Michigan Technological University

Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's Reports

2015

# ENABLING TECHNIQUES FOR EXPRESSIVE FLOW FIELD VISUALIZATION AND EXPLORATION

Jun Tao
*Michigan Technological University*, junt@mtu.edu

Copyright 2015 Jun Tao

## Recommended Citation

Tao, Jun, "ENABLING TECHNIQUES FOR EXPRESSIVE FLOW FIELD VISUALIZATION AND EXPLORATION", Open Access Dissertation, Michigan Technological University, 2015.
https://digitalcommons.mtu.edu/etdr/49

Follow this and additional works at: https://digitalcommons.mtu.edu/etdr

 Part of the Graphics and Human Computer Interfaces Commons

# ENABLING TECHNIQUES FOR EXPRESSIVE FLOW FIELD VISUALIZATION AND EXPLORATION

By

Jun Tao

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Computer Science

MICHIGAN TECHNOLOGICAL UNIVERSITY

2015

© 2015 Jun Tao

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Computer Science.

Department of Computer Science

| | |
|---|---|
| Dissertation Co-advisor: | *Dr. Ching-Kuang Shene* |
| Dissertation Co-advisor: | *Dr. Chaoli Wang* |
| Committee Member: | *Dr. Laura E. Brown* |
| Committee Member: | *Dr. Raymond A. Shaw* |
| Department Chair: | *Dr. Min Song* |

# Contents

# List of Figures

# List of Tables

# Preface

Projects introduced in this dissertation are collaborated with several professors, graduate students and other researchers. For the work in Chapter 3, the algorithm was designed by all the coauthors, Jun Tao, Jun Ma, Professor Chaoli Wang and Professor Ching-Kuang Shene. Jun Tao implemented the GPU component, conducted the user study and wrote most of the paper. Jun Ma implemented the CPU component and generate the results. For the work in Chapter 4, Jun Tao designed the algorithm and finished the implementation under the supervision of Professor Chaoli Wang and Professor Ching-Kuang Shene. Professor Seung Hyun Kim evaluated the approached and provided valuable feedbacks. For the work in Chapter 5, Jun Tao designed and implemented the approach under the supervision of Professor Chaoli Wang and Professor Ching-Kuang Shene. Professor Raymond A. Shaw participated in the empirical expert evaluation and contributed a section of the paper. For the work in Chapter 6, Jun Tao designed the algorithm, implemented the host component and conducted the user study. Xiaoke Huang implemented the web component. Dr. Feng Qiu was involved in the discussions during the design and implementation and provided valuable inputs. This work was finished under the supervision of Professor Chaoli Wang, Professor Ching-Kuang Shene, Professor Ye Zhao and Daphne Yu. Professor Jingfeng Jiang was involved in the empirical expert evaluation and provided valuable suggestions that lead to further developments. For the tools described in Chapter

7, Jun Tao designed and implemented the tools under the supervisions of Professor Ching-Kuang Shene and Professor Chaoli Wang.

# Acknowledgments

This dissertation would never be possible without the guidance from my advisors, collaboration of my coauthors, and support from my friends and family.

I would like to express my deepest gratitude to my advisor Dr. Ching-Kuang Shene, who helped me consolidate my mathematical background and allowed me great freedom to explore on my own. Dr. Shene always encouraged me to improve my abstract thinking ability and taught me to organize and express ideas in an effective way. The discussions with him are critical to the completion of this dissertation.

I owe the same gratitude to my co-advisor, Dr. Chaoli Wang as well. Dr. Wang was actively involved in every project presented in this dissertation. He taught me how to convert a thought into a publication and how to manage time during this process. He also helped me overcome many difficult situations in my research.

I would like to thank Dr. Raymond A. Shaw and Dr. Laura E. Brown to serve in my dissertation committee and provide me many helpful comments to improve this dissertation.

I am thankful to Daphne Yu, Dr. Wei Li and Dr. Feng Qiu, my supervisors when I was an intern at Siemens Corporate Technology. They helped me improve my

knowledge in medical visualizations and developing large projects. I would like to thank my collaborator Xiaoke Huang at Siemens as well. Our joint work would never be finished without his effort.

I am also grateful to my labmates Yifei Li, Jun Ma, Man Wang and Yi Gu for accompanying me in the lab. The conversations with them always inspired me to generate new ideas.

I would also like to acknowledge my friends Dr. Kai Ma, Dr. Xiao Bian and Dr. Jinfei Jia for broadening my views in machine learning.

I am also indebted to all members of the WONTON group: Weilue He, Fan Yang, Pei Hou, Boyi Hao, Huahua Shi and Boming Wu. They really helped me take care of my personal life, and their accompany provided the most unforgettable moments in the past years.

Finally, I would like to thank my parents Gang Tao and Yueyun Chen, and my friend Yanshan Li, for supporting me spiritually through all these years.

# Abstract

Flow visualization plays an important role in many scientific and engineering disciplines such as climate modeling, turbulent combustion, and automobile design. The most common method for flow visualization is to display integral flow lines such as streamlines computed from particle tracing. Effective streamline visualization should capture flow patterns and display them with appropriate density, so that critical flow information can be visually acquired.

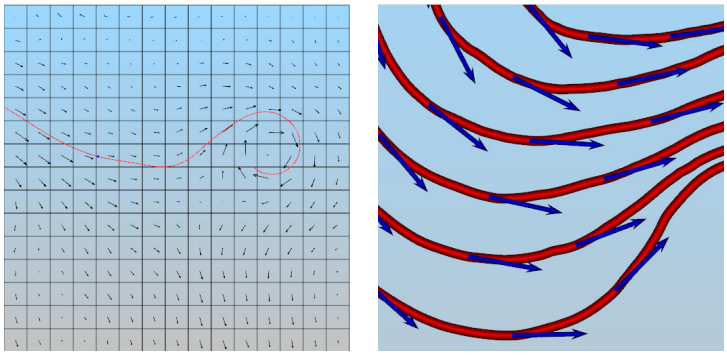
In this dissertation, we present several approaches that facilitate expressive flow field visualization and exploration. First, we design a unified information-theoretic framework to model streamline selection and viewpoint selection as symmetric problems. Two interrelated information channels are constructed between a pool of candidate streamlines and a set of sample viewpoints. Based on these information channels, we define streamline information and viewpoint information to select best streamlines and viewpoints, respectively. Second, we present a focus+context framework to magnify small features and reduce occlusion around them while compacting the context region in a full view. This framework parititions the volume into blocks and deforms them to guide streamline repositioning. The desired deformation is formulated into energy terms and achieved by minimizing the energy function. Third, measuring the similarity of integral curves is fundamental to many tasks such as feature detection,

pattern querying, streamline clustering and hierarchical exploration. We introduce FlowString that extracts shape invariant features from streamlines to form an alphabet of characters, and encodes each streamline into a string. The similarity of two streamline segments then becomes a specially designed edit distance between two strings. Leveraging the suffix tree, FlowString provides a string-based method for exploratory streamline analysis and visualization. A universal alphabet is learned from multiple data sets to capture basic flow patterns that exist in a variety of flow fields. This allows easy comparison and efficient query across data sets. Fourth, for exploration of vascular data sets, which contain a series of vector fields together with multiple scalar fields, we design a web-based approach for users to investigate the relationship among different properties guided by histograms. The vessel structure is mapped from the 3D volume space to a 2D graph, which allow more efficient interaction and effective visualization on websites. A segmentation scheme is proposed to divide the vessel structure based on a user specified property to further explore the distribution of that property over space.

# Chapter 1

# Introduction

Flows widely exist in our world and influence human life in many ways. The behaviors of flows are studied in many scientific areas, such as aerodynamics of aircrafts and vehicles, air flow in meteorology, ocean flow in oceanography, and blood flow in biomedical engineering, etc. Observing the flow pattern provides a comprehensive way to study the flow behaviors. However, most flows are transparent and their patterns are not directly visible. Traditionally, experimental methods were used to make the flow visible, e.g., spilling ink or adding dyes into liquid. With the extensive use of computer models in modern engineering research, huge amount of flow data are simulated through computational fluid dynamics (CFD). Flow visualization plays an important role to make this kind of data visible, so that the flow patterns and behaviors can be studied in a comprehensive way. On one hand, flow visualization provides



(a) (b)

**Figure 1.1:** 2D visualization of (a) a steady vector field, and (b) streamline segments and the associated vectors on their control points.

a method for scientists to verify their computation models with observations. On the other hand, it allows researchers to observe correlations between certain flow patterns and the possible outcomes. This dissertation focuses on the development of visualization techniques that provides clear observations under different view directions and interacts with users to meet different needs.

## 1.1 Background

This dissertation focuses on flows defined on vector fields. A *vector field* on $U \subseteq \mathbb{E}^n$ is a mapping $V : U \to \mathbb{E}^n$ from an open set $U \subseteq \mathbb{E}^n$ to $\mathbb{E}^n$, where $\mathbb{E}^n$ is an Euclidean metric space. This mapping $V$ assigns an $n$-dimensional vector $V(v) \in \mathbb{E}^n$ to each point $v \in U \subseteq \mathbb{E}^n$. The domain $U$ is commonly represented by a grid in CFD, and each cell of this grid is associated with a vector that indicates both direction and

magnitude of the flow at the center of that cell. Vectors at positions rather than cell centers are linearly interpolated. Figure 1.1 (a) shows an example of a 2D vector field.

A *flow* on an open set $X$ is a mapping $\phi : X \times \mathbb{R} \to X$ such that for all $x \in X$ and all $s, t \in \mathbb{R}$, we have the following:

$$\begin{aligned} \phi(x, 0) &= x \\ \phi(\phi(x, t), s) &= \phi(x, s + t) \end{aligned} \tag{1.1}$$

Thus, the flow $\phi$ sends a point $x \in X$ to another point $\phi(x, t) \in X$ for each $t \in \mathbb{R}$. Given a point $x \in X$, the set of $\phi(x, t)$ for all $t \in \mathbb{R}$ is referred to as the *orbit* of $x$ under the map $\phi$. It is the *trajectory* of the movement of $x$ over time. A *steady* flow is a flow in which the properties assigned to any point are independent of the time parameter. Because we only focus on the velocity at each point, a flow is steady if the vector assigned to any point does not vary over $t$; otherwise, a flow is an *unsteady* flow.

A $C^0$ (continuous) vector field can produce a flow $\phi$ so that the given vector field is the induced velocity field of $\phi$. More precisely, the vector assigned to any point is tangent to the orbit of a flow $\phi$ through that point. Intuitively, this means that given a $C^0$ vector field $V : U \to \mathbb{E}^n$, the flow is fully determined in the sense that the orbit

of every point $X_0 \in U$ can be found. The orbit with initial position $X_0$ is simply a curve $C : \mathbb{R} \to U$ (i.e., $C(t) \in U$ for every $t \in \mathbb{R}$) such that $C(0) = X_0$. The vector assigned to a point of $C(t)$ is $V(C(t))$. $V(C(t))$ is supposed to be $\dot{C}(t)$, where $\dot{C}(t)$ is the tangent vector at point $C(t)$. Therefore, we have the following equations:

$$\begin{aligned} V(C(t)) &= \dot{C}(t) \\ C(0) &= X_0 \end{aligned} \tag{1.2}$$

The above is a first-order ordinary differential equation and has a unique solution $C(t)$ via integration. An orbit obtained in this way is referred to as an *integral curve*. In this dissertation, we use the term "flow field" and "vector field" interchangeably, because our flows come from vector fields.

Scientific visualization provides a transformation from vector fields into displayable forms such as images, where phenomenons can be observed by human. The most common way to visualize the flow using the integral curves as described before, which is usually referred to as *field-lines* in flow visualization literature. The field-lines are called *streamlines* (for a steady flow) or *pathlines* (for an unsteady flow). An example of streamlines is shown in Figure 1.1 (b), where the streamlines are drawn in red and the vectors are drawn in blue. Unlike Figure 1.1 (a), which shows the flow direction of every grid point, only the flow directions on control points of streamlines are immediately available in (b). But it is difficult to visually obtain the flow pattern at

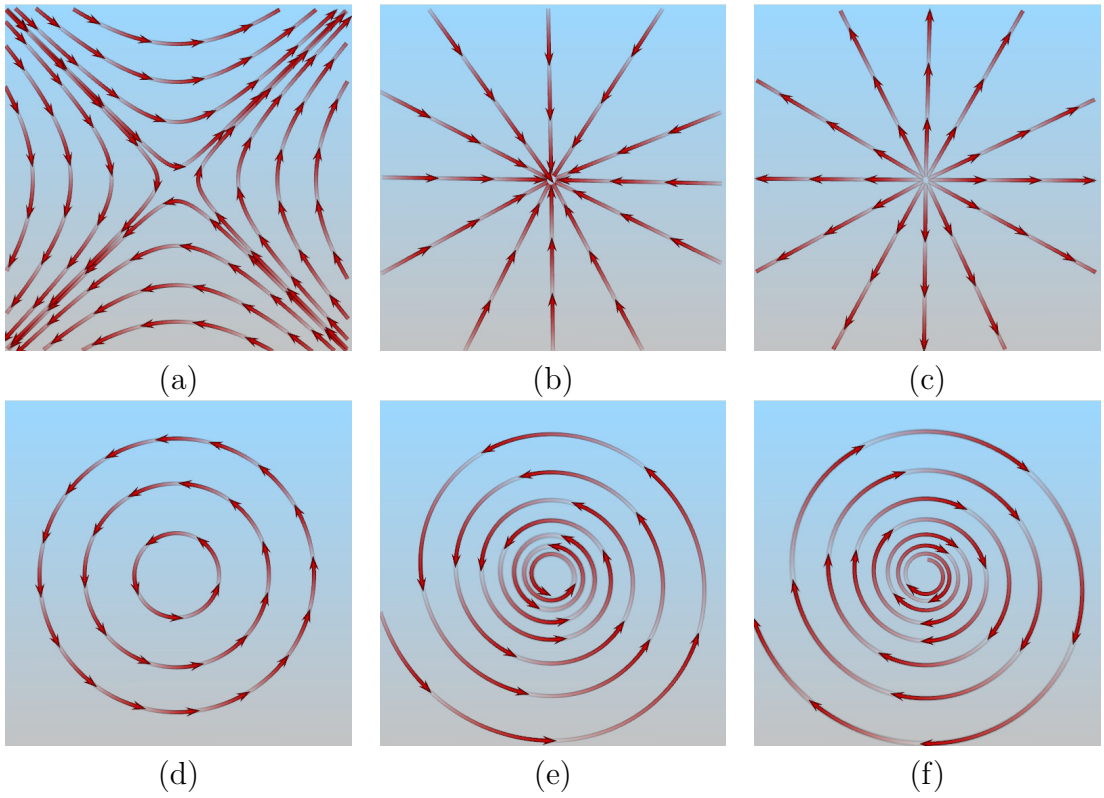
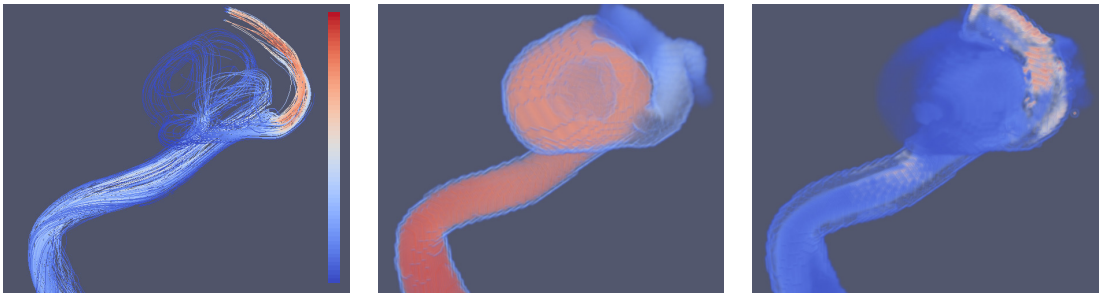



**Figure 1.2:** Six types of 2D critical points. (a) saddle, (b) sink, (c) source, (d) center, (e) attracting spiral, and (f) repelling spiral. (© 2013 ASEE. Reprinted by permission.)

larger scale from an image displaying an arrow for every grid point, especially for a 3D data set where there are much more grid points creating occlusions during projection. On the contrary, using streamlines, the flow pattern of a flow field is revealed in a continuous form, and it is easier for researchers to build connections between different regions from perception. In addition, with the appropriate streamlines, the flow direction at a point can still be implied from the neighboring streamlines, even if that point is not on any streamline.

