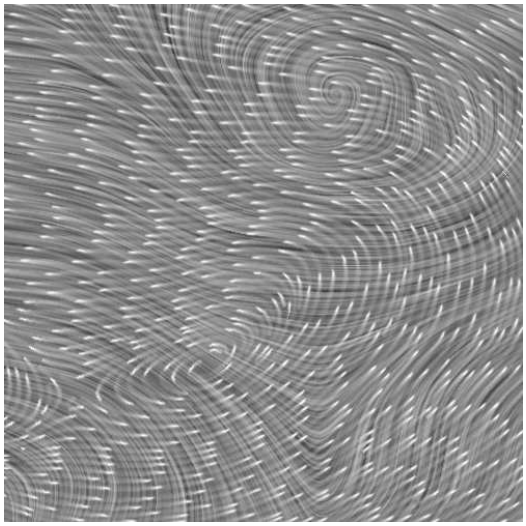

Figure 6.5: Glyph-texture mapped streamlines overlaid on a LIC texture mapped background

6.2.1 Overlay

In figure 6.5, two representations are combined using a *screen overlay* method. This method is similar to a standard image multiplication operation but operates on the additive inverse of each input image. Specifically, if $D(x,y)$ is the output image and $A(x,y)$ and

$B(x,y)$ are the respective input images, and we assume that each pixel intensity is between 0.0 and 1.0, the *screen overlay* operation is defined as

$$D(i,j) = 1.0 - (1.0 - A(i,j)) * (1.0 - B(i,j))$$

Applying this formula allows the intensities of the underlying texture image to show through in the places where the overlying glyph image contains empty (black) space. This technique is most effective when using an image with a black background so the resulting image is highlighted only in regions where the glyph is placed. Of note in figure 6.5 is that the two vector fields can be easily distinguished not only where their orientations are nearly orthogonal, but also in the places (such as the upper right corner) where their orientations are nearly aligned. It is precisely to enable these sorts of comparisons of the relative alignments of the two fields that we pursue these investigations of methods for their combined portrayal.

6.2.2 Embossing

In the overlay method, we primarily rely on luminance differences to distinguish the overlying glyph or line elements from the underlying texture image. As introduced in chapter 4, embossing is an alternative technique that can be employed to distinguish an overlaid image from an underlying image [77]. In this method, each element of the overlaying image is given a distinct 3D visual appearance, in order to enable the elements to perceptually group preferentially with each other and at the same time jointly segregate from the background.

Where it is relatively straightforward to emboss streamlines and glyphs, dense textures typically do not contain a sufficient amount of empty space within the image in which to portray the result of the shading equation. Textures can be embossed by utilizing

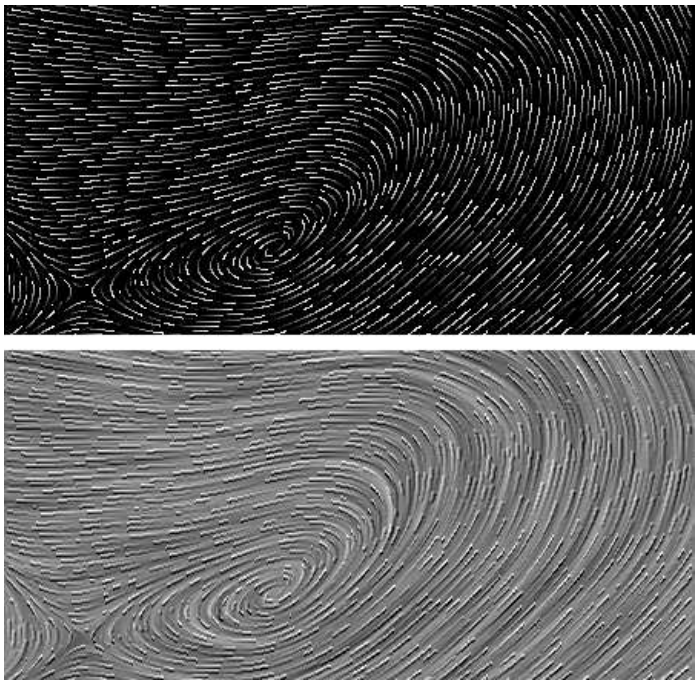

Figure 6.6: Embossing a texture. Top: A sparse texture in which the effects of an embossing algorithm can be perceived. Bottom: An embossed sparse texture combined with a LIC texture.

an algorithm that creates a sparse texture to which the embossing algorithm can then be applied (figure 6.6). The sparse embossed texture and the dense texture are then combined to give an embossed appearance to the texture.

An embossed image can be overlaid on another image by adding the intensities in the two images on a pixel-by-pixel basis, and then subtracting the value that the background color takes in the embossed image. This is equivalent to increasing the intensity in the non-embossed image at the points where the intensity in the embossed image is above average, and decreasing the intensity in the non-embossed image at the points where the intensity in the embossed image is below average (figure 6.7). The embossing, in effect, gives a visual distinction of an applied 3D lighting equation and is visually separable from the non-embossed image.

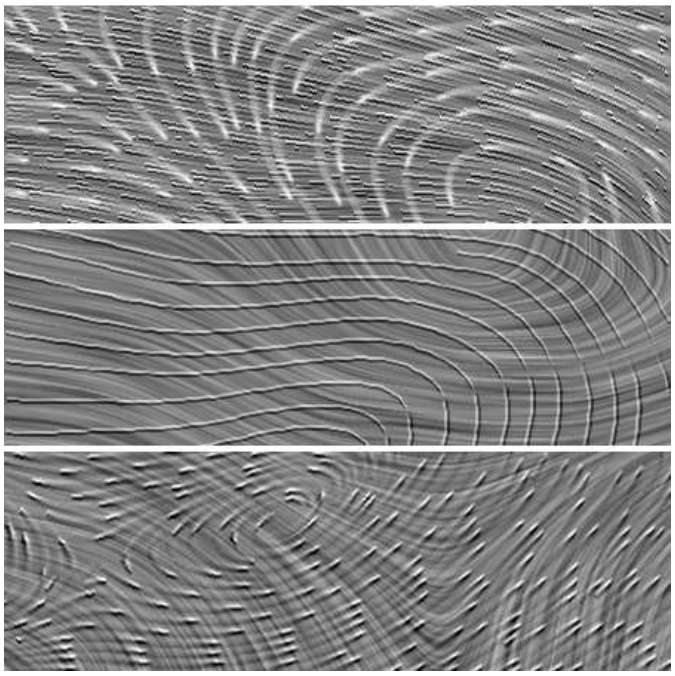

Figure 6.7: Distinguishing different fields by embossing. Top: embossed texture composited with glyphs. Middle: embossed streamlines composited with texture. Bottom: embossed glyphs composited with texture

6.2.3 Multiple Textures

Different techniques such as texture, lines, or glyphs can be used to represent a flow field. These techniques can effectively be combined to depict multiple co-located vector fields by using the different visual characteristics of each technique and luminance differences to delineate the individual vector fields. Similarly, embossing one of the techniques allows for the elements to perceptually group preferentially with each other and at the same time jointly segregate from the background. Next, we focus on how individual attributes of textures can be used to distinguish two texture-based representations of different vector fields. We introduce a technique that facilitates the differences in textures to allow for the analysis of the form and behavior of a vector field within the context of an additional vector field. The terms *overlaying* and *underlying* are used to describe the two layers of vector field images that comprise the final image.

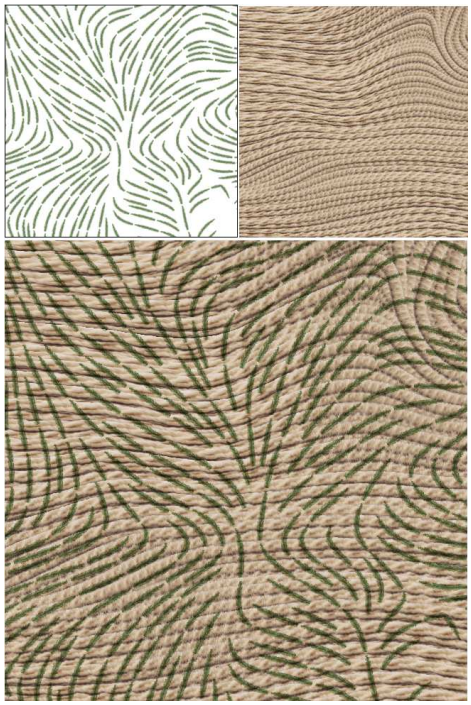

Figure 6.8: Two different textures to represent different vector field representations. Top left: The overlaying image. The pine texture represents the in-plane vorticity vector field. Top right: The underlying image. The yarn texture represents the in-plane velocity vector field. Bottom: the composited image. The continuation of the texture patterns and spaced streamlines allow for both vector fields to be viewed simultaneously.