A flow field may exhibit several special types of flow pattern, which are characterized



(a) (b) (c)

**Figure 1.3:** A vascular data set with an aneurysm. (a) Streamlines traced in the vessel. (b) The pressure field in the vessel. (c) The WSS field in the vessel.

by *critical points*. A critical point is a singularity in the vector field where the velocity vanishes. Figure 1.2 [1] shows six types of 2D critical points. Figure 1.2 (a) demonstrates a saddle, where vectors repel each other at a point. Figure 1.2 (b) presents a sink, where vectors converge into a point. Figure 1.2 (c) shows a source, where vectors emanate from a point. In Figure 1.2 (d), (e) and (f), three types of critical point where vectors revolve along a point are shown. They are a center, an attracting spiral and a repelling spiral. Locating and classifying these critical points in vector fields have important practical meaning in many disciplines.

In addition, some scalar fields may be simulated to describe properties associated with the vector fields. A scalar field $S : U \to \mathbb{R}$ maps a point to a scalar value, which indicates the value of some property at that point. Examples of these properties include carbon dioxide concentration in meteorological data sets and blood pressure in vascular data sets. The flow pattern and the distribution of these scalar properties

[1]The images contained in this figure were previously published in *Proceedings of American Society for Engineering Education Annual Conference* [122].

in these data sets are not immediately visible to researchers, not to mention their relationships. Figure 1.3 shows a vascular data set that consists of multiple fields, including a vector field to describe the blood flow and the corresponding scalar fields of pressure and wall shear stress (WSS). The vessel structure contains an aneurysm, whose rupture could be life-threatening. It is believed that the cause of rupture of an aneurysm is related to multiple factors, which includes the distribution of WSS and pressure and the pattern of spirals in the aneurysm [94, 111]. Therefore, it is critically important to study the relations among these properties, and possibly with the rupture outcomes. In addition, studying the blood flow in a vessel has practical meaning to the treatment of the aneurysm as well. It can help medical experts to find the best locations to release drugs so that they can stay longer in the aneurysm and closer to the vessel wall.

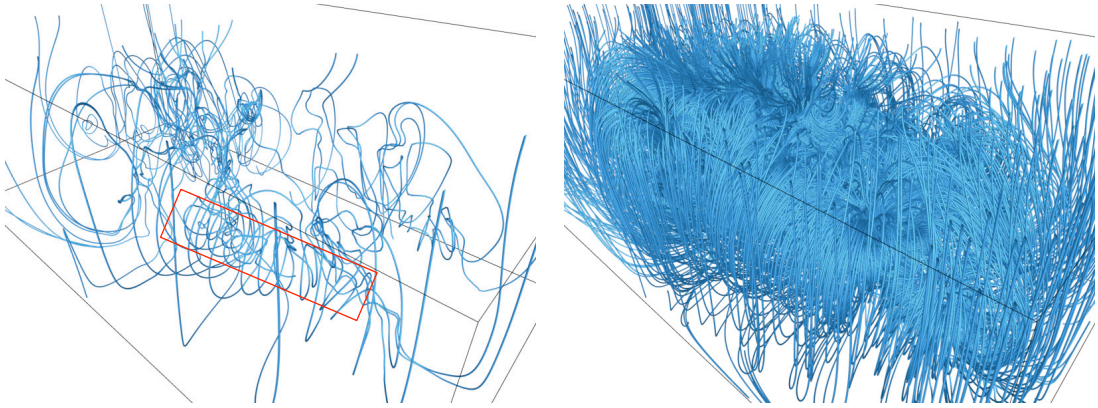
## 1.2 Challenges

The principals for flow visualization are similar to general scientific visualization, including informative, and visually pleasing, etc. In this dissertation, we focus on generating an informative flow visualization that helps the exploration of flow fields. The meaning of an informative flow visualization is multifold. On one hand, being informative indicates flow visualization should preserve as much information in the

original data set as possible. In this sense, using more streamlines seems to be preferred, as the flow directions can only be obtained through streamlines. On the other hand, the information contained in the visualization result should be easily obtained. In contrast to the previous criterion, this normally requires less streamline for clarity. Figure 1.4 demonstrates the impact of streamline density on the information conveyed. In (a), twenty streamlines are displayed, and the flow directions around these streamlines can be perceptually interpreted. However, most regions are left empty where the flow patterns are shown. In (b), although using 200 streamlines fills the entire volume, most regions are occluded by the streamlines in the front, and their flow patterns are still unknown to users.

Moreover, the data sets nowadays can be large and complicated. Therefore, displaying all features clearly becomes impossible. Many recent research focus on discovering and emphasizing features or structures of data sets. These includes the techniques that automatically identify regions of interests and emphasize those regions in the visualization result, and approaches that present the underlying structure of a data set in an abstract way. Finally, visualization somewhat lies between science and art. The ability to generate eye-catching rendered results is important for a visualization approach as well.

There are still many remaining challenges in meeting these requirements. First, it is challenging to maintain appropriate densities for all regions. The criteria of preserving

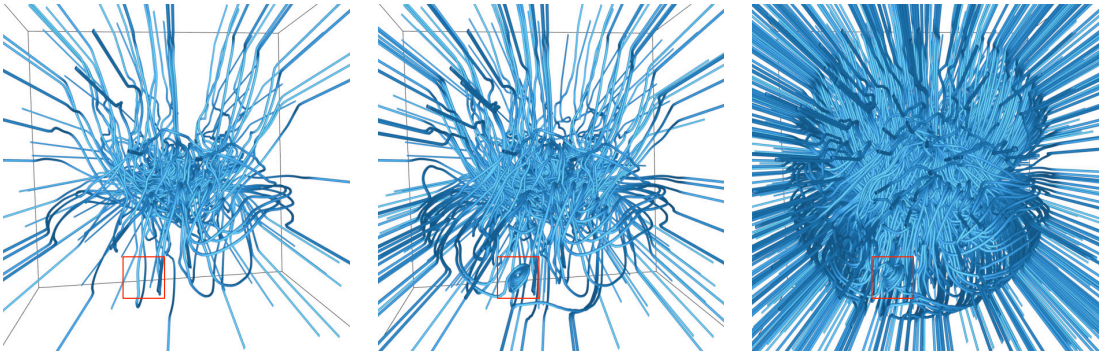


(a) (b)

**Figure 1.4:** Randomly traced streamlines using the crayfish data set. (a) 20 streamlines. (b) 200 streamlines. The feature region highlighted in red is sparse in (a) but occluded in (b).

information and clarity are both closely related to the density of streamlines being displayed and contradictory in some aspect. However, the balance of these factors is not always achievable. The densities in different regions are interrelated, since streamlines usually pass multiple regions. In Figure 1.4 (a), the flow behavior in the inner region is more complicated and therefore more difficult to predict if not captured by streamlines, as highlighted in the red rectangle. Most of the inner region is still empty and more streamlines should be added. However, most of the streamlines pass both the inner and outer regions, and it is difficult to increase the density in inner region and still maintain the density in outer region. For example, after adding more streamlines in Figure 1.4 (b), the inner region is completely occluded by the outer region.

Second, a feature can be too small to be captured and observed. In Figure 1.5 (b), a



(a) (b) (c)

**Figure 1.5:** Randomly traced streamlines using the supernova data set. (a) 100 streamlines. (b) 200 streamlines. (c) 1000 streamlines. The small spiral highlighted in the red square can only be observed clearly in (b).

small spiral is highlighted in the red rectangle, which is small and difficult to observe. However, with less streamlines, this feature is not captured by the streamlines, as shown in (a). With more streamlines, this feature may be easily overlooked by users, as shown in (c).

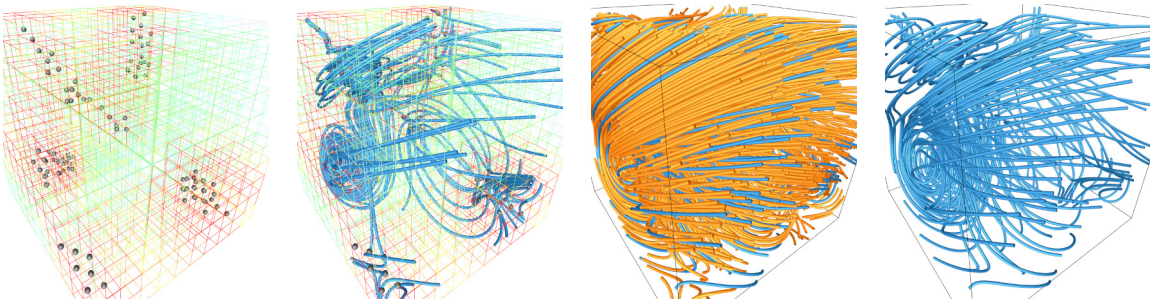
Third, the projection from 3D streamlines to a 2D plane depends on the view. By changing the viewing direction, the local densities of different regions on the projection plane may change accordingly. Hence, the most appropriate density may not exist for all viewing directions/positions. Moreover, the information conveyed by a streamline is dependent on the view direction as well. For example, a planar curve will downgrade to a straight line when the viewing direction lies in the plane that contains the curve. In this case, the information of that curve is greatly lost in final rendered results.

Fourth, there is no universal rule to identify regions of interest or useful information to explore. The goals of flow visualization are commonly discipline/application-dependent. For example, some medical experts may be interested in spirals as they are believed to be related to ruptures of vessels; other medical experts researching drug delivery may be interested in the amount of blood that enters a certain region; and mechanical engineers may be interested in the combustion flows in engines, etc. In recent years, many approaches have been developed for specific applications, while some general approaches can be tuned to fulfill certain requirements.

In conclusion, an effective flow visualization should clearly convey the flow pattern in the flow field. Maintaining appropriate streamline density for a clear observation and allowing users to easily identify features are major tasks. Major effort are placed in the following directions: choosing good seed points to trace streamlines; selecting good streamlines from a large and usually randomly generated pool; measuring streamline similarities to simplified the visualization results; and introducing interactions for users to explore the flow field according to their own needs. Details will be provided in the next section.

## 1.3 Methodology

Our approaches target at the previously mentioned challenges in several aspects. First, we do not consider the density of streamlines explicitly. Instead, the amount of information conveyed by a pool of streamlines is evaluated. This naturally provides streamlines with appropriate overall density, as the information does not increase when a streamline is added to a region that is already dense enough. Second, we build connections between streamline selection and viewpoint selection. We evaluate the information conveyed by a streamline under a certain viewpoint. In this way, not only do we consider the flow pattern represented by a streamline, but also take what users can perceive into account. Third, we operate on streamline segments instead of streamlines. This provides finer level of control, as it breaks the interrelation among regions, so that they can be manipulated relatively independently. Forth, user interactions are highly involved in our approaches. Our approaches are published [103, 104, 105, 106] and covered in details in this dissertation. In this section, we present a high-level description of these approaches and the related works.



(a) (b) (c) (d)

**Figure 1.6:** Generating streamlines using seed placement and streamline selection methods. (a) The grid shows the importance values of each point, and the seeds are placed according to the importance values. (b) Streamlines are traced from the seeds shown in (a). (c) Randomly traced 900 streamlines with 100 of them selected. (d) The selected streamlines in (c).

### 1.3.1 Streamline and Viewpoint Selection

Traditional streamline visualization consists of two major categories: *seed placement* and *streamline selection*. Seed placement aims at carefully placing seeds in the domain to generate streamlines that capture the essence of the flow field [45, 60, 78, 109, 116, 130, 133]. A seed is an initial position to trace a streamline. Ideally, an algorithm for seed placement should generate visually pleasing and technically illustrative images. These methods either target at evenly spaced streamlines for visual consistence, or place more seeds around feature regions so that more streamlines will pass those regions, which not only ensure the features are captured but also highlight those regions with higher streamline density. Figure 1.6 (a) demonstrates a common procedure of seed placement. An importance value is evaluated for each voxel in the volume, and the seeds are placed at the voxels with highest importance

values. In this example, the importance value is computed using the entropy of flow directions, which indicates how complicate the flow is in a small neighborhood. Then streamlines are traced from the seeds, as shown in (b).

An alternative to seed placement is streamline selection. Streamline selection aims at carefully selecting streamlines from a large streamline pool for effective display [13, 57, 70, 74]. Figure 1.6 (c) shows a pool of 900 streamlines that are randomly generated in the flow field, from which 100 are selected and highlighted in red. The selection is performed based on the *streamline information* described in Section 3.4. The final rendered result is displayed in Figure 1.6 (d) with only the selected streamlines.

Compared to selecting seeds, selecting streamlines is directly related to the streamlines, which are the final visualization results. Streamline selection can measure the information contained by a streamline, or even a pool of streamlines, while seed placement usually evaluates a seeding position locally, where the global information of a streamline is missing. On the other hand, in Figure 1.6 (b), seed placement produces streamlines that concentrate on the important regions and their patterns are similar as well. On the other hand, streamline selection has a better coverage of information contained in the vector field. Seed placement was commonly used in the past as it required less computation power. With the rapid advances of general-purpose computing on GPUs, it is quite affordable nowadays to generate a large pool of streamlines. As such, streamline selection has become a promising alternative to

seed placement and has received increasing attention. Although the task is shifted from selecting *good seeds* to selecting *good streamlines*, the goal remains the same: we want to produce a set of streamlines that are not only descriptive as individuals, but also informative as a group: i.e., highlighting flow features and patterns while reducing visual clutter.

Our streamline selection improves the existing methods in two aspects: first, it considers the information contained in a streamline group and the information conveyed by a single streamline. Therefore, it provides better global pictures. Second, it considers streamline selection to be closely related to viewpoint selection. Therefore, it considers not only the shape of a 3D streamline, but also the information conveyed by its 2D projection on the screen. In Chapter 3, we will present our approach to formulate streamline selection and viewpoint selection into a unified information-theoretic framework and solve them simultaneously. This approach not only selects the most informative streamlines to depict the vector field, but also suggests the best views to observe the selected streamlines.

### 1.3.2 Focus+Context Flow Visualization

Streamline selection has its limitation as well. A streamline may pass multiple regions with various degrees of importance. Ideally, it is desirable to achieve different densities

for these regions according to their own importance values. However, streamline selection either selects the entire streamline or completely ignores it. Both changes the densities of all regions passed by this streamline in the same way. Figure 1.4 (b) presents a good example in this case. Most streamlines pass both the inner feature region where the flow is complicated, and the outer context region where the flow is simple and aligns along the boundary. Therefore, an effective visualization should produce results that reduce the density of the outer region, while maintaining appropriate density for the inner region. But we find this difficult to achieve since removing streamlines reduces the density of the inner region as well, as shown in Figure 1.4 (a). Moreover, some features are small in size, as shown in Figure 1.5. They may be easily overlooked even if they are captured by some streamlines, since only short segments of streamlines cover these features. In this case, it will help to enlarge the regions containing this kind of segments. These all raise the need for a finner level of control over the streamlines.

To this end, it will be beneficial to adopt a focus+context visualization to magnify small features so that they could be easier to observe, and stretch the regions in front of feature regions so that the streamlines blocking the features get sparser. To achieve these two goals, we introduce our focus+context flow visualization approach to maintain the shape of flow features and utilize less important regions to absorb the distortion. Unlike the streamline selection approaches, this deformation framework provides us with higher flexibility to adjust the densities of streamlines in different

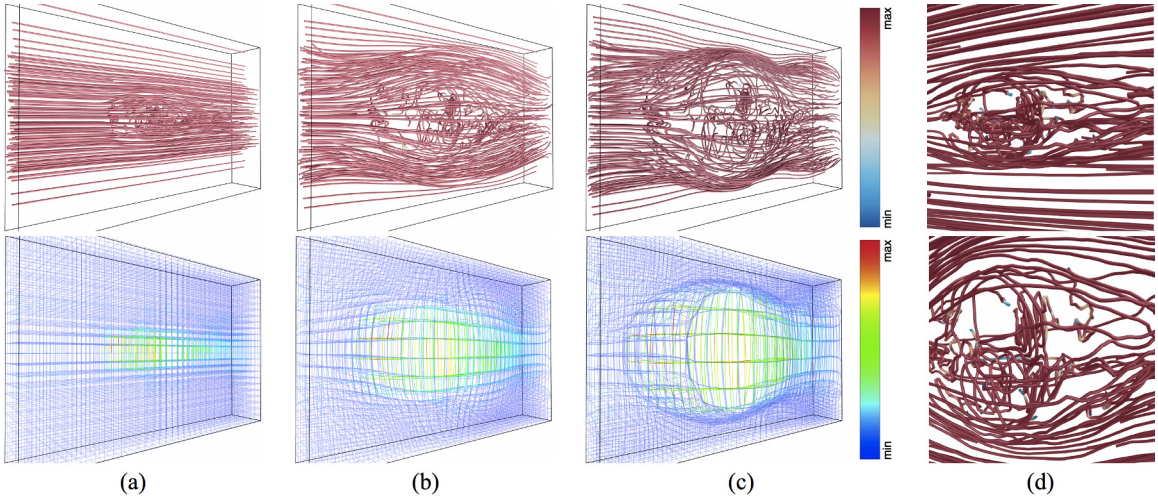



**Figure 1.7:** Deforming the streamline and corresponding grids with our focus+context visualization. The streamlines are shown in the first row, and the grids are shown in the second row. (a) displays the original streamlines and grid. (b) displays the visualization with a moderate level of deformation. (c) displays the visualization with a larger deformation. The first and second row in (d) show the enlarged feature regions in (b) and (c), respectively. (© 2014 IEEE. Reprinted by permission.)

regions relatively independently. In Figure 1.7[2], the interesting features only occupy a small region at the center of the volume. It cannot be observed clearly, as this region is both small and occluded by other streamlines, as shown in (a). We measure the importance at each voxel using the entropy of flow directions. The high entropy regions colored in red and yellow correspond to the feature region. In (b) and (c), we consider the feature region under focus and use different parameters to deform the volume, so that the feature region expands and other regions shrink to absorb the distortion. In column (d), the feature region is enlarged to show the details. We observe that the flow patterns is more spreading out and clearer in the deformed

[2]The images contained in this figure were previously published in *IEEE Transactions on Visualization and Computer Graphics 2014* [105].

volume. In Chapter 4, we will provide the details of this approach.

### 1.3.3 Streamline Similarity Measure

In flow visualization, measuring the similarity between discrete vectors or the similarity between integral curves is of vital importance for many tasks such as data partitioning, seed placement, field-line clustering and hierarchical exploration. This need has become increasingly urgent, but fulfilling it has become more challenging as the size and complexity of flow field data continue to grow dramatically over the years. Early research in this direction focused on vector field similarity and hierarchical classification [39, 108]. This focus has shifted to similarity measure of integral curves in recent years. Many of the similarity measures designed were targeted on fiber bundle clustering in diffusion tensor imaging (DTI). In this context, spatial proximity is the major criterion for clustering these DTI fiber tracts. Fiber bundles can be formed to characterize different bunches of tracts that share similar trajectories.

In computational fluid dynamics, integral curves such as streamlines or pathlines traced from flow field data are more complex than DTI fiber tracts. Many CFD simulations produce flow field data featuring regular or turbulent patterns at various locations, orientations and sizes. Clearly, only considering the spatial proximity alone is not able to capture intrinsic similarity among integral curves traced over the field.

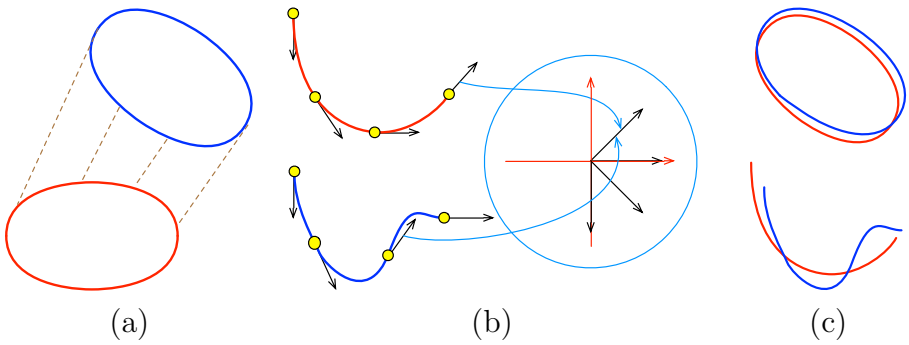



**Figure 1.8:** Streamline Similarity Measures. (a) pointwise distance depends on not only the shape of two streamlines but also their positions and orientations. (b) distribution-based distance may fail to catch the order information. Therefore, two streamlines different in shape may have the same distribution of some attribute. (c) Procrustes distance measure by computing pointwise distance after registration.

As a matter of fact, pointwise distance calculation commonly used in proximity-based distance measures is not invariant under translation, rotation and scaling. Although other measures have also been presented that extract features from integral curves and consider feature distribution or transformation for a more robust similarity evaluation, none of them is able to explicitly capture intrinsic similarity that is invariant under translation, rotation and scaling. Figure 1.8 illustrates simple examples of measures using pointwise distance, distribution distance and our distance measure. Figure 1.8 (a) shows an example of pointwise distance. It computes the distance between two streamlines as the average of distances between closest points, as indicated by the brown dashed lines. We observe that using pointwise distance, two ellipses that are almost identical in shape may have a large distance due to their different locations and orientations. Figure 1.8 (b) shows an example of distribution distance. We first sample the two streamlines to generate distributions of velocity directions, as

indicated in the blue circle. These two streamlines are different in shape but share the same direction distribution, since the order of sample points is not preserved. In this case, these two streamlines may be considered to be almost identical using distribution-based distance measures. Figure 1.8 (c) demonstrates the Procrustes distance used in our approach. Streamlines are first aligned using registration, which translates, rotates and uniformly scales one streamline. Then the Procrustes distance is given by the pointwise distance calculated between the registered lines. We observe that the two elliptical curves align well and has a small distance, but the other pair of streamlines cannot be completely aligned with each other due to shape difference. Furthermore, most of the existing solutions for streamline similarity measures take each individual streamline of its entirety as the input, measuring partial streamline similarity is not naturally integrated. We propose a scheme that divides streamlines into basic segments, which allows partial streamline to be matched. In Chapter 5, we present FlowString, our approach for streamline similarity measures using shape invariant features.

### 1.3.4 Exploration of Multivariate Vascular Data Sets

It is common that a flow field also associates with multiple other scalar fields. For example, atmospheric data sets might include variables such as temperature, pressure, and concentration of $CO_2$, etc.; and vascular data sets might associate with pressure,

wall shear stress, and level set, etc. The relationship among these scalar fields and the flow field is of particular interest, since the changes of one field might lead to changes in other fields. However, we still lack effective approaches for guiding the users to explore these essential relationships.

Our multivariate exploration approach focuses on vascular data sets. Visualization of vessel data is critically important for disease diagnosis and therapeutic monitoring, surgical planning and simulations. Vascular data are referring to complex three-dimensional geometric data, hemodynamic data (e.g., time-resolved 3D blood velocity) and other parameters derived from hemodynamics. It is well known that cerebral aneurysm is a multi-factorial diseases and relationships among these parameters are not well understood. For example, Shojima et al. [94] found that the spatially averaged wall shear stress (WSS) in ruptured cerebral aneurysms was significantly higher, but the WSS was lower at the apex, which could be the thinnest part of the aneurysm and therefore may be vulnerable. However, Valancia et al. [111] suggested that low WSS regions were larger in ruptured aneurysms. They argued that low and oscillatory flow might be responsible for vessel wall remodeling associated with the rupture. Visual exploration of those hemodynamic parameters and their spatial distributions facilitates our understanding of this kind of disease.

In Chapter 6, we propose a web-based application specifically designed for analyzing the vascular data sets. It allows users to perform queries of regions that fulfill

user-specified criteria, and provides tools to analyze the extreme value of the scalar properties as well as their relationships. Finally, it provides a 2D representation that serves multiple visualization purposes.

## 1.4 Organization

This dissertation is organized as follows. Chapter 2 introduces the related works and discusses how our approaches relate to and differ from these approaches. Chapter 3 presents the unified information channel that solves the streamline selection and viewpoint selection problems simultaneously. Chapter 4 describes a focus+context flow visualization based on deformation to magnify the small features and reduce the occlusion over regions of interest while preserving the context streamlines. Chapter 5 introduces FlowString, a vocabulary approach for partial streamline matching, which facilitates efficient common pattern discovery and effectively locates features based on user query. Chapter 6 presents VesselMap, a web-based approach that allows users to interactively explore the vascular data sets based on histogram computation of particles traced in the vessels. Chapter 7 discusses several pedagogical visualization tools that provide users with an intuitive way to learn and understand some cryptography algorithms, including Data Encryption Standard (DES) cipher, RSA cipher, and ciphers based elliptic curves over finite fields.

# Chapter 2

# Related Work

In this chapter, we start with a review of previous work on flow visualization. Specifically, we review streamline visualization, which is a key technique in traditional flow field visualization. In Section 2.1, the two most important branches of streamline visualization are described: seed placement and streamline selection. Then, we discuss related algorithms that facilitate flow visualization. Section 2.2 presents previous work in focus+context visualization and Section 2.3 reviews the work in measuring the similarity among field-lines. For each section, we start with a general description of that field, followed by a brief introduction to each work. Finally, we link the previous work and our approach together and also point out the differences.

## 2.1 Streamline Visualization

Streamline visualization is the most extensively used approach to visualize the flow pattern in a steady flow field. It consists of two major approaches: seed placement and streamline selection. Seed placement focuses on finding good places to seed the streamlines, so that the traced streamlines will not only capture the pattern of the flow field but also maintain the streamline density to reduce occlusion. On the other hand, streamline selection starts with a large pool of streamlines and selects a good subset of streamlines to render, so that the selected streamlines will clearly show the flow pattern.

### 2.1.1 Seed Placement

Seed placement is a widely used strategy in flow visualization. Early work includes image-guided [109] and evenly-spaced [45] streamline placements. Similar to the evenly-spaced strategy [45], Mebarki et al. [78] suggested to place a seed at the position that is farthest from all existing streamlines, i.e., the center of the largest void area. Verma et al. [116] introduced a feature-based approach which detects critical points and uses seeding templates to capture the 2D flow field features. This approach was extended to 3D vector fields by Ye et al. [133]. Li and Shen [60] placed seeds

on a 2D projection plane and unprojected the seeds back to the 3D vector field to avoid clutter. Xu et al. [130] used seeding templates for regions with high entropy and then placed additional seeds at locations where the conditional entropy is high, i.e., much information is still unrevealed. Other seeding techniques include priority seeding [91], dual seeding [85], and surface seeding [97].

### 2.1.2 Streamline Selection

An alternative to seed placement is streamline selection. Previously, Chen et al. [13] defined a metric for local similarity between streamlines and used it to explicitly control the streamline density displayed. Marchesin et al. [74] measuresd the contribution of each streamline to the understanding of the vector field, and selected those streamlines that have higher contribution to the rendering and lower probability leading to visual clutter. Lee et al. [57] proposed to generate a maximum entropy projection buffer and then selected the streamlines that cause the minimum occlusion to important regions.

### 2.1.3 Our Approach

Our approach falls into the category of streamline selection. Marchesin et al. [74] evaluated a streamline based on its 3D linear and angular entropy, which is view-independent. For a particular viewpoint, they selected most streamlines according to the resulting density under that viewpoint. Unlike their method, our approach directly evaluates how much information a streamline could convey under a certain viewpoint by measuring the information lost from 3D to its 2D projection using mutual information. In this way, we build a transition matrix between all streamlines and all viewpoints, which is also known as the information channel. Furthermore, our view-independent streamline selection is performed by utilizing the information channel to consider all streamlines and their 2D projections in all sampled views at the same time. Due to the symmetric structure of the information channel, our approach can also suggest good viewpoints for users to observe the flow features, which is not achieved by previous methods.

## 2.2 Focus+Context Visualization

Focus+context (F+C) visualization stems from the need to show both overview (context) and detail information (focus) simultaneously within a limited display area.