Since two textures cannot co-exist at the same location, we employ an equally-spaced streamline algorithm with a large distance between streamlines to depict the overlaying vector field. We have found success in using glyph-like textures, such as the pine texture, to represent the overlaying vector field as the orientation and direction of the vector field are easy to perceive without the need to use an outline texture (figure 6.8 top left). Both vector fields in this example have been magnified and interpolated to a scale that allows the sparse streamlines to cover the domain adequately without sacrificing accuracy by not displaying streamlines over the entire domain.

The underlying texture has more flexibility in the type of texture and spacing of the streamlines available to represent the flow accurately. A texture with dense streamlines works well to display the flow field. In figure 6.8 (top right) a yarn texture is used with

densely packed streamlines. The u component of the texture is used to reflect velocity magnitude which results in the yarn being "stretched" in areas of high vector magnitude.

The composite image of the overlaying and the underlying images is composed on a pixel-by-pixel basis. Final pixel values are calculated by taking the normalized product of respective pixel values from the two individual vector field images. The long texture-mapped streamlines give a unique and natural continuation to the vector field which allows for the two vector fields to be visualized simultaneously. To increase the perception of both flow fields within the same image, we have found the best results using two textures that contain unrelated color schemes.

6.3 Interweaving Streamlines

The method of overlaying one image on top of another image is most effective when the methods of representation are different for each vector field being displayed. As presented in section 6.1.2, different representations allow for the different visual attributes of each technique to be distinguished when combined. However, when two streamline images are combined, it becomes difficult to distinguish the physical structures of one vector field from the other.

Many algorithms exist for the seeding or placement of streamlines of a single vector field [33, 49, 75, 92]. The goal of these algorithms is to place streamlines in a way to avoid visual clutter and accurately depict the key physical structures of the vector field. We utilize several of these algorithms and present techniques for the placement of two vector fields, representing both fields accurately, with equal prominence, and minimizing the visual artifacts that may occur when streamlines are combined.

6.3.1 Streamline Overlay

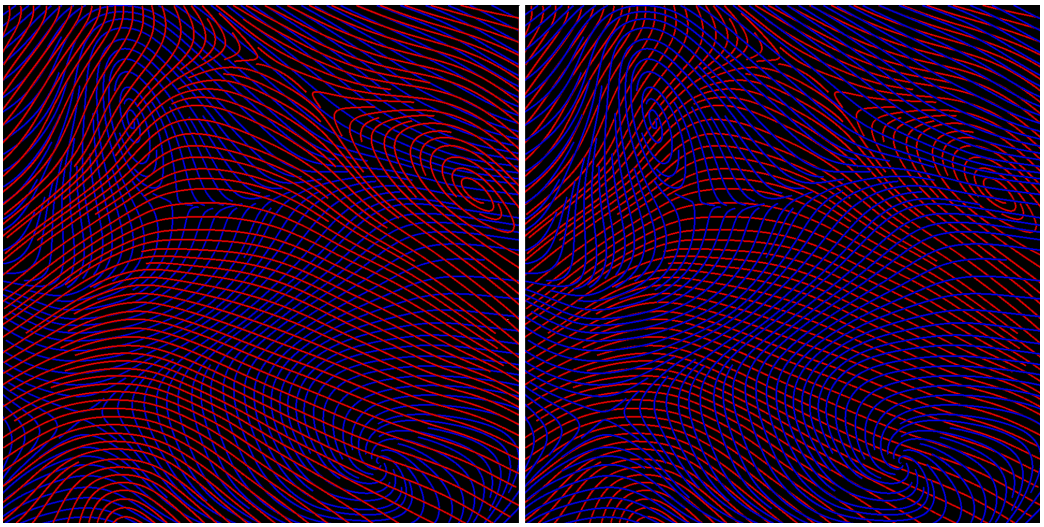

Figure 6.9: Overlaying streamline images result in the overlaying image being more prominent than the underlying image. Left: blue streamlines over red streamlines. Right: red streamlines over blue streamlines

In an effort to illustrate how to visually represent two vector fields simultaneously, we examine two streamline representations distinguished by color (blue and red). In figure 6.9, two different equally-spaced streamline representations are simply overlaid. The red streamlines overlay the blue streamlines in the leftmost image; the blue streamlines overlay the red streamlines in the rightmost image. While the color of the streamlines distinguishes each vector field appropriately, the vector field in which the streamlines are "on top" is more prominent than the underlying vector field. As one of the goals of this research is to represent both vector fields without one being more visually prominent than the other, this simple technique produces an undesirable result. This research focuses on methods that will produce images representing two vector fields in which the key structures and characteristics of both fields can be seen without a bias of one vector field over another. For example, figure 6.10 is constructed by *interweaving* the streamlines of both vector fields such that one representation is not "on top" or more visually salient than the other vector field. The algorithm for constructing overlapping streamlines is presented next.

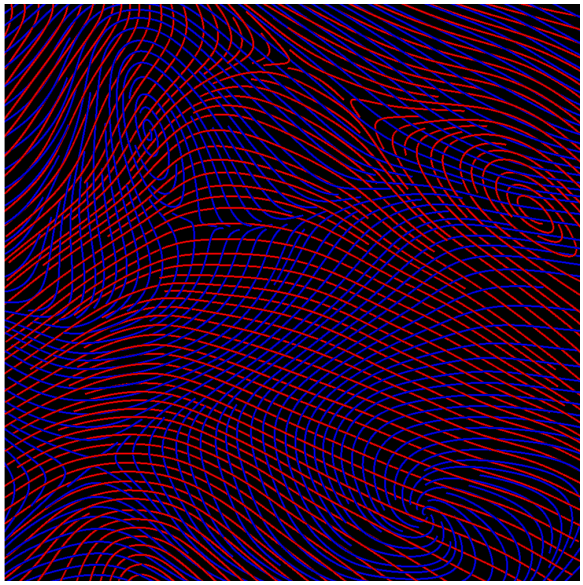

Figure 6.10: Interweaving streamlines allows for each vector field to be represented equally

6.3.2 Streamline Weave

One of the most important concerns in depicting the interweaving of streamlines is how to graphically represent the overlapping elements to be distinct from each other. Artists and illustrators have traditionally used the technique of line continuity to depict one line passing underneath another. The size, quality, thickness, and discontinuity of lines allows for an enriched vocabulary in the representation of visual objects [12, 58]. Inspired by [29], we define a 2D texture "halo" around the streamlines as defined in section 5.1.3, to not only give it a 3D appearance, but also to visually separate it from streamlines from a different vector field at a point of intersection (figure 6.11). While the vector fields represented are defined on the same 2D plane, discontinuities in the lines are utilized to allow for the perception that one line is underneath the other when two lines cross.

The effect of streamlines weaving over and under other streamlines from a different vector field is achieved by altering the opacity values of the polygons that constitute the streamline. This method allows for the computational complexity of the algorithm to be minimized as the geometry for intersecting polygons does not have to be explicitly calculated. The

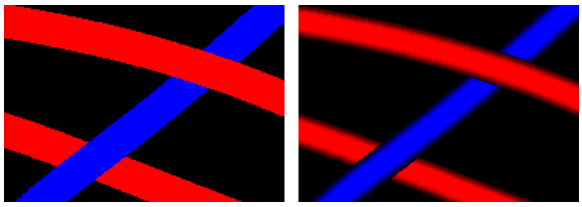

Figure 6.11: Left: Color continuity depicts that one line is underneath the other when two lines cross. Right: A 2D texture "halo" adds a 3D appearance and an additional visual cue of line discontinuity to depict one line crossing over another line.

intersections are handled through changing the opacity of the streamline as it intersects with the streamlines already placed. Thus, by altering the opacity values for a streamline being drawn, it can effectively pass over and under streamlines that have already been computed and displayed.

The additional data structure of an alpha buffer must be maintained in order to determine which pixels have been covered by the streamlines of the first vector field; the opacity values are also available using the **glReadPixels** command supplied by OpenGL. All pixel values covered by polygons representing the streamlines are assigned a value of 1.0, and all other pixels are assigned an opacity value of 0.0.

The first step in the process of displaying inter-weaving streamlines is to determine where the polygons that constitute the streamlines will be placed. Assuming the seed points for the streamline placement have already been determined by an equally-spaced streamline algorithm, the geometry for the thick streamline can been calculated – as presented in section 5.1. Thus, the only criterion that remains to be determined is how and when the opacity of the streamline should be displayed as opaque (as it passes over an existing streamline) or should be displayed as transparent (as it appears to pass underneath).

The algorithm works by weaving one vector field amongst the streamlines defined by the accompanying vector field. Initially, the streamlines from the first vector field can be placed without the need to consider the second field. The more computationally-intensive process of computing the second vector field's streamlines will essentially be woven into

the structure of the first vector field's streamlines.