Many F+C visualization techniques design lens that magnify focuses. Examples in this category are the fisheye view [31, 32, 89] for text, image and graph visualization, and the magnification lens [54, 121] and conformal magnifier [139] for image and volume visualization. Another category of techniques magnifies focuses without a predefined lens, e.g., Wang et al. [123, 124] achieved the desired F+C visualization through energy minimization. Some other approaches also emphasize the focus by adjusting the transparency of different regions. This is specially useful when the focus is not a region, but a property or variable in multivariate data sets, which is common for flow visualization.

In flow visualization, Fuhrmann and Gröller [30] presented magic lenses and magic boxes to examine the region of interest with greater detail by showing denser streamlines. This technique was later extended to magic volumes of varying focal regions such as cubes, prisms and spheres [75]. Laramee et al. [55] leveraged feature-based techniques [22] to extract interesting flow regions, such as stagnant flow, reverse-longitudinal flow and regions of high pressure gradient, as the focus. They achieved F+C rendering through interactive thresholding that reduces flow complexity and resulting visualization. Correa et al. [17] introduced physical and optical operators to intuitively visualize the inner regions of 3D flow through illustrative deformation. By cutting along flow traces, they allowed a clear observation of the inner regions through optical transformation and elastic deformation. To explore blood flow in

cerebral aneurysms, Gasteiger et al. [34] proposed an interactive 2D widget for flexible visual filtering and visualization of the F+C pairs (i.e., relevant hemodynamic attributes). Their widget supports local probing and conveys changes over time for the lens region. Van der Zwan et al. [112] modeled several visualizations of a data set as abstractions which represent the information at different levels of details. Given the user-selected level of focus, F+C visualization is achieved by manipulating the transparency of each abstraction. All these methods, however, do not shrink the context of the flow field while magnifying a specific focal region in order to best utilize the available volume space. In contrast, we devise a continuous deformation solution based on an energy optimization model to achieve F+C streamline visualization.

### 2.2.1 Our Approach

Unlike fisheye view or magnification lens, our method leverages an optimized deformation to minimize the global distortion, which can magnify multiple streamlines in different focal regions simultaneously. Closely related to our work are the F+C techniques presented by Wang et al. [123, 124] for surface models and volumetric data. In [123], they presented a F+C technique for surface models that magnifies a region of interest for closer examination while deforming other regions without perceivable distortion. In [124], a similar technique was presented for volumetric data for feature-preserving data reduction and F+C visualization. However, to the best of our

knowledge, the full benefits of such a deformation framework have not been demonstrated for interaction and visualization of flow fields. The value of our approach lies in that it targets on solving the several challenges when applying this grid-based deformation framework to flow fields. Unlike surface models and volumetric data where the visualization is 2D or 3D continuous, streamlines are only 1D entities and therefore, their F+C effect may not be readily perceivable yet the distortion could be much easier to spot. Therefore, we carefully design energy terms specifically for the flow field to maintain key physical properties for streamlines during the deformation process. Finally, to make this deformation framework truly useful, we incorporate the unique features of flow field and streamlines for both automatic and manual focus identification and F+C visualization.

## 2.3 Field-line Similarity Measures

A good field-line similarity measure would benefit flow visualization and exploration in many aspects. For example, it might guide streamline selection, so that similar streamlines could be removed to reduce occlusion. Furthermore, it could lead to approaches that search for similar flow patterns, and even automatically discover frequently appeared flow patterns in a flow data set. Over the years, many different similarity measures have been presented for field-line clustering. We categorize these measures into four different kinds: *proximity-based*, *feature-based*, *distribution-based*

and *transformation-based* measures. Note that many of the similarity measures we review in the following are actually hybrid ones: e.g., computing or extracting features first and then applying distribution-based or transformation-based solutions for measuring.

### 2.3.1 Proximity-based Measures

The measures used to determine the spatial proximity between two integral curves are the foundation for streamline similarity measures. The proximity of two streamlines can be defined based on the Euclidean distance of their sampled point pairs, with one point from each streamline. Such a measure captures the *spatial closeness* between elements and therefore can be used to determine the *geometric similarity* of the two curves. General examples include the closest point measure, the Hausdorff distance and the Fréchet distance. Customized examples include the average of point-by-point distances between corresponding pairs [21], the mean of thresholded closest distances [137], the mean of closest point distances [16], the thresholded average distance [12], and the weighted normalized sum of minimum distance [44]. For fiber bundle clustering in diffusion tensor imaging, Moberts et al. [81] evaluated combinations of four clustering methods and four similarity measures, and reported that the use of hierarchical single-link clustering combined with the mean of closest point distances gives the best results.

### 2.3.2 Feature-based Measures

While proximity-based measures are solely based on point locations, feature-based measures extract geometrical, topological or domain specific features from the vector field or integral curves for similarity analysis. For example, Chen et al. [13] compared randomly-seeded candidate streamlines based on the Euclidean distance among the streamlines as well as their shape and orientation. Shi et al. [93] leveraged the variation of different local and global geometric properties of pathlines for effective classification. Li et al. [63] used the bag-of-features approach to evaluate similarities among streamlines based on multiple scalar properties. Salzbrunn and Scheuermann [88] presented *streamline predicates* as functions that indicate the connection between streamlines and features selected by the user, such as which streamlines flow through a given vortex, separation bubble or shock wave.

### 2.3.3 Distribution-based Measures

Distribution-based distance measures aim to capture the feature distributions of integral curves for a more robust similarity comparison. Compared to pointwise distance measures, these measures are less sensitive to noise in the data and sharp turns or

twists at certain locations. Information-theoretic measures such as entropy, conditional entropy and mutual information have been extensively used in streamline seeding, importance ranking, and similarity measure [10, 74, 103, 130]. These measures are commonly based on the computation of the distribution of vector direction and magnitude over the field or the streamlines. Lu et al. [68] employed a distribution-based solution for robust streamline similarity evaluation and performed a detail study that explores different distance measures between two distributions. McLoughlin et al. [76] targeted rake-based streamline seeding and computed streamline signatures based on a set of curve-based attributes. Fast similarity comparisons are performed using the $\chi^2$ method on the derived signatures.

### 2.3.4 Transformation-based Measures

Transformation-based measures map data properties or features into a transformed space for measuring the similarity or difference between integral curves. For instance, Brun et al. [6] embedded the fiber tracts into a feature space for distance calculation, and created a weighted undirected graph which is partitioned into coherent sets using the normalized cut criterion. Wei et al. [125] extracted the shape of a streamline as a string by sampling curvature and torsion values at equal arc length intervals, and measured the similarity between two strings using the edit distance. Rössl and Theisel [86] presented *streamline embedding* which constructs a map from the space

of all streamlines to points in $R^n$ based on the preservation of the Hausdorff metric.

### 2.3.5 Our Approach

Our FlowString advocates a *shape-based* solution for streamline resampling, feature characterization, and pattern search and recognition. It distinguishes from all previous solutions in that it is specifically designed for robust and flexible partial streamline matching, invariant under translation, rotation and uniform scaling. We enable this through the construction of character-level alphabet and word-level vocabulary. A character is a primitive extracted from a streamline which is invariant to its geometric position and orientation. A word is a common pattern of concatenation of characters, which captures a meaningful pattern of the flow. Closely related to our work are the work of Schlemmer et al. [90], Wei et al. [125] and Lu et al. [68]. Schlemmer et al. [90] leveraged moment invariants to detect 2D flow features or patterns which are invariant under translation, scaling and rotation. However, their work is restrictive to 2D flow fields and patterns are detected based on local neighborhoods rather than integral curves. Wei et al. [125] extracted features along reparameterized streamlines at equal arc length and used the edit distance to measure streamline similarity. Features of varying scales are only roughly captured by simply recording the length of each resampled streamline. Lu et al. [68] computed statistical distributions of measures, such as curvature, curl and torsion, along the trajectory to measure streamline

similarity. Their approach is invariant to translation and rotation, but not scaling. Other than the geometry invariant similarity measure, our FlowString approach is also nicely integrated into a user interface to support intuitive and convenient user interaction and streamline exploration, expressing a more powerful way to visual analytics of flow field data.

## 2.4 Exploration of Multivariate Vascular Data

Computed tomography angiography (CTA) is commonly visualized with *maximum intensity projection* (MIP) and *direct volume rendering* (DVR) techniques. To enhance the perception of vessel structures, researchers often approximate vessel surface using model-based or model-free surface rendering approaches. The model-based surface rendering approach utilizes the information of centerline and radius, and approximates the vessel surface using models, such as truncated cones [36], B-splines [40], subdivision surfaces [27], or convolution surfaces [84]. The model-free surface rendering approach extracts the isosurface using algorithms such as marching cubes [67] based on a given threshold. Instead of approximating the vessel surface, Lathen et al. [56] proposed spatially varying transfer functions. It locally shifts the transfer function to enhance the perception of low intensity structure. Mistelbauer et al. [80] used halo rendering to enhance the lumen of vessel structure. Schumann et al. [92] used Multi-level Partition of Unity Implicits (MPUI) approach to reconstruct the

surfaces.

Unlike approaches based on MIP or DVR, other approaches flatten the vessel structure and map the corresponding information to 2D images. One of the commonly used approaches in this category is *curved planar reformation* (CPR). Kanitsar et al. [47] introduced CPR as a curved cutting through the data set along the centerline of a single vessel. Then, they extended the CPR approach to multi-path CPR (mpCPR) that supports multiple vessel branches and spiral CPR that flattens the vessel along a spiral to show its interior [47]. Kretschmer et al. [52] extended the mpCPR approach and used a bilateral filtering to remove undesired depth discontinuities. Mistelbauer et al. [79] proposed an approach based on CPR that aggregates the information around the centerline along circular rays. Borkin et al. [4] introduced a 2D vessel visualization method that uses a tree diagram to represent the structure of a coronary artery tree. Each branch is straighten and displayed as a tape with varying widths, which represents the diameter of the vessel. Zhu et al. [140] presented a work that produces flattened visualization of vessel branches. Two algorithms were proposed in this work. The first one is a conformal mapping algorithm by minimizing two Dirichlet functionals, and the second one adjusts the conformal mapping to produce a flatten representation that preserves areas.

Other than scalar volumes, vascular data sets often come with a simulated blood flow field as well. Recently, different flow visualization techniques have been developed

specifically for these data sets. van Pelt et al. [113] used various techniques to depict the blood flow and associated characteristics in different styles, together with an evaluation to measure the value of those visualization styles. Köhler et al. [51] extracted vortices in blood flow data sets using line predicates and highlighted the corresponding regions. van Pelt et al. [114] proposed to semi-automatically place and align a probe in the blood flow field, which serves as an seeding basis. Then, particles, integral curves and integral surfaces are used to convey distinct characteristics of the flow field. Born et al. [5] found the representatives of a bundle of lines, and used streamtapes with arrow heads to visualize the bundles. The tape-like structure provides a clear picture of how the representatives diverge and merge. Oeltze et al. [83] proposed to cluster the streamlines and use the cluster representatives for a clear view. They conducted a qualitative study on using different similarity measures, including geometry-based similarities and attribute-based similarities.

Due to the presence of multiple fields, some vessel visualization approaches also provide contextual information. Straka et al. [98] proposed VesselGlyph which combines both DVR and CPR. It depicts the vessels using CPR which is naturally placed in a DVR anatomic context. Mistelbauer et al. [79, 80] provided optional context rendering that displays the volume outside the lumen of vessel as well. Gasteiger et al. [34] presented a focus+context approach called FlowLens that uses some predefined lens templates to combine visualization results of different properties. The property of focus and property in context are both selected by users.

### 2.4.1 Our Approach

First, our approach focuses on providing an abstract view of the vessel structure in 2D, which enables the efficient interaction and observation. Among the above approaches, those that flatten the vessel and produce 2D images are closest to our approach. However, unlike those approaches, our 2D visualization serves not only as an overview but also as an interface for convenient interaction. We provide very concise information on the 2D visualization as a guidance to explore the data. Our approach does not require the centerline and radius information as CPR-based techniques do [46, 47, 79], or the triangulated vessel surface as the conformal mapping technique does [140].

Second, in terms of exploring multiple fields, our approach provides an interface guided by histograms of different properties to enable users to discover the statistical information for regions of interest. A segmentation scheme is proposed based on the local histograms of a user specified property. This facilitates the process of finding a region with a certain feature, or exploring relationships among different regions. It differs from those focus+context techniques [34, 98], which focus on visualizing multiple fields at the same time by blending several rendered results.

# Chapter 3

# A Unified Approach to Streamline Selection and Viewpoint Selection for 3D Flow Visualization

## 3.1 Overview

Streamline selection and viewpoint selection are two major problems in flow visualization. In this work [1] , we combine these two problems into a unified information-theoretic framework by constructing two interrelated *information channels* between

[1]The material contained in this chapter was previously published in *IEEE Transactions on Visualization and Computer Graphics 2013* [103].

a set of streamlines and a set of viewpoints. Based on the information channel from streamline to viewpoint, we define *streamline information* as a measure of streamline quality to guide streamline selection. Similarly, in the inverted channel from viewpoint to streamline, we define *viewpoint information* to guide viewpoint selection for the selected streamlines. Leveraging the two channels, we also present a unified algorithm for streamline clustering and viewpoint partitioning. In addition, a camera path is designed for automatic exploration of the flow field.

In our approach, the information channel is built in two directions $S \to V$ and $V \to S$, where $S$ is a set of streamlines and $V$ is a set of viewpoints. These two directions are characterized by two probability transition matrix $p(S|V)$ and $p(V|S)$, which connects two distributions $p(S)$ and $p(V)$, where $p(S)$ and $p(V)$ represent how interesting each streamline and viewpoint is, respectively. Intuitively, $p(s|v)$ indicates how informative a streamline $s \in S$ is from a viewpoint $v \in V$ and $p(v|s)$ indicates how appropriate a viewpoint $v$ is to show the information of a streamline $s$. Based on the transition matrix $p(V|S)$ and the distribution $p(S)$, we can update $p(V)$ to incorporate the contribution of each streamline, and based on $p(S|V)$ and $p(V)$, $p(S)$ can be updated similarly. In the following section, we will show how these probability distributions are used to capture the importance of each streamline and viewpoint, and how the relationship among streamlines and viewpoints are taken into consideration.

This chapter is organized as follows. Section 3.2 introduces the information channel

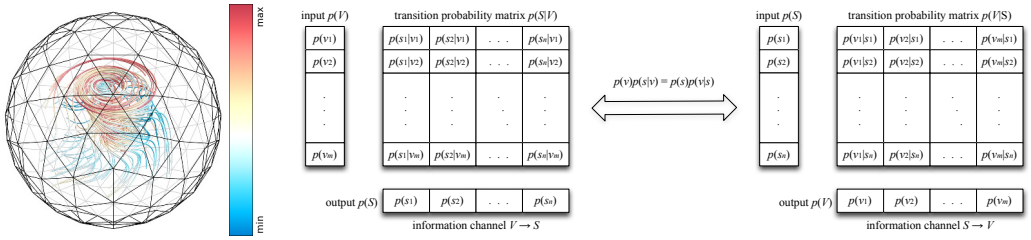
between a streamline set and a viewpoint set is introduced. Section 3.3 defines the conditional probability used in the information channel. Section 3.4 describes the best streamline selection and streamline clustering based on the information channel. Section 3.5 describes the best viewpoint selection and viewpoint partitioning and further explains the construction of a camera path. Finally, Section 3.6 compares the visualization results with existing approaches.

## 3.2 Information Channel

We propose to solve the problems of streamline selection and viewpoint selection in a single, unified framework. We consider a set of streamlines $S = \{s_1, s_2, \ldots, s_n\}$ and a set of viewpoints $V = \{v_1, v_2, \ldots, v_m\}$ as discrete random variables and build two interrelated information channels between them: $V \rightarrow S$ and $S \rightarrow V$. Our assumptions for viewpoints are (1) the flow field is centered in a sphere of sample viewpoints constructed from the recursive discretization of an icosahedron; and (2) the camera at a sample viewpoint is looking at the center of the sphere.

The main components in the information channel $V \rightarrow S$ are the following:

- The *transition probability matrix* $p(S|V)$ where conditional probability $p(s|v)$ represents the probability of "seeing" streamline $s$ from viewpoint $v$ (i.e., the




(a) (b)

**Figure 3.1:** We model the problems of streamline selection and viewpoint selection in a single, unified framework. (a) Sample viewpoints are constructed along a sphere from the recursive discretization of an icosahedron. Velocity magnitudes are mapped to streamline colors. (b) The information channel $V \rightarrow S$ (left) and the inverted channel $S \rightarrow V$ (right) are connected via the Bayes theorem. (© 2013 IEEE. Reprinted by permission.)

importance of $s$ with respect to $v$).

- The *input probability distribution* $p(V)$ where $p(v)$ represents the probability of selecting viewpoint $v$. If we assume $p(v)$ to be evenly distributed, then $p(v) = 1/m$ where $m$ is the number of sample viewpoints.
- The *output probability distribution* $p(S)$ where $p(s)$ represents the average probability that streamline $s$ is seen from all viewpoints $V$. That is, $p(s) = \sum_{v \in V} p(v)p(s|v)$.

Similarly, we can construct the inverted information channel $S \rightarrow V$, where the input and output probability distributions are swapped: $p(S)$ becomes the input and $p(V)$ becomes the output. In this inverted channel, the new transition probability matrix is $p(V|S)$, where $p(v|s)$ represents the probability of selecting viewpoint $v$ given streamline $s$. As shown in Figure 3.1 (b), these two channels are connected via the Bayes theorem, i.e., $p(v)p(s|v) = p(v, s) = p(s, v) = p(s)p(v|s)$, which provides us

a means to compute $p(v|s)$ given $p(v)$, $p(s)$, and $p(s|v)$.

## 3.3 Conditional Probability Definition

The key for deriving the information channel $V \rightarrow S$ lies in how to define the conditional probability $p(s|v)$. In our scenario, we consider the following two view-dependent factors for computing $p(s|v)$:

### 3.3.1 Mutual Information

This measure, denoted as $I(s; s_v)$, indicates how much information about streamline $s$ is revealed in its 2D projection $s_v$ under viewpoint $v$. We know that information loss is inevitable due to streamline projection. A large value of $I(s; s_v)$ shows that 3D streamline $s$ itself contains a high amount of information and its 2D projection $s_v$ preserves well the information of $s$. Therefore, the probability of "seeing" $s$ from $v$ is high. Conversely, if $s$ itself contains a low amount of information or its 2D projection $s_v$ loses much of the information of $s$, then the probability of "seeing" $s$ from $v$ is low. $I(s; s_v)$ is defined as [18]

$$I(s; s_v) = \sum_{i \in s} \sum_{j \in s_v} p(i, j) \log \frac{p(i, j)}{p(i)p(j)}, \tag{3.1}$$

where $p(i)$ and $p(j)$ are the marginal probabilities of $s$ and $s_v$ respectively, and $p(i, j)$ is their joint probability. Here we treat a streamline as a finite set of points. That is, $i$ and $j$ loop through the lists of points obtained either from streamline tracing or parameterization by the arc length along $s$ and $s_v$, respectively. To compute $p(i)$, we interpolate vectors from the original flow data based on the positions of all the points along $s$. These vectors are used to construct a 2D histogram based on vector magnitude and direction. To compute $p(j)$, we use the projections of these vectors along $s_v$ to construct the corresponding 2D histogram. To quantize vector directions, we use the recursive discretization of an icosahedron for 3D quantization, and the even circle partition by angle for 2D quantization. All vectors falling into the same range are quantized into the same bin of vector direction. The joint probability $p(i, j)$ can be computed by constructing a joint histogram for $s$ and $s_v$ where each of the two axes consists of all vector direction and magnitude combinations. In the joint histogram, the normalized bin count corresponds to $p(i, j)$.

### 3.3.2 Shape Characteristics

This property indicates how stereoscopic the shape of streamline $s$ is reflected under viewpoint $v$. Since the number of points along each streamline could be fairly large (e.g., in the order of hundreds or thousands of points), we propose to approximate a streamline using its *skeleton* for fast shape characteristics analysis. The "skeleton" of

a streamline is obtained using a uniform subsampling scheme along the integration points of the streamline to reduce the number of points to a smaller scale (e.g., in the order of tens of points). Let us denote the skeleton of streamline $s$ as $\tilde{s} = \{\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_k\}$, the viewing vector as $\overrightarrow{v}$, and the angle between $\overrightarrow{v}$ and $\overrightarrow{\tilde{p}_i\tilde{p}_{i+1}}$ as $\theta$. We define the shape characteristics of $\overrightarrow{\tilde{p}_i\tilde{p}_{i+1}}$ as

$$\alpha_{\tilde{p}_i\tilde{p}_{i+1};v} = \alpha_{\min} + (1.0 - \alpha_{\min})\Big(1.0 - \frac{|\pi/4 - \theta'|}{\pi/4}\Big), \tag{3.2}$$

where $\alpha_{\min}$ is the minimum value for the shape characteristics (we set $\alpha_{\min} = 0.1$ in this paper) and

$$\theta' = \begin{cases} \pi - \theta, & \theta > \pi/2 \\ \theta, & \theta \leq \pi/2 \end{cases} \tag{3.3}$$

The intuition is that $\alpha_{\tilde{p}_i\tilde{p}_{i+1};v}$ gets its maximum (minimum) value of 1.0 ($\alpha_{\min}$) when $\overrightarrow{v}$ and $\overrightarrow{\tilde{p}_i\tilde{p}_{i+1}}$ form a 45° or 135° (0°, 90°, or 180°) angle. The shape characteristics of streamline skeleton $\tilde{s}$ is defined as

$$\alpha_{\tilde{s};v} = \frac{\sum_{i=1}^{k-1} \alpha_{\tilde{p}_i\tilde{p}_{i+1};v} \|\tilde{p}_i\tilde{p}_{i+1}\|}{\sum_{i=1}^{k-1} \|\tilde{p}_i\tilde{p}_{i+1}\|}. \tag{3.4}$$

### 3.3.3 Conditional Probability

With mutual information and shape characteristics defined for streamline $s$ under viewpoint $v$, we define conditional probability $p(s|v)$ as

$$p(s|v) = \frac{\alpha_{\tilde{s};v} I(s; s_v)}{\sum_{s \in S} \alpha_{\tilde{s};v} I(s; s_v)}. \tag{3.5}$$

With $p(s|v)$ defined, besides simply assuming $p(v) = 1/m$, we can also obtain $p(v)$ from the normalization of the summation of all streamlines' conditional probabilities under $v$ over all viewpoints $V$. That is, $p(v) = p(S|v)/p(S|V)$, where $p(S|v) = \sum_{s \in S} p(s|v)$ and $p(S|V) = \sum_{v \in V} p(S|v)$. We use this nonuniform specification of $p(v)$ in our work.

## 3.4 Streamline Selection and Clustering

In this section, we propose two methods to evaluate streamline quality, so that the *best streamlines* can be selected to capture the features of the flow. In addition, we explain how the *representative streamlines* are selected guided by streamline information of a set of streamlines, and how the streamlines are clustered based on their distances to the representatives.

### 3.4.1 Best Streamlines Selection

We start from a pool of randomly or uniformly traced streamlines and select the best streamlines for display. The "best" streamlines are those that best capture flow features by passing through the vicinity of critical points or interesting regions. We propose two methods to evaluate each individual streamline and then introduce our

selection process.

Our first method uses the probability distribution $p(S)$. Since $p(s|v)$ indicates how interesting streamline $s$ is from viewpoint $v$, $p(s)$ gives us the summation of importance of $s$ from all viewpoints $V$. If the distribution $p(V)$ is not uniform, $p(s)$ can be considered as a weighted summation, in which a more interesting viewpoint has a higher weight.

Our second method uses the *streamline information* (SI). In the information channel $S \rightarrow V$, We define SI as

$$I(s;V) = \sum_{v \in V} p(v|s) \log \frac{p(v|s)}{p(v)}, \tag{3.6}$$

which represents the degree of dependence between streamline $s$ and the set of viewpoints $V$. Intuitively, SI indicates the quality of $s$ with respect to $V$. Note that $I(s;V)$ is the contribution of streamline $s$ to $I(S;V)$ which expresses the degree of correlation between the set of streamlines $S$ and the set of viewpoints $V$. Low values of SI correspond to streamlines seen by a large number of viewpoints in a *balanced* way. The term "balance" indicates that the conditional probability distribution $p(V|s)$ is similar to $p(V)$. This similarity can be expressed by the Kullback-Leibler divergence [53] between $p(V|s)$ and $p(V)$, which equals zero when $p(V|s) = p(V)$. Conversely, a high value of $I(s;V)$ means a high degree of dependence between streamline $s$ and the set of viewpoints $V$. Therefore, streamline $s$ that shows more information over

the set of viewpoints $V$ have a lower value of SI. The advantage of this streamline information over the streamline entropy, i.e., $H(V|s) = -\sum_{v \in V} p(v|s) \log p(v|s)$, lies in its robustness to deal with any type of discretization or resolution of the viewpoints $V$. This property has been shown by Viola et al. [117] in the context of volume visualization.

After streamline evaluation, we sort all the streamlines $S$ into a priority queue. If $p(s)$ is used, the streamlines are sorted in the *decreasing* order of $p(s)$, where a streamline with a higher value of $p(s)$ is preferred. If SI is used, the streamlines are sorted in the *increasing* order of SI, since a streamline with a lower value of SI is better.

The best streamlines are selected according to the sorted order. However, it is very likely that two or more streamlines are spatially close to each other and have a similar shape. In this case, those similar streamlines might not only convey redundant information, but also cause occlusion and clutter. Therefore, we use the *mean of closest point distances* as suggested by Moberts et al. [81] to evaluate streamline similarity. A streamline will not be selected if it is very similar to another streamline which is already selected.

### 3.4.2 Streamline Clustering

The streamline clustering algorithm also leverages the information channels built between $S$ and $V$. The first stage of our algorithm is to find the representative streamlines. Unlike the "best" streamlines (Section 3.4) which are evaluated individually, the "representative" streamlines are defined as a small set of streamlines in which the streamlines as an entirety best characterize the flow field. This is formed by selecting the streamlines such that their merging minimizes the distance to the target distribution $p(V)$. That is, our selection algorithm should select $n'$ streamlines ($n' \ll n$) so that their merging $\hat{s}$ minimizes $I(\hat{s}; V)$. Since finding an optimal solution to this algorithm is NP-complete, we adopt a greedy strategy by selecting successive streamlines to minimize $I(\hat{s}; V)$. At each merging step, we aim to maximize the Jensen-Shannon divergence between the set of previously merged streamlines and the new streamline to be selected.

Our solution proceeds as follows. First, we select the best streamline $s_1$ with distribution $p(V|s_1)$ corresponding to the minimum $I(s; V)$. Next, we select $s_2$ such that the mixed distribution $\frac{p(s_1)}{p(\hat{s})}p(V|s_1) + \frac{p(s_2)}{p(\hat{s})}p(V|s_2)$ minimizes $I(\hat{s}; V)$, where $\hat{s}$ represents the merging of $s_1$ and $s_2$ and $p(\hat{s}) = p(s_1) + p(s_2)$. At each step, a new mixed distribution

$$\frac{p(s_1)}{p(\hat{s})}p(V|s_1) + \frac{p(s_2)}{p(\hat{s})}p(V|s_2) + \ldots + \frac{p(s_i)}{p(\hat{s})}p(V|s_i), \tag{3.7}$$

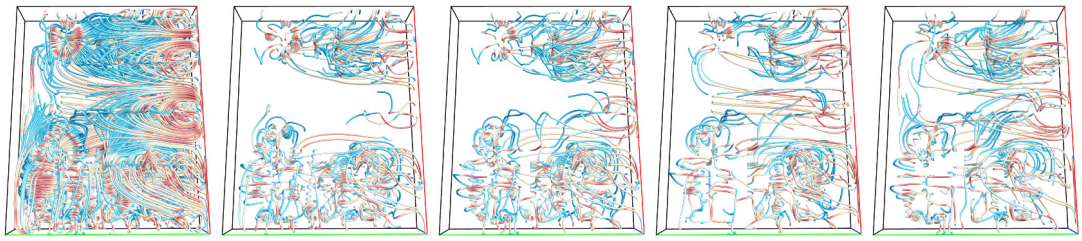



**Figure 3.2:** Streamline selection of the computer room data set. (© 2013 IEEE. Reprinted by permission.)

where $p(\hat{s}) = p(s_1) + p(s_2) + \ldots + p(s_i)$, is produced until the *streamline information ratio* (SIR), denoted as $I(\hat{s};V)/I(S;V)$, is lower than a given threshold or we have selected $n'$ streamlines. The SIR can be interpreted as a measure of the representativeness of the selected streamlines.