The second vector field is introduced one streamline at a time. Each streamline is rendered starting at the furthest most point of negative integration along the streamline and continues forward until it reaches the complete streamline length. At the beginning of the streamline, a random variable determines whether it will begin by passing over or under the first streamline of the opposing vector field it encounters. If it is determined to pass over the first streamline, the OpenGL blending function is set to replace the pixels already stored in the frame buffer to the pixels being computed by the polygons of the second vector field – thus, writing over the first vector field and passing over the streamline. This blending function is maintained until the streamline being drawn has completely crossed the streamline from the first vector field. This condition is met when the value of the alpha buffer for all four corners of the calculated polygon have opacity values of 0.0. The blending function is then changed to allow for the streamline to pass under the next intersection (figure 6.12). The polygons representing the streamline are drawn until the next intersection occurs. At this point, the opacity values of the first vector fields' streamline will hold, and areas of the polygon that are not opaque will be shown – as if the streamline is passing under the previously defined streamline. When the alpha buffer values for the four corners of the polygon are once again 0.0, the blending function values are flipped to allow the streamline to pass over the next intersection. This process continues for all streamlines and polygons,

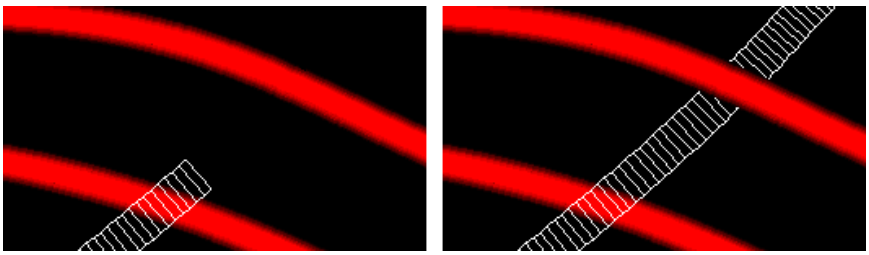

Figure 6.12: Illustration of how the weaving streamline is constructed. Initially, the blending function is set so the streamline will pass over other streamline (left). When four corners of a polygon are clear from the underlying streamline, the blending function is changed to allow the streamline to pass under the next streamline it encounters (right).

alternating between the blending function parameters to allow for the streamlines to appear to be passing over and underneath each other.

Eliminating Artifacts

Figure 6.13 illustrates a problem than can occur by interweaving two equally-spaced streamline representations. In a single vector field, the algorithm that determines the placement of the streamlines is designed to avoid situations in which streamlines become "too close." However, when the placement of these algorithms are combined, it is possible that streamlines will overlap in areas where both vector fields are parallel. The streamlines becoming coincident results in an clustering artifact that the equally-spaced algorithm had avoided when dealing with only a single vector field. Next we look at two solutions, based on previously defined equally-spaced streamline algorithms, that consider both vector fields when determining the location for every streamline.

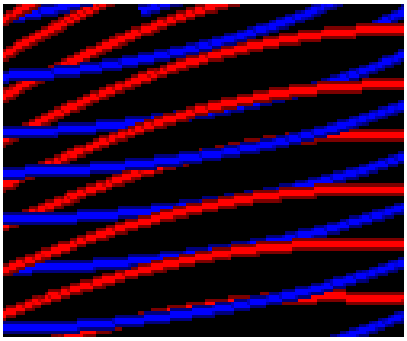

Figure 6.13: A region of figure 6.10. The equally spaced streamline algorithms do not account for the other vector field resulting in the potential for streamlines to overlap in regions where the vector fields are parallel.

Image-Guided Streamline Placement

Turk and Banks [75] developed an algorithm that uses an energy function to guide the placement of streamlines at a specified density for a single vector field. While computationally expensive, the technique provides a method in which the placement and

length of each streamline can be refined in order to create a collection of streamlines that are equally spaced. The algorithm employs a low-pass filter of the streamline image to measure the separation and quality of the placement of the streamlines. Quantifying the "quality" of the placement of streamlines by using the low-pass filtered image allows for a measure of the difference between the current image and the desired visual density. The quality can be increased by changing the positions and/or lengths of streamlines, joining streamlines that are nearly touching, or creating new streamlines to fill sufficiently large gaps. After an initial set of streamlines are placed at random or on a regular grid, the process iterates by making changes in streamlines, checking to see if the changes would increase the quality of the image, and finally adopting the changes if the alteration would improve the quality of the image. The original algorithm effectively creates an image of equally-spaced streamlines for a single vector field.

In order to incorporate two vector fields, several alterations to the original code must be made. First, when streamlines are first introduced to the domain (Turk and Banks use the term "birthed"), each streamline is assigned one of the two vector fields in an alternating manner. Thus, when each streamline is being drawn, it will be defined according to the vector field that it was assigned when it was first created. Care must also be taken in the process that joins streamlines that are nearly connected. In this case, it is necessary to check to make sure both streamlines are defined from the same vector field prior to joining as it is possible that two abutting streamlines may not be from the same vector field.

Utilizing the low-pass filtered image to minimize the energy function with two vector fields yields interesting results. The quality metric using the low-pass filter ensures that the streamlines are equally-spaced and a similar density is maintained throughout the image. By introducing a second vector field, it is inevitable that streamlines from different vector fields will cross. However, crossing streamlines causes a high penalty in the metric defined by the low-pass filtered image as regions of intersection will not have a similar density as

regions of non-intersection. The areas of intersection result in an area of higher density in the low-pass filtered image, which is above the target density. Figure 6.14 illustrates the effect of the low-pass filter in regions of intersecting lines. Thus, applying the original image quality metric of the Turk and Banks technique will yield patches of streamlines of one vector field instead of overlapping streamlines (figure 6.15).

As this adaptation of original technique does not give a representation in which the fields appear *integrated*, we further modify the algorithm to allow the coverage of each individual vector field to contribute to the energy function. This is done by redefining the quality metric to include a low-pass filter image of each of the individual vector field streamlines in addition to the global set of streamlines.

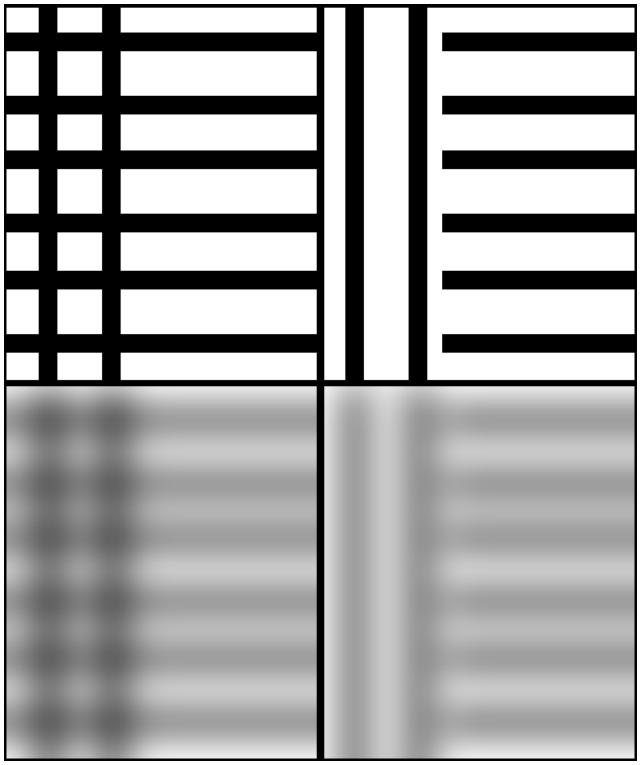

Figure 6.14: Example of a low-pass filter on two different line patterns. Top: Two images of lines. Bottom: a low-pass filter of the above line images. The tendency for the original image-guided streamline placement algorithm to avoid intersecting lines as explained by the even distribution of the low-pass filter on the non-intersecting lines (right) compared to the intersecting lines (left).

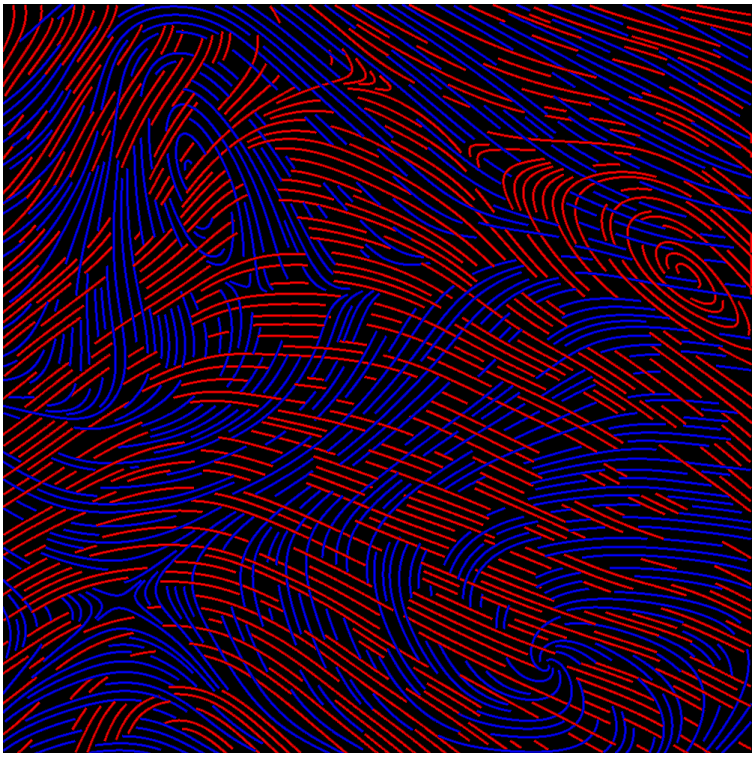

Figure 6.15: Two vector fields using the original metric of a single low-pass filter. The metric induces a high penalty for intersecting streamlines resulting in areas where streamlines from only one vector field are present.