The second stage of our algorithm is to cluster other streamlines to the representatives we have identified in the first stage. Following the data processing inequality [18], we know that any clustering of streamlines reduces the mutual information $I(S;V)$ between the set of streamlines $S$ and the set of viewpoints $V$. Therefore, a good clustering is the one that minimizes this mutual information loss. Assuming that two streamlines $s_1$ and $s_2$ are merged into one cluster $\hat{s}$, the reduction of mutual information can be described by

$$\begin{aligned} \delta I(s_1;s_2) &= I(S;V) - I(\hat{S};V) \\ &= p(s_1)I(s_1;V) + p(s_2)I(s_2;V) - p(\hat{s})I(\hat{s};V), \end{aligned} \tag{3.8}$$

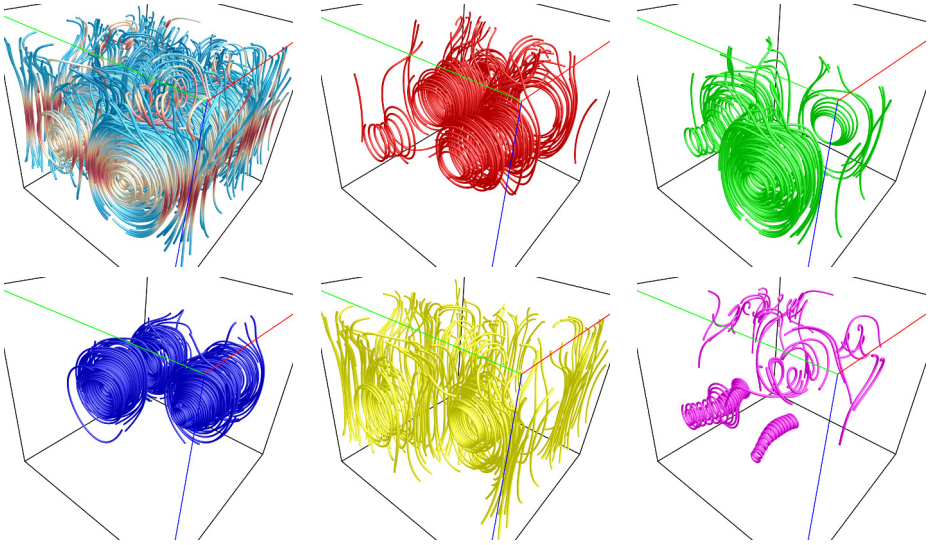


**Figure 3.3:** Streamline clustering of the two swirls data set. Five clusters are produced from 500 streamlines. The appropriate number of clusters is suggested by the elbow criterion. (© 2013 IEEE. Reprinted by permission.)

where $\hat{S}$ is the resulting streamline set and $p(\hat{s}) = p(s_1) + p(s_2)$. Note that $\delta I(s_1; s_2)$ is small if the two streamlines have very similar distributions (i.e., $p(V|s_1) \approx p(V|s_2)$) and it reaches zero if the two streamlines share the same distribution (i.e., $p(V|s_1) = p(V|s_2)$). At each step, we pick a streamline $s$ and calculate $\delta I(s; s')$ for each of the streamlines $s'$ in the representative set. Then, $s$ is merged into the cluster in which $\delta I(s; s')$ between $s$ and its representative $s'$ is minimal. Figure 3.2 shows an example, where streamlines are selected from an initial pool of 800 streamlines using the streamline selection based on $p(s)$, $I(s; V)$ and a representative set.

We use the elbow criterion to determine the proper number of clusters. That is, we should choose a number of clusters so that adding another cluster does not greatly increase the percentage of variance explained (i.e., the ratio of the between-group variance to the total variance). Specifically, if we plot the percentage of variance

explained by the clusters against the number of clusters, the first few clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph (the elbow). In practice, we run from two to ten clusters from which we choose the appropriate number of clusters. In Figure 3.3, five clusters are generated from 500 streamlines using the two swirl data set.

## 3.5 Viewpoint Selection and Partitioning

Similar to the streamline selection, we derive the *viewpoint information* to rank and select the *best viewpoints*. Similar to the streamline clustering, the viewpoints partitioning is performed by clustering the viewpoints to the representatives. Furthermore, we propose a camera path construction strategy to connect a set of viewpoints.

### 3.5.1 Best Viewpoints Selection

In the information channel $V \rightarrow S$, we define the viewpoint information (VI) as

$$I(v;S) = \sum_{s \in S} p(s|v) \log \frac{p(s|v)}{p(s)}, \tag{3.9}$$

which represents the degree of dependence between viewpoint $v$ and the set of streamlines $S$. Note that in our scenario, the set of streamlines now is actually the set of

*selected* streamlines, not the original pool of streamlines. This corresponds to (1) removing all rows in the transition probability matrix $p(V|S)$ in the channel $S \rightarrow V$ and the input probability distribution $p(S)$ for all streamlines that are not selected; and (2) renormalizing all remaining $p(s)$ in $p(S)$ and recomputing all $p(v)$ in the output probability distribution $p(V)$. For simplicity, we still use the notation $S$ in this section when referring to the selected streamlines.

Similar to streamline selection, the best viewpoints can be defined either by $p(v)$ or VI. If we use $p(v)$ to select the best viewpoints, we mainly consider the amount of information about the set of streamlines $S$ revealed by viewpoint $v$. As a result, the best viewpoints are those that show more information of $S$ than others. If we use VI to select best viewpoints, VI indicates the quality of viewpoint $v$ with respect to the set of streamlines $S$. Low (high) values of VI correspond to more independent (coupled) viewpoints. Thus, viewpoints with low values of VI are considered as better ones.

To avoid selection of viewpoints providing similar information, we filter the viewpoints according to similarity among them. Considering $p(S|v)$ as a vector associated with each viewpoint (i.e., $p(S|v) =< p(s_1|v), p(s_2|v), \ldots, p(s_n|v) >$), the difference between two viewpoints can be expressed as the Euclidean distance between their corresponding vectors. Thus, a viewpoint is not selected if its distance to any of the selected viewpoints falls below a given threshold $d_v$. In Figure 3.4, we show the best

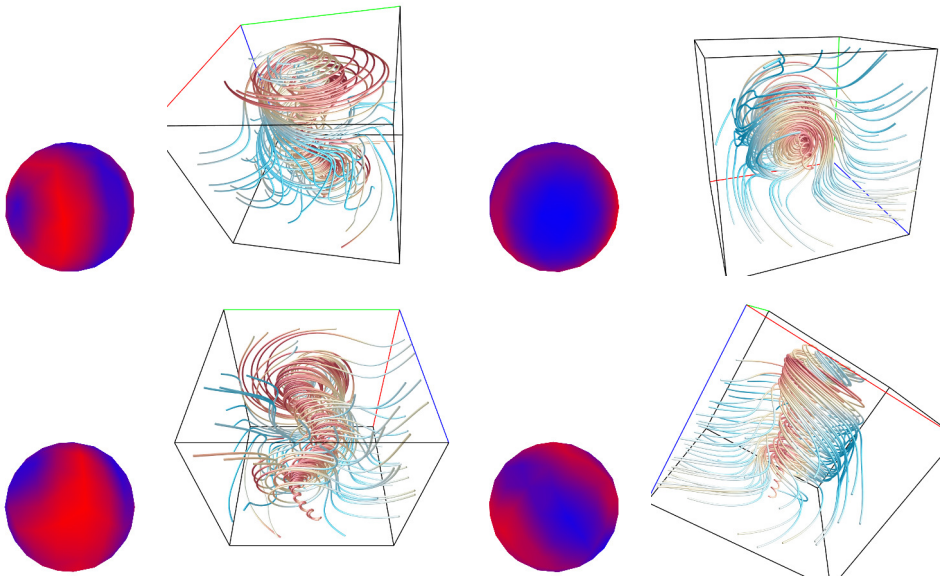



**Figure 3.4:** Viewpoint ranking of the tornado data set. In each of the view sphere images, red to blue is for the best viewpoint to the worst viewpoint. Streamline rendering from the best viewpoint and the worst viewpoint is also shown. All cases use the best streamlines selected. (© 2013 IEEE. Reprinted by permission.)

and worst viewpoints selected by $p(v)$ and $I(v;S)$, respectively. For the two best viewpoints, both the circular pattern and the stereoscopic pattern can be observed. But these patterns are not available in the worst viewpoints. Therefore, we consider these results are reasonable, although there may not be ground truth for best and worst viewpoints.

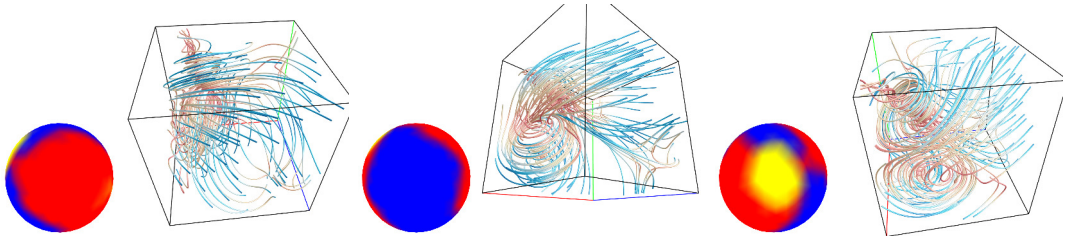


**Figure 3.5:** Viewpoint partitioning of the five critical points data set. We denote the three partitions in red, blue, and yellow, respectively. Streamline rendering corresponding to the viewpoint centering at each of the view sphere partition images is also shown. (© 2013 IEEE. Reprinted by permission.)

### 3.5.2 Viewpoint Partitioning

Similar to streamline clustering, we perform viewpoint partitioning in two stages. The first stage is the selection of representative viewpoints and the second stage is clustering other viewpoints to the representatives. The most representative viewpoints are a small number of viewpoints ($m' << m$) that provide the best representation of the selected streamlines. Leveraging the VI measure (Equation (3.9)), our solution for viewpoint selection is the same as the greedy solution we propose for identifying representative streamlines (Section 3.4) with the only difference being the swap of notations for streamline and viewpoint. The viewpoint selection process stops when the *viewpoint information ratio* (VIR), denoted as $I(\hat{v}; S)/I(V; S)$, is lower than a given threshold or we have selected $m'$ viewpoints. Similar to the SIR, the VIR can be interpreted as a measure of the representativeness of the selected viewpoints.

For viewpoint partitioning, we measure the difference between two viewpoints by the

reduction of mutual information, where the reduction $\delta I(v_1; v_2)$ is defined in the same way as $\delta I(s_1; s_2)$ (Equation (3.8)). Then, we apply the same procedure of streamline clustering to partitioning viewpoints in a similar manner: at each step, a viewpoint $v$ is merged into the partition whose representative $v'$ minimizes the information loss measured by $\delta I(v; v')$. Similarly, we use the elbow criterion to identify the proper number of partitions for all viewpoints. Figure 3.5 demonstrates an example of viewpoint partitioning using the five critical points data set.

### 3.5.3 Camera Path

Given a set of best or representative (Section 3.5) viewpoints, we construct a smooth camera path that goes through all selected viewpoints for automatic flow field exploration. Our algorithm creates a graph by treating all sample viewpoints as nodes and their neighboring relationships as edges. The weight of an edge is defined as the Jensen-Shannon divergence between the two viewpoints. With this graph, we can define the camera path by finding the shortest path among the set of selected viewpoints using the Dijkstra's algorithm. Specifically, we use the best (or the most representative) viewpoint as the starting point, and find the nearest viewpoint (with the minimum Jensen-Shannon divergence) from selected viewpoints as the next target viewpoint. The path between these two viewpoints is derived from the shortest path between their corresponding nodes in the graph. When the first target viewpoint is

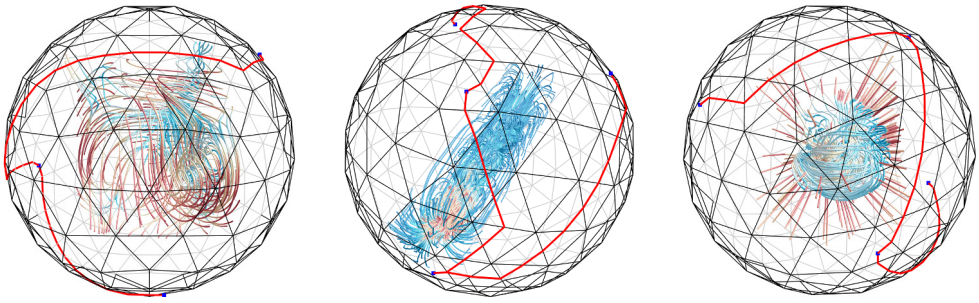


**Figure 3.6:** Camera paths for the five critical points data set (left), the solar plume data set (middle), and the supernova data set (right). (© 2013 IEEE. Reprinted by permission.)

achieved, we select a new target viewpoint among the rest of selected viewpoints and proceed in the same way until all viewpoints have been considered.

Figure 3.6 shows the camera paths we derived using the shortest path strategy. The shortest path is not based on geodesic distances, but according to the Jensen-Shannon divergences. Representative viewpoints were used to plan the camera path. Each path visits the representative viewpoints one by one. The resulting camera path is smooth because the shortest path between any two target viewpoints ensures that the change along the path is minimized. In other words, the viewpoints selected along the path are the most stable. Our solution is able to make meaningful exploration in an automatic fashion. For example, the camera path of the five critical points data set clearly reveals the two spirals, two saddles, and one source in the data.

## 3.6 Comparisons

We compared our streamline selections with two existing approaches by Xu et al. [130] and Marchesin et al. [74]. The visualization results are shown and discussed for a qualitative comparison. In addition, a user study was performed for a quantitative comparison.

### 3.6.1 Comparison with Other Methods

We compared our algorithm with other information theory based streamline placement and streamline selection algorithms. For streamline placement, we chose to implement a prototype of the entropy-guided streamline placement algorithm proposed by Xu et al. [130]. We implemented their template-based seeding technique based on the derived entropy field in conjunction with redundant streamline pruning. We used a moving window of $9^3$ to compute the entropy centered at each voxel. Vector directions are quantized into 50 bins for histogram computation. If a voxel has a high entropy value, we placed the seeds at the voxel and also its eight corners of the $9^3$ window. For streamline selection, we selected the view-dependent streamline visualization algorithm presented by Marchesin et al. [74]. We implemented their streamline evaluation based on angular and linear entropies and an approach similar

to their occupancy buffer to account for streamline occlusion. We weighted angular and linear components equally by setting $\alpha = \beta = 0.5$. For the initial streamlines, we used the same pool of streamlines used in our method. For streamline pruning in [130] and occlusion consideration in [74], we used the mean of the closest point distances between two streamlines. The threshold was set as 5.0. This parameter determines the minimum distance between any two streamlines in the streamline pool that can be selected for visualization.