As streamlines are being added to the image, each streamline is assigned one of the two vector fields in an alternating manner. Each streamline is additionally added to two of three low-pass filter images: one that contains all of the streamlines, one that contains streamlines only from the first vector field, and one that contains streamlines from only the second vector field. The density of the low-pass filter of these three images is added, in equal parts, to define the final quality metric. This ensures that areas in which no streamline for a vector field exist are minimized as this would result in a high penalty from the low-pass image that represents the coverage from that particular vector field. In turn, this encourages streamlines to overlap as desired. The algorithm continues as defined by Turk and Banks and each streamline is placed, altered, and adjusted according to the new metric. In the event that the alteration improves the overall quality (including the overall coverage of each individual field and the global set of streamlines) the change is applied.

The addition of the low-pass filter images for the individual vector fields does not dramatically increase the computation time of the original algorithm. The quality variable for each of the low-pass filter images is stored as a variable, allowing for an efficient look-up. The most computationally expensive portion of the algorithm exists in the process of determining the updated quality for the potential moving of a streamline. The original algorithm must compute the contributions of each pixel in the streamline to derive the update quality. Adding this functionality to the additional low-pass filter images proves to be negligible as the streamline's location and properties have already been determined in the original code. Only a few more additions were necessary to complete the implementation.

The results of the improved algorithm are displayed in figure 6.16. The addition of the individual vector field's low-pass filter to the quality metric allows for the streamlines to overlap. Additionally, the global coverage component of the quality equation ensures

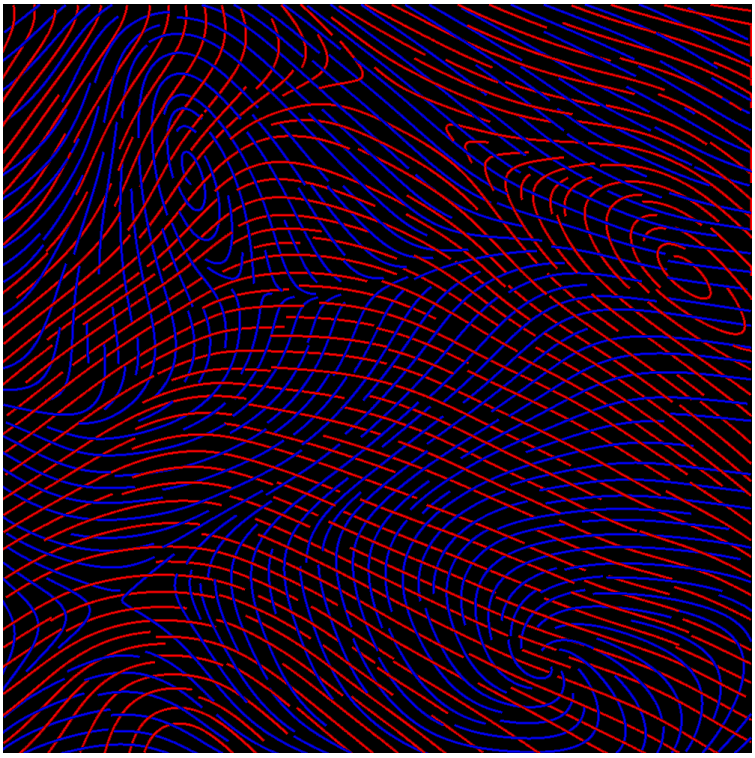

Figure 6.16: The coverage of the individual vector fields are added to the quality metric of the Image-Guided Streamline Placement algorithm to allow for each vector field to be equally distributed producing a globally balanced streamline distribution

that regions where streamlines from different vector fields are parallel do not contain overlapping, coincident streamlines.

Seed Point Streamline Placement

The equally-spaced streamline approach introduced by Jobard and Lefer [33] provides a computationally efficient method to place streamlines for a single vector field. The algorithm works by initially placing a seed point at a random location and computing the interpolating streamline. As shown in figure 6.17, new seed candidates are selected that are a user-defined distance from the points on the previously defined streamline. Streamlines are constructed from these seed points continuing until it reaches a singular point, the end of the domain, or it comes within a user-supplied distance of another, previously defined streamline. This process of determining new seed points at a distance from existing streamlines continues until no other seed points can be found.

We extend the Jobard and Lefer algorithm to incorporate two vector fields and minimize the coincident intersections that lead to a large amount of negative space in the final image. The goal is to create a final image in which the two vector fields are as mutually exclusive as possible – allowing for streamlines of both fields to be as equally-spaced from streamlines

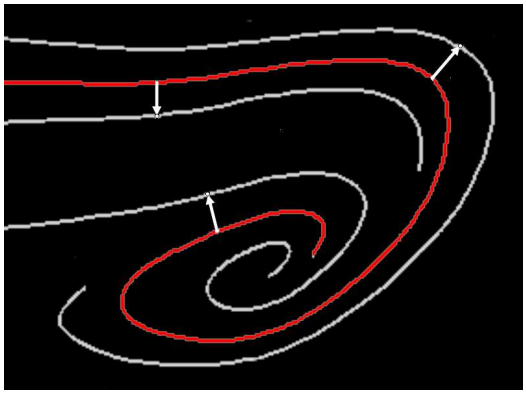

Figure 6.17: The Jobard and Lefer algorithm selects seed points at a equal distance away from a calculated streamline. The original streamline is colored red. Equally-spaced streamlines are colored grey.

in the same vector field and in the other vector field being simultaneously visualized.

We consider two different spacing distances: the first is d_{sep}, the user-defined distance in the original algorithm that is used to define the distance for the seed points away from a calculated streamline. Since the streamlines will be constructed from different fields on an alternating basis, we essentially use a value proportional to $d_{sep}/2$ as to allow streamlines of different vector fields to be "in-between" the equally-spaced streamlines of the same vector field. The second spacing distance, d_{test}, is the stopping criteria that the distance in which a new streamline can be within another streamline from the same vector field before it is stopped because it is "too close." Note that the d_{test} criteria is only enforced with streamlines of the same vector field, as intersections between the vector fields will occur naturally and should not be suppressed.

The algorithm begins, in the spirit of the original Jobard and Lefer algorithm, by selecting a random point and constructing a streamline. The algorithm progresses by alternating the vector fields in which the streamlines are drawn. The next candidate seed point is selected to be a user-defined distance, $d_{sep}/2$, away from the previous streamline drawn. However, unlike the original algorithm designed for a single vector field, we also consider the other vector field to be visualized when selecting potential seed points.

In addition to creating two vector fields that are equally spaced, we seek to eliminate the phenomena of overlapping streamlines when both vector fields are parallel. In order to accomplish this we scan all points on the previously drawn streamline to find the location in which the two vector fields are most closely aligned. Computationally, this is an inexpensive process as each vector field is considered along the finite number of pixels that constitute the streamline. The angle between both vector fields is calculated by evaluating the dot product between the corresponding vectors. Given a vector A from the first vector field, and the corresponding vector B from the second vector field, the measure of the angle between them, θ, can be calculated as $A \cdot B = |A||B|\cos\theta$. Furthermore,

normalizing the vectors results in the dot product providing the value of the cosine of the angle. Taking the absolute value of this quantity will yield a maximum value of 1.0 when the vectors are perfectly aligned (0 degrees) or perfectly opposite (180 degrees). Thus, the algorithm simply finds the maximum value along the streamline of the dot product for the corresponding normalized vectors.

The point in the streamline where the vector fields are the most closely aligned is used to calculate the next seed point at a distance of $d_{sep}/2$ perpendicular from the streamline. Following this step guarantees that the streamlines of both vector fields will not be coincident at this point. A list of appropriate seed point candidates is stored for each streamline based on the alignment between the vector fields at each point. The process of computing seed points along streamlines for alternating vector fields where the corresponding vectors are aligned continues until a valid seed point does not exist. Seed points may not be valid for a number of reasons: the seed point is within the distance of d_{sep} of a streamline from the same vector field, the resulting streamline length is below a user-defined tolerance, the seed point is at a critical point. In the case that a seed point is not valid, the algorithm begins again with streamlines already computed to find appropriate seed points in which the vector fields are aligned using the list of seed point candidates stored when the streamline was first computed. As with the original Jobard and Lefer

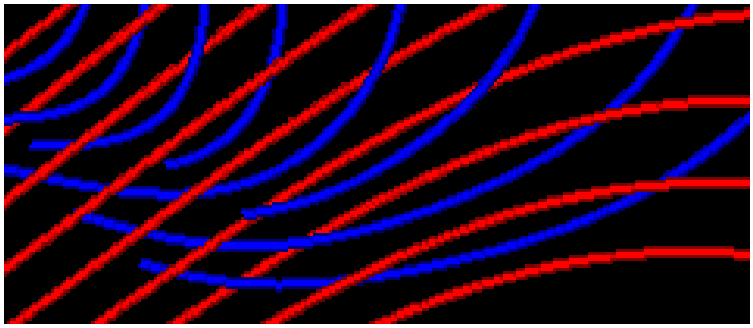

Figure 6.18: Adaptations to the Jobard and Lefer algorithm allow for effective placement for seed points of two vector fields. Alternating streamlines from different vector fields allows the streamlines to be “in-between” other equally-spaced lines.

algorithm, streamlines are drawn in the fashion of selecting seed points in areas where the vector fields are most aligned until no other seed points can be found (figure 6.18).

6.4 Applications

In this section, we employ the methods previously presented to several specific applications, and explain the contributions to the scientific discovery process made using these techniques. The discussion emphasizes the visualization technique utilized and the scientific conclusions that can be inferred. The reader is referred to chapter 2 for the background explanation for the individual applications. Effective results of simultaneously visualizing multiple vector fields are illustrated through three different scientific applications: the visualization of velocity and vorticity fields in experimentally acquired turbulent boundary layer flow data, the visualization of velocity and magnetic fields in computational simulations of astrophysical jets, and the visualization of different layers within a numerical simulation of a turbulent channel flow.

6.4.1 Experimental Turbulent Flow

We use the methods described to visualize data acquired from experimentally generated dual plane PIV experiments. Our goal is to further understand the relationship between the vorticity vector field, velocity vector field, and 2D and 3D swirl strength scalar distributions. The characteristics of vortex cores, including their orientation given by 2D and 3D swirl, are useful in theories designed to reduce skin-friction drag in turbulent flow. Standard PIV datasets, however, are limited to 2D calculations without the out-of-plane gradients. The 2D and 3D swirl distributions are further discussed in conjunction with velocity and vorticity vector fields in this section.

We first analyze the in-plane velocity vector along with the 2D and 3D swirl fields. The

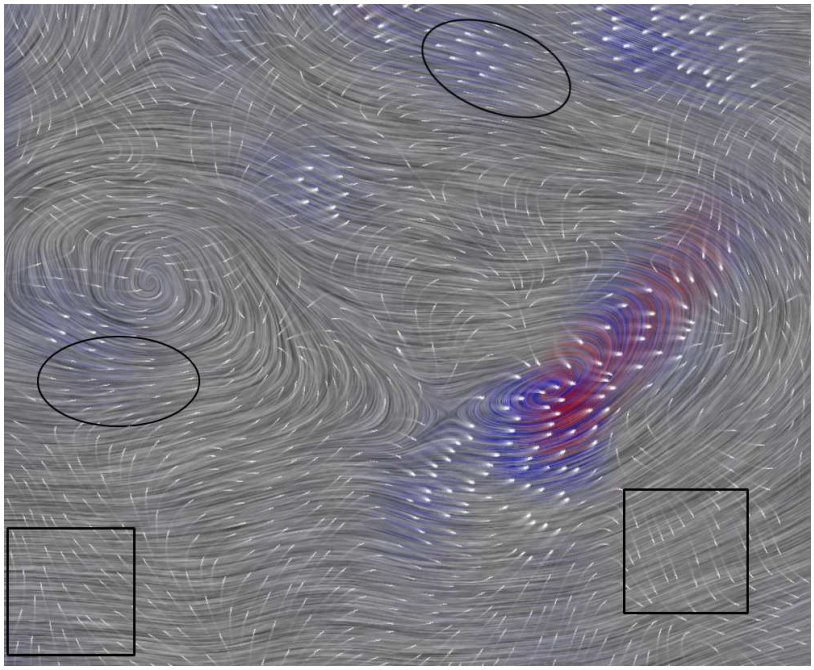

Figure 6.19: Visualization of multiple scalar and vector fields from a dual plane PIV experiment. The underlying LIC texture represents the velocity vector field. The field of glyphs represent the vorticity vector field.

velocity vector field is visualized using a LIC texture and the 2D and 3D swirl variables are encoded in color using a technique that maps each color to alternating streamlines to avoid the ambiguity that occurs when colors are mixed [76]. The LIC texture serves as a method to convey the underlying vector field while simultaneously providing the means to distribute the color representing additional scalar fields.

The velocity vector field is obtained by subtracting the mean streamwise velocity from the in-plane velocity component. Thus, the locations of critical points are subject to the relative velocity of the observer and are not germane to the analysis in this example. Swirling streamlines in the vicinity of the area of high swirl strength will only occur in the vicinity of the particular vortex cores that convect at a rate similar to the mean streamwise velocity.

Analyzing the relationship between the 3D swirl and 2D swirl components leads to a valuable understanding the of the orientation of a vortex core. As 3D swirl is calculated using all gradient components, it can accurately measure the existence of a vortex core at

any orientation to the referencing plane. The 2D swirl distribution only measures the swirl of a vortex core that is oriented orthogonal to the plane. Accordingly, 2D swirl is present only where there is also significant 3D swirl.

To develop an understanding of how the vorticity vector field contributes to the phenomenon of swirl strength and the potential for a vortex core, we *screen overlay* the velocity vector field LIC image with a glyph image representing the vorticity vector field (figure 6.19). The thickness of the glyphs is defined to be directly proportional to the in-plane vorticity magnitude. The two vector fields displayed together reveal zones where velocity and vorticity are nearly perpendicular (regions within a square) and nearly parallel (regions within an oval) for the particular convection velocity being visualized. The region in the center of the image exhibiting strong 2D swirl (red) also contains strong in-plane vorticity, as evidenced by thick glyphs, and additional 3D swirl (blue), suggesting a vortex core that is inclined at a significant angle to the measurement plane. The remaining zones of 3D swirl that lack a prominent indication of 2D swirl represent cores whose vorticity is more closely aligned with the measurement plane. The glyphs give the predominant vorticity direction of the core.

The integrated visualization of these components allows for vorticity direction and magnitude to be correlated with the direction of a vortex core delineated by swirl strength. Effectively combining techniques for the visualization of multiple vector and scalar fields allows for a better understanding of vortex frequency, strength, and orientation as well as the 3D interactions among and caused by vortices. This technique can be particularly useful in the analysis of large fields characterized by the occurrence of multiple interactions [19].

6.4.2 Astrophysical Jets

The modeling of magnetohydrodynamic light supersonic jets in the context of astrophysical galaxy clusters is one avenue of active research in the field of computational physics [51].

In addition to the velocity field, physicists are concerned with the magnetic vector field advected by these supersonic jets and are interested in analyzing the relationship between both vector fields. Developing a deeper understanding of the relationship between the vector fields' topology, magnetic field strength, and corresponding velocity structures results in a greater understanding about how kinetic and magnetic energy distributions evolve in these systems and contributes to the explanation of radio emissions that can be physically observed.

In addition to understanding the relationship of the topology of both vector fields, astrophysicists are interested in the interplay between magnetic field strength and the corresponding velocity structures as magnetic field enhancement naturally results from shear and compression in the flow. Through simultaneous visualization of the simulated velocity and magnetic fields, we have been able to identify several regions of magnetic field enhancement and their antecedent velocity structures.

We first visualize the velocity field using a high-frequency LIC texture. Next, the streamlines of the magnetic vector field are applied using the embossing technique introduced in section 6.2.2. The result is shown in figure 6.20.