Figure 3.7 shows the comparison of the results on six data sets. Our judgement is that our approach yields results that are as good as the ones produced by the other two methods for the first five data sets. In Sections 3.6.2, we show our user study to testify this. For the computer room data set, our results perform better. In [130], in order to avoid large voids, the seeding method needs to consider the conditional entropy between the original field and the field reconstructed from currently displayed streamlines to decide the additional seeding locations. In [74], using the linear and angular entropies actually favors streamlines that have a constant angle change and a constant segment length along the points of the streamlines. This criterion, however, actually prefers well-behaved streamlines and misses those interesting streamlines that vary greatly in length and angle along their points. Our information channel approach works well and is conceptually simple and easy to understand. It does not involve several steps as required in other methods for additional touch-up treatment (e.g., importance-based seed sampling in [130] and view-dependent streamline addition in

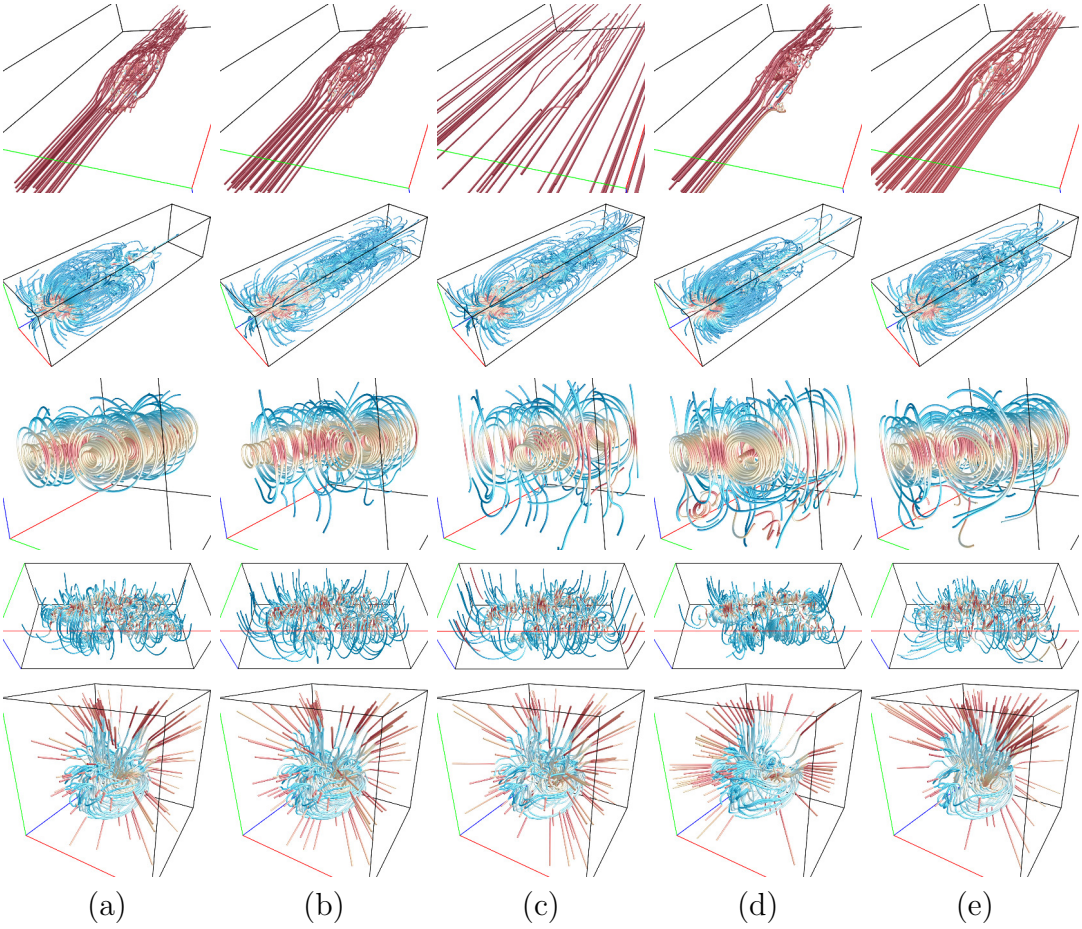



**Figure 3.7:** Comparison of our approach based on (a) $p(s)$, (b) $I(s;V)$, and (c) representative with (d) Xu et al. [130] and (e) Marchesin et al. [74]. Top to bottom are the car flow, solar plume, two swirls, crayfish, supernova, and computer room data sets, respectively. All five methods show the same number of streamlines: 40, 100, 60, 70, 100, and 100 for the six data sets, respectively. (© 2013 IEEE. Reprinted by permission.)

[74]). Moreover, our approach is powerful as the same solution for streamline selection applies to viewpoint selection in the inverted information channel. This feature is not available in other methods.

### 3.6.2 User Study

We conducted a user study to evaluate the effectiveness of our approach based on $p(s)$, $I(s;V)$, and representative (REP). We also implemented Xu et al. [130] and Marchesin et al. [74] for comparison. We did not use the conditional entropy to introduce new streamlines as in [130], because this technique could also be applied to other streamline selection methods to fill in void regions. All methods are view-independent, except for [74] where we selected the streamlines with respect to a good viewpoint and kept the set of streamlines selected for view-independent observation. The major goal of this study is to find out how effective our methods are compared to the existing ones and whether our methods work in the way they are designed to be.

The five methods were evaluated anonymously by a set of questions without timing followed by a feature identification task with timing. The users were 20 unpaid graduate students, including 12 students majoring in computer science and eight majoring in mechanical engineering, physics, and mathematics. All students majoring in computer science (CS) have knowledge in flow visualization and the students from other disciplines (non-CS) have flow field backgrounds. In the following, we describe the design of our user study, analyze the rating score, timing and accuracy results, and present the user comments.

#### 3.6.2.1 Rating Task Design and Procedure

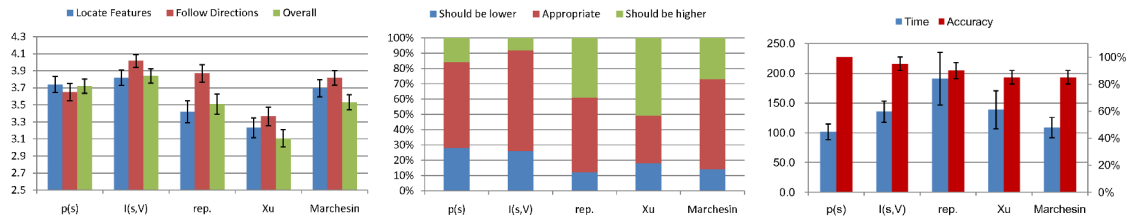
We conducted a *within-subjects experiment* for this task using five data sets: the car flow, crayfish, solar plume, supernova, and two swirls. Two more data sets were used for initial practice: the computer room and tornado. The users were asked to rate the five methods for each data set in the following three aspects:

- *ease to locate flow features and identify their patterns*;
- *ease to follow flow directions*; and
- *overall effectiveness to help understand the flow field.*

For each method, each of these three aspects was rated by an integer between 1 and 5 with 1 being the worst and 5 the best. We collected the evaluation scores and the background information of the users (rank and major). This part of the evaluation was not timed and the users had enough time to complete the work.

This user study was conducted in a lab using four PCs with the same configuration. Each PC has a monitor with the resolution of $1920 \times 1080$ and the visualization result occupied an $800 \times 800$ viewport. The users could sit in any fashion they found comfortable. They started with a practice session to become familiar with our visualization system and the rating criteria. They could ask questions about the interface, interaction, and rating criteria, but not which visualization result is better.

Evaluation activities began when the users felt ready and were performed one data set at a time. The users were not allowed to go back to a previous data set once they move forward. For each data set, the five methods were displayed anonymously in a random order, and the user could switch among the visualization results of all five methods for cross comparison. Specifically, we used a $5 \times 5$ Latin square for counterbalancing to rule out the learning effect. The order of methods for each of the five data sets was decided by a row of the Latin square. For the five methods to be evaluated, the users could rotate and zoom, but could not change the number of streamlines displayed. As a reference, the user could also display streamlines randomly selected from the streamline pool and rotate, zoom, and change the number of selected streamlines. This helps them answer questions such as if the pre-determined streamline density for each of the five methods is appropriate or not. Random selection also avoids any bias in the users' subsequent rating of the five methods. Two sets of open questions were asked for the crayfish and solar plume data sets, which required the users to elaborate why the most and least helpful methods were selected and to comment on the limitation of each method. For each user, it took about 20 minutes for introduction, 40 minutes for the rating tasks, and 10 minutes for the timing and accuracy tasks.




(a) (b) (c)

**Figure 3.8:** (a) Mean values and standard errors of user rating for "ease to locate flow features", "ease to follow directions", and "overall effectiveness". (b) User rating for streamline density. (c) Mean values and standard errors of the completion time (in seconds) and accuracy for identifying five critical points. (© 2013 IEEE. Reprinted by permission.)

#### 3.6.2.2 Effectiveness Evaluation

Using Kolmogorov-Smirnov test, we found out that most of our data do not pass the normality test. Therefore, instead of using ANOVA, we mainly used Kruskal-Wallis non-parametric test (KW-test) and Mann-Whitney $U$ test for effectiveness evaluation. We used significant level $\alpha = 0.05$ in all tests and investigated the following four important issues.

First, we study the effectiveness of locating flow features. Since Figure 3.8 (a) shows that the average scores for our and Xu et al. are lower than the others, there is a significant difference for the five methods ($H(4) = 11.35, p = 0.023$). Further analysis shows that excluding Xu et al. yields an insignificant result, and pairwise $U$-tests suggest significant difference between REP and Xu et al. and other methods. Consequently, $p(s)$, $I(s;V)$, and Marchesin et al. are comparable to each other

($H(2) = 1.40, p = 0.50$) and better than our REP and Xu et al. in terms of locating features. Additionally, both the CS ($H(4) = 4.39, p = 0.36$) and non-CS ($H(4) = 8.44, p = 0.077$) groups show no difference among the five methods. For the non-CS group, the $p$-value is much higher ($H(3) = 2.22, p = 0.53$) for the four methods excluding Xu et al. As mentioned earlier, we only implemented the entropy-based seeding part of Xu et al. Since a seed placed around the critical regions does not guarantee that the streamline will capture the features, this method might not show a clear flow pattern. Our REP is designed to focus on the general flow patterns, which makes it less effective to locate the features. However, our $p(s)$, $I(s; V)$, and Marchesin et al. are all based on streamline importance evaluation (albeit different criteria), which might explain why they were viewed similarly.

Second, we investigate the effectiveness of following flow directions. A significant effect is found for the five methods ($H(4) = 19.71, p = 0.0006$), and there is no significant difference for the remaining four methods if Xu et al. is excluded ($H(3) = 7.12, p = 0.068$). Moreover, the CS group ($H(3) = 8.67, p = 0.034$) exhibits a significant difference while the non-CS group ($H(3) = 2.59, p = 0.459$) does not. Our $I(s; V)$ has the highest average score of 4.02, and our REP and Marchesin et al. are very close, while that of Xu et al. is lower. Furthermore, since the $p$-values of $U$-test between Xu et al. against other methods are all small and the other four methods have no significant difference ($H(3) = 7.12, p = 0.068$), our $p(s)$, $I(s; V)$, REP, and Marchesin et al. do not have a significant performance difference. In addition, a

$U$-test was performed to compare locating flow features and following flow directions and the test result ($z = -2.62, p = 0.009$) suggests that REP is better in following flow directions (average = 3.87) than in locating features (average = 3.42). Since our REP only considers the overall information revealed by the selected streamlines without evaluating each individual streamline, its selection result provides a good indication in terms of general flow directions but does not guarantee that the detailed features will be captured.

Third, as for overall effectiveness, our analysis indicates that there is a significant difference for the five methods ($H(4) = 19.65, p = 0.0006$). If Xu et al. is excluded, no significant difference is found ($H(3) = 5.25, p = 0.155$). Thus, our $p(s)$, $I(s;V)$ and REP, and Marchesin et al. do not have a significant performance difference. The CS group and non-CS group do not exhibit in-group difference.

Fourth, for density analysis, the five methods do exhibit a significant difference ($H(4) = 25.32, p = 0.00004$). We divided the five methods into two groups, and found that there is no significant difference between our $p(s)$ and $I(s;V)$ ($H(1) = 0.04, p = 0.831$) and among the other three methods ($H(2) = 3.30, p = 0.192$). Therefore, our $p(s)$ and $I(s;V)$ do not have a significant performance difference and are better than our REP, Xu et al. and Marchesin et al. as indicated by the averages shown in Figure 3.8 (b).

#### 3.6.2.3 User Comments

For the crayfish data set, we asked the users which method was the most/least helpful to "locate and identify the features", and also requested them to comment on each method. Our $p(s)$ and $I(s;V)$ were each selected four times as the most helpful methods with similar reasons and typical comments were "*it provides general idea of surrounding streamlines, while putting more streamlines in the focus regions*" and "*it captures the characteristics of the feature regions with less occlusion*". One user selected $p(s)$ as the least helpful method because the feature regions were too dense, and three users selected $I(s;V)$ as the least helpful one because the feature regions could be a little denser. REP was selected by two users as the most helpful one to locate the features, yet by seven users as the least helpful one, although this method does not focus on the feature regions. Xu et al. was selected five times as the most helpful method, but it was also selected seven times as the least helpful one. Some users stated that it mainly placed streamlines in the interesting regions, which made the features stand out, while other users considered the feature regions to be too cluttered. Marchesin et al. was rated as the most and least helpful methods by five and two users, respectively, with similar reasons as our $I(s;V)$.

For the solar plume data set, we asked the users to select the most/least helpful method to "show the flow directions", and also requested them to comment on each

method. Our REP was selected by eleven users as the most helpful one, mostly due to "*it fills the entire volume evenly without much occlusion*". Our $I(s;V)$ was also considered as the most helpful one by six users for a similar reason. Note that these two methods are the only two that take the spacing and overall density into consideration. Our $p(s)$ was rated as the least helpful one by fifteen users, since it left a large portion empty. Xu et al. was selected as the most helpful one by two users, since they believed a few streamlines were enough for the non-feature regions. On the other hand, three users considered it as the least helpful one because some regions were too sparse. Marchesin et al. was neither selected as the most helpful one nor as the least helpful one.

#### 3.6.2.4 Timing and Accuracy Task

We conducted a *between-subjects experiment* for this task using the five critical points data set. The ABC flow data set was used for initial practice. The users were asked to locate the five critical points in the task. Since it would be difficult to locate 3D points using mouse, the users selected only the 2D projection of each critical point by mouse clicking. For each critical point selected, an image was saved with a red circle marking the selected position. We then graded these images manually to derive the accuracy of user selection. Each user was required to complete the task with one method, and each method was performed by four users. We informed the users

that accuracy is more important than timing, so that they would try their best to identify the correct locations of critical points. Moreover, the users could also switch to previous selection results and make modification if needed. The timer started when the data set was displayed, and stopped when the users clicked a button to finish.

Figure 3.8 (c) shows our $p(s)$ has the shortest average completion time and is closely followed by that of Marchesin et al. Our REP has the longest average completion time with the largest standard error, since the representatives do not necessarily capture the features. In terms of accuracy, our $p(s)$ is the highest (100% correct), while the other methods are close. In terms of the type of critical points, seven users missed one saddle, and one user missed one saddle and one spiral. This is probably because streamlines passing a saddle do not have high importance values compared to those passing spirals. We also observed that most users took a long time to find saddles. Among the five methods, our $p(s)$ appears to be the best one in terms of capturing saddles, since all users located the two saddles successfully. Both timing and accuracy results indicate that our $p(s)$ is a good performer in terms of locating features. However, this is not verified by statistical testing, since our sample size is too small.