Careful analysis of figure 6.20 reveals that there are regions of high magnetic field strength obviously correlated with particular velocity structures such as high-speed flows, flow compression, and shearing between flow structures. The underlying LIC texture depicts the velocity vector field in which the filter length of the LIC convolution kernel is varied to reflect the magnitude of the velocity vector. Purple (negative) and orange (positive) colors are used to represent the magnitude of the rate of change of the magnetic field. The embossed streamlines represent the magnetic vector field. The circled area in the center of the image is a region of significant magnetic field amplification where the magnetic field lines change orientation to be oriented perpendicular to the flow. The regions within the rectangles highlight an orthogonal correlation between flow structures and magnetic field

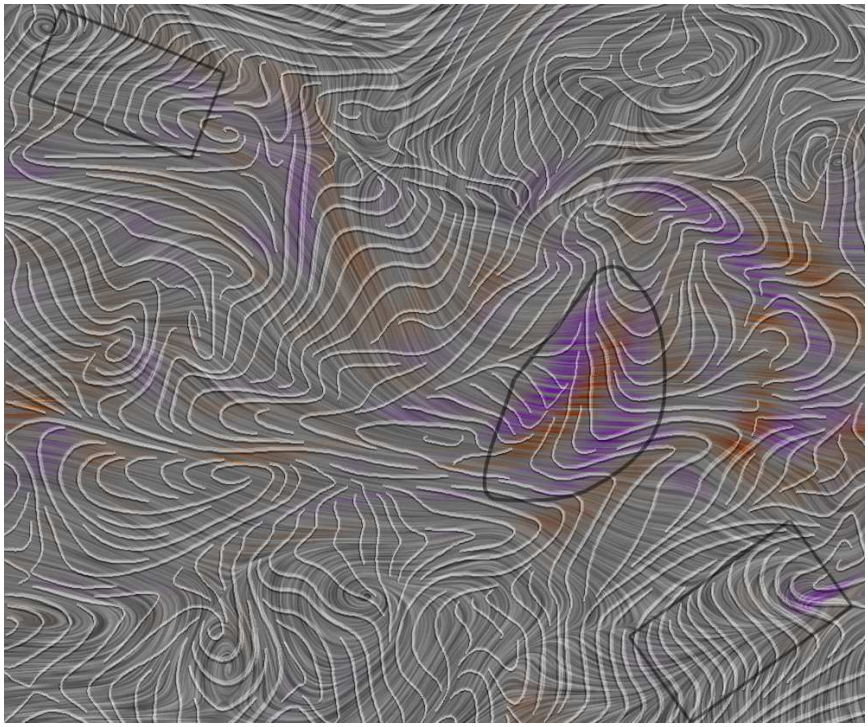

Figure 6.20: Visualization of multiple scalar and vector fields within a magnetohydrodynamic supersonic jet. The underlying LIC texture depicts the velocity vector field. The embossed streamlines represent the magnetic vector field.

lines. The line width of the embossed streamlines is also varied to reflect the magnitude of the magnetic field. The visualization of the vector and scalar fields in combination allows for advanced study of the correlation of variables within a magnetohydrodynamic supersonic jet.

Areas in figure 6.20 in which there exist positive regions of magnetic change indicate regions of magnetic field amplification. Regions of alignment of the magnetic vector field and the velocity field are a result of shear enhancement of the magnetic field. There are, however, additional magnetic enhancements with which the velocity field is not obviously causally connected. Moreover, there are many instances in which the magnetic field is unaligned or even orthogonal to the velocity field. This observation runs contrary to theoretical expectations, assuming shear is the dominant form of magnetic field amplification.

An understanding of the processes of magnetic field amplification and their relationship to flow velocity is important to astronomers since these magnetic structures are responsible for the observed radio emission characteristic of these systems. Through simulations and advanced visualization techniques, we expect to learn more about what observations of radio galaxies can tell us about the physics governing their evolution.

6.4.3 Direct Numerical Simulation of Turbulent Channel Flow

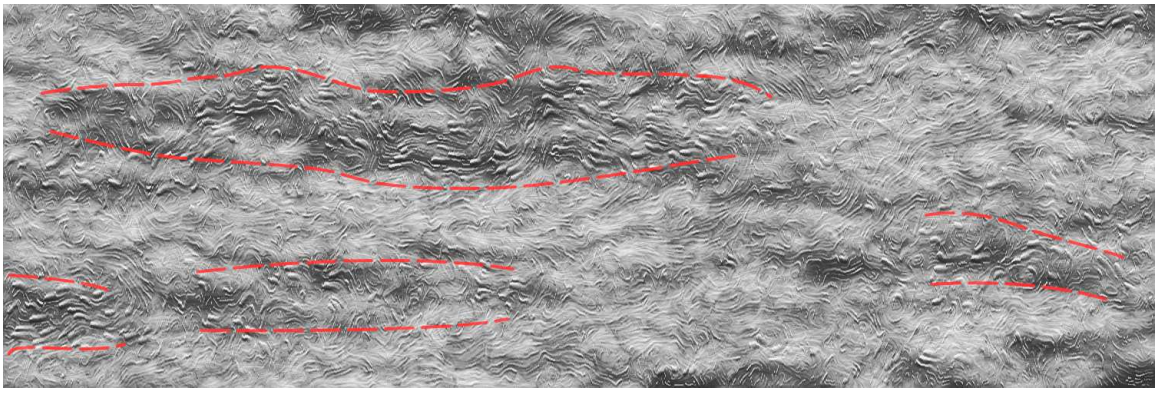

Figure 6.21: Visualization of two separate layers of the numerically generated wall-bounded turbulent flow. The red lines indicate some examples of elongated regions where the streamwise velocity is highly correlated in both layers.

The modeling of a turbulent channel flow is of interest to researchers in fluid dynamics because the simulation allows for the analysis of large, energetic features in the flow and the ability to correlate the flow structures in different layers within the 3D model. Developing a deeper understanding between the different regions within a channel flow results in a better understanding of the formation of vortical structures and the key-physical features that cause skin-friction drag. In this application, we consider wall-bounded turbulent flow data from a direct numerical simulation (DNS) of the full Navier-Stokes and continuity equations [11].

The discovery of the very long “superstructures” is particularly significant as it indicates an outer-layer scaled phenomena that may have influence all the way down to wall in

the inner layer. The DNS data is very useful for investigating this conjecture, and some results are shown in figure 6.21. Here two planes are shown simultaneously from the DNS dataset. The embossed glyphs data represents a plane in the log region ($z^+ = 150$) and the underlying texture data is in the viscous buffer zone ($z^+ = 15$) at the wall-normal location of maximum turbulent energy production. The luminance values for the texture and the glyph size show the instantaneous streamwise values for each plane. Dark, long, low-speed structures are visible in the viscous buffer zone, and careful examination indicates that the footprints of these structures do extend to the log region based on the presence of large glyphs in coincident locations. The very long "superstructures" described above are approximately outlined in red. These results indicate that near-wall regeneration mechanisms are *not independent* of the slow dynamics associated with structures on the order of the external dimension of the flow, as has always been previously believed.

6.5 Discussion

Creating an image that accurately displays the nature of two related vector fields in a single image is a challenging problem. It is necessary to be able to differentiate separate vector components while maintaining the perception of the pattern in each field with the ability to understand how the different fields interact with, relate to, and are unique from one another. Mining the knowledge base of techniques designed for single vector field visualizations and experimenting with ways of combining them yielded important findings and insight to assist the research of our specific applications.

Through combining images, we have found that using the screen overlay method works well to composite a high-contrast glyph or line-based representation with a dense, texture-based image. Embossing one of the representations also allows for the elements to perceptually group preferentially and segregate from the elements in the other

representation. We have also introduced several methods for the combination of two different fields of streamlines.

The applications presented in this chapter have been driven by the needs of fundamental fluids research problems: investigations by aerospace engineers into the physics of turbulence, and the computational modeling and simulation by astrophysicists of the behavior of supersonic jets in galactic clusters. The focus of the research has been the exploration of different visualization methods. The ultimate goal of this project is to create a user-friendly application that the application scientists can use on a routine basis for the analysis of coincident multiple vector fields.

Chapter 7

Conclusions

This dissertation provided a collection of techniques designed to effectively visualize multi-field flow data. In order to produce an image that successfully reflects multi-field data it is necessary not only to be accurate in the representation of individual distributions, but also to portray each specific component in a way that does not interfere with the accurate perception of the other components. The challenge of visualizing multiple scalar fields in combination with flow data has inspired many different techniques.

Many of the techniques presented in this dissertation involve using color, texture, or embossing in novel ways in order to accomplish the task of representing multi-field data. We have explored alternative methods to traditional approaches in the quest for creating a visual representation that leads to the accurate perception of complicated data sets. The overarching motivation for this work has been to provide scientists with tools to better understand the complicated interactions that occur between coincident variables.

7.1 Future Work

Future work in scientific visualization is strongly motivated by the desire for researches for deeper understanding of the key physical mechanisms that govern such problems as turbulent flow or astronomical structures. Building off of the contributions for visualization techniques made in this dissertation, there are many avenues for future research.

The problem of representing 3D vector components located on a 2D plane can not be easily solved with alterations of a 2D texture. The geometry of the vector field can be represented directly with geometric icons, such as glyphs or ellipsoids, as has traditionally been done with tensor visualizations [38, 43]. Once the geometry is defined, a texture depicting flow may be introduced by "mapping" the flow to the geometry in a manner similar to the techniques suggested by Laramee et al. [47] or van Wijk [81]. This would allow for the flexibility for multiple variables to be represented through geometry and the texture applied. Additional scalar variables may be introduced via sophisticated light reflection models (such as Bidirectional Reflection Distribution Functions). Using [90] as inspiration, the effectiveness of using gloss or haze to denote the presence or absence of scalar quantity could be analyzed. By altering surface characteristics and verifying the human perceptual system can accurately distinguish such differences, a wide range of variables may be visualized.

The techniques introduced in this dissertation mainly address the problem of multi-field flow data within a single time slice within the flow evolution. This is referred to as static flow. Unsteady flow, the problem of including the dimension of time, is an active area of research [47, 80, 81]. Developing visualization techniques that accurately reflect the multi-field components of unsteady flow is an exciting and challenging direction for future research.

The problem of dense, 2D flow visualization is close to being solved [46, 80]. However,

the problem of scaling the 2D solutions to 3D data is extremely challenging. The difficulty lies in creating images such that the human perceptual system can interpret three data dimensions containing multiple variables. The need for effective tools to interpret 3D data is significant as many scientists are investigating structures that cannot be fully appreciated with 2D slices. Consequently, 3D flow visualization is area of active research.

Current techniques in texture synthesis allow for the "steering" of the applied texture according to a vector field [21, 68, 69]. If a suite of textures could be constructed that would represent the different possibilities of vector directions *within* the texture, the two vector fields could contribute to the selection of the appropriate texture to apply and synthesize into a final image.

An area left largely unaddressed in this dissertation is the issue of interactivity. A fundamental understanding can be more easily obtained by interacting with an object, rather than simply observing it. The scientific visualization process could be enhanced by developing applications that are interactive via a virtual reality environment instead of simply creating images that are aesthetic and static [34, 35].

The interaction between visualization researchers and domain scientists in the process of developing the work presented has proven extremely beneficial to both parties. Through our collaboration we have advanced our ability to create effective visualizations and made important discoveries that further the development of important theories related to the applications. The application scientists played a critical role in defining the specific needs that the visualization techniques presented here were developed to address. In addition, they provided an objective assessment of the functionality of the methods, in terms of how well they meet their goals of gaining greater insight into their data.

Bibliography

[1] M. Ashikhmin. Synthesizing natural textures. *Proceedings of 2001 Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[2] L. D. Bergman, B. E. Rogowitz, and L. A. Treinish. A rule-based tool for assisting color map selection. *Proceedings of IEEE Visualization 95*, pages 118–125, 1995.

[3] D. Biskamp. *Magnetohydrodynamic Turbulence*. Cambridge University Press, 2003.

[4] B. Cabral and C. Leedom. Imaging vector fields using line integral convolution. *Proceedings of SIGGRAPH 93*, pages 263–269, 1993.

[5] R. Chevray and J. Mathieu. *Topics in Fluid Mechanics*. Cambridge University Press, 1993.

[6] R. Crawfis and M. Allison. Scientific visualization synthesizer. *Proceedings of IEEE Visualization 91*, pages 262–267, 1991.

[7] W. C. de Leeuw and R. van Liere. Comparing lic and spot noise. *Proceedings of IEEE Visualization 98*, pages 359–365, 1998.

[8] W. C. de Leeuw and J. J. van Wijk. Enhanced spot noise for vector field visualization. *Proceedings of IEEE Visualization 95*, pages 233–239, 1995.

[9] W. C. de Leeuw and J. J. van Wijk. Enhanced spot noise for vector field visualization. *Proceedings of IEEE Visualization 95*, pages 233–239, 1995.

[10] D. B. DeGraaff and J. K. Eaton. Reynolds number scaling of the flat-plate turbulent boundary layer. *Journal of Fluid Mechanics*, 422:319–346, 2000.

[11] J. C. del Álamo, J. Jiménez, P. Zandonade, and R. D. M oser. Scaling of the energy spectra of turbulent channels. *Journal of Fluid Mechanics*, 500:135–144, 2004.

[12] D. Dooley and M. F. Cohen. Automatic illustration of 3d geometric models: Surfaces. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 307–314, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[13] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001*, pages 341–346, 2001.

[14] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. *International Conference on Computer Vision*, pages 1033–1038, 1999.

[15] S. Eskinazi. *Principles of Fluid Mechanics*. Allyn and Bacon, 1962.

[16] B. Ganapathisubramani. *Investigation of turbulent boundary layer structure using stereoscopic particle image velocimetry*. PhD thesis, University of Minnesota, 2004.

[17] B. Ganapathisubramani, N. Hutchins, W. T. Hambleton, E. K. Longmire, and I. Marusic. Investigation of large-scale coherence in a turbulent boundary layer using two-point correlations. *Journal of Fluid Mechanics*, 524:57–80, 2005.

[18] B. Ganapathisubramani, E. K. Longmire, and I. Marusic. Characteristics of vortex packets in turbulent boundary layers. *Journal of Fluid Mechanics*, 478:35–46, 2003.

[19] B. Ganapathisubramani, E. K. Longmire, I. Marusic, and S. Pothos. Dual-plane piv technique to resolve complete velocity gradient tensor in a turbulent boundary layer. *Experiments in Fluids*, 39:222–231, 2005.

[20] J. Gomes and L. Velho. *Image Processing for Computer Graphics*. Springer, 1997.

[21] G. Gorla, V. Interrante, and G. Sapiro. Texture synthesis for 3d shape representation. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):512–524, 2003.

[22] C. D. Hansen and C. R. Johnson. *The Visualization Handbook*. Elsevier, 2005.

[23] C. Healey and J. Enns. Building perceptual textures to visualize multidimensional datasets. *Proceedings of IEEE Visualization 98*, pages 111–118, 1998.

[24] C. G. Healey. Choosing effective colours for data visualization. *Proceedings of IEEE Visualization 96*, pages 263–270, 1996.

[25] I. Hotz, L. Feng, B. Hamann, K. Joy, and B. Jeremic. Physically based methods for tensor field visualization. In *Proceedings of IEEE Visualization 2004*, pages 123–130, 2004.

[26] S. Hoyas and J. Jiménez. Scaling of the velocity fluctuations in turbulent channels up to $Re_\tau = 2003$. *Physics of Fluids*, 18:011702, 2006.

[27] N. Hutchins, W. T. Hambleton, and I. Marusic. Inclined cross-stream stereo PIV measurements in turbulent boundar y layers. *Journal of Fluid Mechanics*, 541:21–54, 2005.

[28] V. Interrante. Illustrating surface shape in volume data via pricipal direction-driven 3d line integral convolution. In *Proceedings of SIGGRAPH 97*, pages 109–116. ACM, ACM Press / ACM SIGGRAPH, 1997.

[29] V. Interrante and C. Grosch. Strategies for effectively visualizing 3d flow with volume lic. *Proceedings of IEEE Visualization 97*, pages 421–424, 1997.

[30] J. Jeong and F Hussain. On the identification of a vortex. *Journal of Fluid Mechanics*, 258:69–94, 1995.

[31] B. Jobard, G. Erlebacher, and M. Y. Hussaini. Lagrangian-eulerian advection of noise and dye textures for unsteady flow visualization. *Transactions on Visualization and Computer Graphics*, 8(3):211–222, 2002.

[32] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. *Proceedings of the 8th Eurographics Workshop on Visualization in Scientific Computing*, pages 43–55, 1997.

[33] B. Jobard and W. Lefer. The motion map: Efficient computation of steady flow animations. *Proceedings of IEEE Visualization 97*, pages 323–328, 1997.

[34] F. P. Brooks Jr. What's real about virtual reality? *IEEE Computer Graphics and Applications*, 19(6):16–27, 1999.

[35] D. Keefe, D. Karelitz, E. Vote, and D. H. Laidlaw. Artistic collaboration in designing VR visualizations. *IEEE Computer Graphics and Applications*, 25(2):18–23, March/April 2005.

[36] L. Khouas, C. Odet, and D. Friboulet. Vector field visualization using furlike texture. *Proceedings of Eurographics/IEEE TCVG Symposium on Data Visualization*, pages 35–44, 1999.

[37] K. C. Kim and R. J. Adrian. Very large-scale motion in the outer layer. *Physics of Fluids*, 11(2):417–422, 1999.

[38] G. Kindlmann. Superquadric tensor glyphs. *Proceedings of Eurographics/IEEE TCVG Symposium on Data Visualization*, pages 147–154, 2004.

[39] G. Kindlmann, G. Reinhard, and S. Creem. Face-based luminance matching for perceptual colormap generation. *Proceedings of IEEE Visualization 2002*, pages 299–306, 2002.

[40] R. Kirby, H. Marmanis, and D. H. Laidlaw. Visualizing multivalued data from 2d incompressible flows using concepts from painting. *Proceedings of IEEE Visualization 99*, pages 333–340, 1999.

[41] M.-H. Kiu and D. Banks. Multi-frequency noise for lic. *Proceedings of IEEE Visualization 96*, pages 121–126, 1996.

[42] R. V. Klassen and S. J. Harrington. Shadowed hedgehogs: A technique for visualizing 2d slices of 3d vector fields. *Proceedings of IEEE Visualization 91*, pages 148–162, 1991.

[43] D. H. Laidlaw, E. T. Ahrens, D. Kremers, M. J. Avalos, R. E. Jacobs, and C. Readhead. Visualizing diffusion tensor images of the mouse spinal cord. *Proceedings of IEEE Visualization 98*, pages 127–134, 1998.

[44] D. H. Laidlaw, M. Kirby, J. S. Davidson, T. Miller, Marco DaSilva, W. Warren, and M. Tarr. Quantitative comparative evaluation of 2d vector field visualization methods. *Proceedings of IEEE Visualization 2001*, pages 143–150, 2001.

[45] D. H. Laidlaw, M. Kirby, C. Jackson, J. S. Dav idson, T. Miller, M. DaSilva, W.Warren, and M. Tarr. Comparing 2D vector field visualization methods: A us er study. *Transactions on Visualization and Computer Graphics*, 11(1):59–70, January-February 2005.

[46] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum*, 23(2):203–221, 2004.

[47] R. S. Laramee, B. Jobard, and H. Hauser. Image space based visualization of unsteady flow on surfaces. *Proceedings of IEEE Visualization 2003*, pages 131–138, 2003.

[48] H. Levkowitz and G. T. Herman. Color scales for image data. *IEEE Computer Graphics and Applications*, 12(1):72–80, 1992.

[49] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. *Proceedings of IEEE Visualization 2005*, page 61, 2005.

[50] H. K. Moffatt. Magnetostatic equilibria and analogous euler flows of arbitrari ly complex topology. i - fundamentals. *Journal of Fluid Mechanics*, 159:359–378, 1985.

[51] S. M. O'Neill, I. L. Tregillis, T. W. Jones, and D. Ryu. Three-dimensional simulations of mhd jet propagation through uniform and stratified external environments. *The Astrophysical Journal*, 633(2):717–732, 2005.

[52] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, , and H. Doleisch. Feature extraction and visualization of flow fields. *Proceedings of Eurographics 2002*, pages 69–100, 2002.

[53] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, , and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

[54] C. Pozrikidis. *Introduction to Theroetical and Computational Fluid Dynamics*. Oxford University Press, 1997.

[55] B. E. Rogowitz and A. D. Kalvin. The "which blair project": A quick visual method for evaluating pereptual color maps. *Proceedings of IEEE Visualization 2001*, pages 183–188, 2001.

[56] B. E. Rogowitz and L. Treinish. The end of the rainbow. *IEEE Spectrum*, pages 52–59, December 1998.

[57] B. E. Rogowitz and L. A. Treinish. How not to lie with visualization. *Computers in Physics*, 3:268–274, May/June 1996.

[58] T. Saito and T. Takahashi. Comprehensible rendering of 3-d shapes. *Proceedings of SIGGRAPH 90*, pages 197–206, 1990.

[59] A. Sanna and B. Montrucchio. Adding a scalar value to 2d vector field visualization: the blic (bumped lic). *Eurographics 2000 Short Presentations Proceedings*, pages 119–124, 2000.

[60] A. Sanna, B. Montrucchio, P. Montuschi, and A. Sparavigna. Visualizing vector fields: the thick oriented stream-line algorithm (tosl). *Computers and Graphics*, 25(5):847–855, 2001.

[61] A. Sanna, B. Montrucchio, C. Zunino, and P. Montuschi. Enhanced vector field visualization by local contrast analysis. *Proceedings of Eurographics/IEEE TCVG Symposium on Data Visualization*, pages 35–41, 2002.

[62] G. Scheuermann, H. Burbach, and H. Hagen. Visualizing planar vector fields with normal component using line integral convolution. *Proceedings of IEEE Visualization 99*, pages 255–261, 1999.

[63] W. J. Schroeder and K. M. Martin. *The Visualization Handbook: Overview of Visualization*, chapter 1, pages 3–35. Elsevier, 2005.

[64] H.-W. Shen, C. R. Johnson, and K.-L. Ma. Building perceptual textures to visualize multidimensional datasets. *Proceedings of IEEE Visualization 96*, pages 63–70, 1996.

[65] H.-W. Shen, G. Li, and U. Bordoloi. Interative visualization of three-dimensional vector fields with flexible appearance control. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):434–445, 2004.

[66] D. Stalling and H.-C. Hege. Fast and resolution-independent line integral convolution. *Proceedings of SIGGRAPH 95*, pages 249–256, 1995.

[67] C. Stoll, S. Gumhold, and H. Seidel. visualization with stylized line primitives. *Proceedings of IEEE Visualization 2005*, pages 695–702, 2005.

[68] F. Taponecco and M. Alexa. Vector field visualization using markov random field texture synthesis. *Proceedings of Eurographics/IEEE TCVG Symposium on Data Visualization*, pages 195–202, 2003.

[69] F. Taponecco and M. Alexa. Steerable texture synthesis. *Proceedings of Eurographics 2004*, pages 57–60, 2004.

[70] I. L. Tregillis. *Simulated Relativistic Particle Transport and Nonthermal Emission in Three-Dimensional Magnetohydrodynamical Models of Radio Galaxies*. PhD thesis, University of Minnesota, 2002.

[71] I. L. Tregillis, T. W. Jones, and D. Ryu. Simulating electron transport and synchrotron emission in radio gala xies: Shock acceleration and synchrotron aging in three-dimensional flows. *The Astrophysical Journal*, 557(1):475–491, 2001.

[72] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[73] E. R. Tufte. *Envisioning Information*. Graphics Press, 1990.

[74] E. R. Tufte. *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press, 1997.

[75] G. Turk and D. Banks. Image-guided streamline placement. *Proceedings of SIGGRAPH 96*, pages 453–460, 1996.

[76] T. Urness, V. Interrante, E. Longmire, I. Marusic, and B. Ganapathisubramani. Effectively visualizing multi-valued flow data using color and texture. *Proceedings of IEEE Visualization 2003*, pages 115–121, 2003.

[77] T. Urness, V. Interrante, E. Longmire, I. Marusic, and B. Ganapathisubramani. Techniques for visualizing multi-valued flow data. *Proceedings Eurographics/IEEE TCVG Symposium on Data Visualization*, pages 165–172, 2004.

[78] T. Urness, V. Interrante, E. Longmire, I. Marusic, S. O'Neill, and T. W. Jones. Strategies for the visualization of multiple 2d vector fields. *IEEE Computer Graphics and Applications (to appear)*, 2006.

[79] J. J. van Wijk. Spot noise — texture synthesis for data visualization. *Proceedings of SIGGRAPH 91*, pages 309–318, 1991.

[80] J. J. van Wijk. Image based flow visualization. *ACM Transactions on Graphics*, 21(3):745–754, 2002.

[81] J. J. van Wijk. Image based flow visualization for curved surfaces. *Proceedings of IEEE Visualization 2003*, pages 123–130, 2003.

[82] J. J. van Wijk. The value of visualization. *Proceedings of IEEE Visualization 2005*, pages 79–86, 2005.

[83] V. Verma, D. Kao, and A. Pang. Plic: Bridging the gap between streamlines and lic. *Proc. Symposium on Data Visualization 1999*, pages 341–348, 1999.

[84] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufman, 2000.

[85] C. Ware and W. Knight. Using visual texture for information display. *ACM Transactions on Graphics*, 14(1):3–20, January 1995.

[86] R. Wegenkittl and E. Gröller. Oriented line integral convolution for vector field visualization via the internet. *Proceedings of IEEE Visualization 97*, pages 309–316, 1997.

[87] R. Wegenkittl, E. Gröller, and W. Purgathofer. Animation flowfields: Rendering of oriented line integral convolution. *Proceedings of IEEE Computer Animation 97*, pages 15–21, 1997.

[88] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. *Proceedings of SIGGRAPH 2000*, pages 479–488, 2000.

[89] C. Weigle, W. Emigh, G. Liu, R. Taylor, J. Enns, and C. Healey. Oriented sliver textures: A technique for local value estimation of multiple scalar fields. *Graphics Interface*, pages 163–170, 2000.

[90] H. B. Westlund and G. W. Meyer. Applying appearance standards to light reflection models. *Proceedings of SIGGRAPH 2001*, pages 501–510, 2001.

[91] P. C. Wong, H. Foote, D. L. Kao, L. R. Leung, and J. Thomas. Multivariate visualization with data fusion. *Information Visualization*, 1(3-4):182–193, 2002.

[92] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3d streamlines. *Proceedings of IEEE Visualization 2005*, page 60, 2005.

[93] J. Zhou, R. J. Adrian, R. J. Balachandar, and T. M. Kenda ll. Mechanisms for generating coherent packets of hairpin vortices in channel flow. *Journal of Fluid Mechanics*, 387:353–396, 1999.