#### 3.6.2.5 Summary

Our methods are designed to focus on different aspects: $p(s)$ selects more streamlines that show interesting patterns, REP mainly produces evenly-spaced results, and $I(s;V)$ is somewhat in between. The major goal of this evaluation is to determine whether our methods are effective in the way they are designed to be. The averages of rating scores seem to support this to some degree. Our REP has high scores for following flow directions and low scores for locating flow features, and $p(s)$ has higher scores for locating features than following flow directions. In addition, our $I(s;V)$ has the highest average scores for all the three aspects. The timing and accuracy study shows a consistent result that $p(s)$ has highest accuracy with the least completion time, while REP takes the longest time to complete.

Hypothesis tests based on KW-test suggest that our methods do not have a significant performance difference as other existing methods. KW-test also indicates that our REP is more helpful to follow flow directions than to locate flow features, which confirms that it focuses on a different aspect compared to other methods. This is also an advantage, since we may benefit not only from the fact that our framework can provide different meaningful results, but also from the potential that we can develop a hybrid method based on this framework.

User comments indicate that streamline density is a very important factor. The methods that generate higher density around the feature regions and lead to a balanced overall density are highly appreciated. We also found that there was a connection between the three ratings and the density rating (Figure 3.8 (b)). Our $I(s;V)$, which has the highest average score, also has the highest percentage of being rated "appropriate". The users tended to rate the density of methods that are not satisfactory to be either "should be higher" or "should be lower", although some users also mentioned that the problems for those methods might be the locations of streamlines instead of the number of streamlines. The methods that miss certain kind of streamlines are more likely to be rated "should be higher", e.g., our REP might miss the features and Xu et al. might miss the surrounding streamlines. Finally, the methods that place many streamlines in the feature regions are often rated "should be lower".

# Chapter 4

# A Deformation Framework for Focus+Context Flow Visualization

## 4.1 Overview

Our focus+context (F+C) flow visualization[1] is designed to magnify small flow features while keep all the context information and reduce occlusion over critical regions at the same time. The basic idea of our F+C flow visualization is to partition the flow field's volume space into blocks and deform the blocks to guide streamline repositioning. Given a vector field, we uniformly partition it into a grid space, $\mathbf{G} = \{\mathbf{V}, \mathbf{E}, \mathbf{B}\}$,

[1]The material contained in this chapter was previously published in *IEEE Transactions on Visualization and Computer Graphics 2014* [105].

where $\mathbf{V}$ is the set of all vertices $\mathbf{V} = \{\mathbf{v}_0^T, \mathbf{v}_1^T, \ldots, \mathbf{v}_n^T\}$, and $\mathbf{E}$ and $\mathbf{B}$ are the sets of all edges and all blocks, respectively. During a deformation, we compute a new set of vertex positions $\mathbf{V}' = \{\mathbf{v}_0'^T, \mathbf{v}_1'^T, \ldots, \mathbf{v}_n'^T\}$, with the intention that the deformed blocks under focus will grow while others blocks will shrink. Clearly, some distortion will be introduced in this deformation process. By minimizing the energy function described in Section 4.4, we aim to spread the unresolved distortion to the blocks according to their importance values, so that interesting blocks of focus can maintain their shapes while less interesting blocks and empty blocks can absorb more distortion and even be squeezed excessively into a plane. Our deformation framework consists of four key steps: *block importance evaluation*, *manual feature specification*, *grid space deformation* and *streamline repositioning*. The user can choose automatic block importance evaluation and/or manual feature specification for F+C visualization. Note that although the outputs of some steps depend on how the streamlines are placed or selected, our deformation framework can work with any streamline placement and selection algorithm.

This chapter is organized as follows: in Section 4.2 and Section 4.3, we introduce the automatic block importance evaluation and user-specified importance based on different feature templates, respectively; in Section 4.4, the grid deformation based the block importance values is covered; in Section 4.5, we explain how to reposition the streamlines based on the deformed grid and discuss the evaluation of errors caused by the deformation; in Section 4.6, we present the visualization results and compare

our method with the traditional fisheye view; finally, in Section 4.7, the results of an empirical evaluation is presented.

## 4.2 Block Importance Evaluation

Our deformation framework supports two different ways to define the importance of a block. One is to automatically derive block importance based on flow information, such as the flow entropy [10, 130]. The other way is to manually decide block importance by incorporating user input (Section 4.3). Once we derive the importance values of all blocks, we normalize them to $[0, 1]$ and use them as the weighting factors for individual block expansion (Section 4.4). Note that our deformation framework does not depend on any specific approach for importance evaluation. Thus, other importance evaluation techniques could also be applied.

### 4.2.1 Automatic Importance Computation

For the automatic importance evaluation, we measure the importance of a block using its entropy. Intuitively, by considering both the magnitude and direction of a vector, the blocks that contain simple flow patterns will have small entropy values, since the vectors in those blocks are similar; while the blocks that contain complicated flow

patterns will have large entropy values, since the vectors in these blocks might vary in both direction and magnitude.

There are multiple ways to compute the entropy of a block. For example, we can compute the entropy of a block over the vectors at all points sampled along the streamlines passing through the block, or we can compute it over the vectors at all grid points inside the block. In our framework, we apply the first method, since it considers only the current streamline pool, and will give a high entropy value for blocks that are intersected by the generated streamlines and zero for blocks that do not contain any streamline. We favor this method since the original volume of vectors are not directly visible to user. The features can only be observed if they are captured by the streamlines. Thus, by applying the first method, we consider those regions that contain interesting patterns and are well captured by the streamlines to be more important. The flow patterns that are more complicated and difficult to predict are considered to be more interesting. Note that the importances of blocks also depend on the tasks of users. In some scenarios, the users might prefer to enlarge a region even if there are only a few streamlines passing through it. We enable this through manual feature specification (Section 4.3).

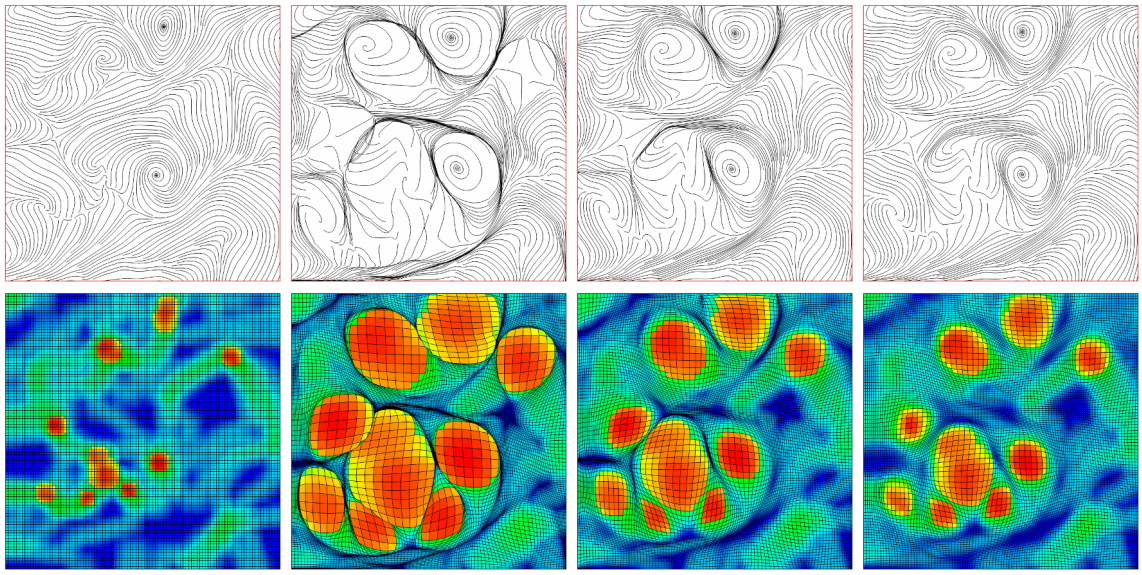



**Figure 4.1:** Automatic multi-F+C visualization on a 2D flow field and the corresponding block importance grid. (a) is the original streamline visualization. (b) is the naïve deformation that only considers individual block expansion. (c) is the deformation with adding neighboring block smoothing. (d) is the deformation with considering both neighboring block smoothing and flow-aware adjustment. (© 2014 IEEE. Reprinted by permission.)

### 4.2.2 Flow-aware Adjustment

The computed entropy field might has some discontinuities, where the streamlines cross multiple blocks of varying importance value might suffer from greater distortion. The distortion will be obvious when a less important block is surrounded by the important ones, as shown in Figure 4.1 (c). To address this problem, we introduce flow-aware adjustment to smooth the importance values between neighboring blocks that share a large number of streamlines. The importance values are considered as some energy term: when the flow moves from an important block to a less important

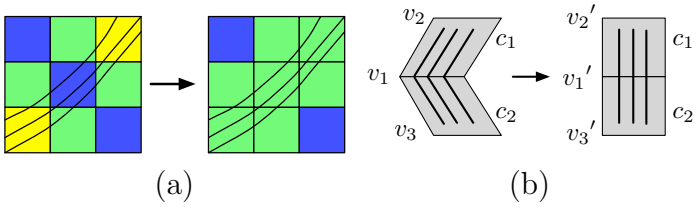



**Figure 4.2:** (a) flow-aware adjustment. (b) flow-aware smoothing. (© 2014 IEEE. Reprinted by permission.)

one, it will also carry the energy along with it. In other words, neighboring blocks sharing more streamlines in common should have more similar importance values, as illustrated in Figure 4.2 (a). This can be formulated as minimizing the following term

$$D_d = \sum_{\mathbf{b}_i} \sum_{\mathbf{b}_j \in B(\mathbf{b}_i)} w_{ij} \big(I'_i - I'_j\big)^2 + \sum_{\mathbf{b}_i} \beta \big(I'_i - I_i\big)^2, \tag{4.1}$$

where $B(\mathbf{b}_i)$ is the set of neighboring blocks of block $\mathbf{b}_i$, $w_{ij} = n_{ij}/n_{\max}$ is a normalized weight, $n_{ij}$ is the number of streamlines shared by blocks $\mathbf{b}_i$ and $\mathbf{b}_j$, $n_{\max}$ is the maximum number of streamlines shared by two neighboring blocks, $\beta$ is a user-specified weight, $I_i$ is the importance value of $\mathbf{b}_i$ before the adjustment, and $I'_i$ and $I'_j$ are the importance values of $\mathbf{b}_i$ and $\mathbf{b}_j$ after the adjustment, respectively. This is similar to filtering the importance values along the streamlines. Note that the adjusted field will be closer to the original one if $\beta$ is larger, and the effect of this flow-aware adjustment will be weaker. The adjusted field will be almost unchanged when $\beta = 1.0$. Since Equation (4.1) is a quadratic function, the method of least squares can be used to solve for the importance values.

## 4.3 Manual Feature Specification

Besides the automatic importance evaluation, we can also assign importance values based on the focal region specified by the user. Our approach provides two different options for manual feature specification: *block focus* and *streamline focus*.

### 4.3.1 Block Focus

For this option, the user simply clicks on the visualization result to specify a block as the center of focus. For a 3D flow field, the visualization result is a 2D projection, so we still need to estimate the depth value to pinpoint the focal point. A straightforward solution is to follow the first hit on the streamlines displayed in the projection, but the focal region selected in this way may miss internal flow features. A better solution which we will use is to identify the most prominent feature along the direction of projection. For instance, Lee et al. [57] presented the concept of maximal entropy projection (MEP). For each pixel on the screen, a ray is cast into the entropy volume and the $z$-value is given by that of the voxel with the maximal entropy. In this way, we will select the most important block along the ray specified by the user. We can use this manual feature selection to modulate the automatic importance evaluation and modify the automatic focus result.

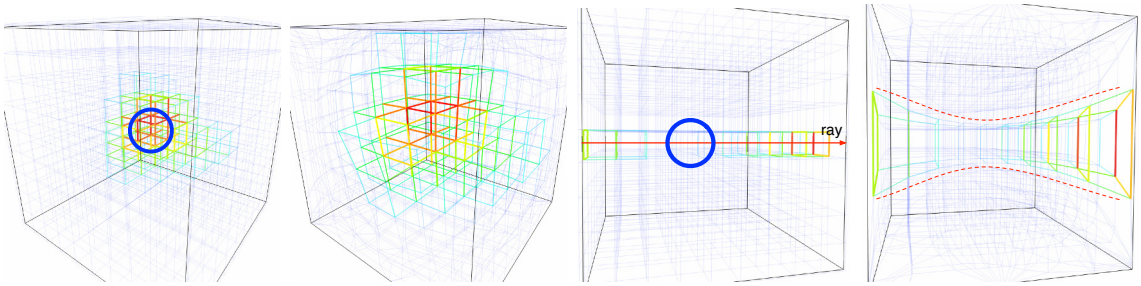



**Figure 4.3:** The grids before and after the deformation using block focus. (a) and (b) show original and deformed grids for the spherical block focus, where the focus is at the center of the volume with the largest importance value. (c) and (d) show original and deformed grids for the hourglass block focus, where the focus is highlighted with the blue circle in the first row and the deformed shape is enhanced by the red dashed lines for clearer exposition. (© 2014 IEEE. Reprinted by permission.)

We can also use the manual focus independently, starting from uniform importance values for all blocks and modulating those values according to some predefined templates. We design two templates for exploring small or occluded flow features: *spherical block focus* and *hourglass block focus*. Examples are shown in Figure 4.3. The spherical block focus assigns the largest importance value to the specified block, and the importance values gradually decrease for blocks further away from the focus. This template is suitable for magnifying small features, since the center region, which contains the features, will grow as other regions shrink.

This template, however, could be ineffective when the features are hidden by other streamlines due to occlusion, since the blocks located at the outer rings or layers could be denser than those at the center. The hourglass block focus is designed to solve this problem. Instead of magnifying the feature region, we enlarge the blocks

along the user-specified ray except the feature region. It assigns larger importance values for the blocks that are closer to the ray and gradually decreases the values for those further away. Furthermore, it assigns smaller importance values for the blocks whose depth are close to the depth of the specified block. In this way, the blocks with large importance values form the shape of an hourglass, whose axis is along the ray and whose center is the block of focus. By magnifying the blocks that occlude the block of focus, we are now able to see through and observe the flow features that are previously occluded.

### 4.3.2 Streamline Focus and Animation

Another useful way of manual focus specification is to allow the user to select a streamline of interest through first hit. We then perform F+C visualization on the entire streamline by assigning larger importance values for the blocks that the streamline goes through and smaller values for other blocks. An animation of F+C visualization can also be generated by moving the focal point along a streamline from end to end. To produce smoother animation, we insert additional frames, in which the grid vertices are linearly interpolated from the grid vertices of the two neighboring frames and the streamlines are repositioned according to the intermediate grids. In cases where there are no desired streamlines going through the regions to be explored, a user-drawn path could be used instead for an effective exploration.

## 4.4 Grid Space Deformation

The grid space deformation should serve the following purposes: the blocks in focus regions should be magnified while pushing the context regions to be shrunk; the streamlines in focus regions should keep their shape unchanged while the distortion for other streamlines should be minimized; and the relative position among streamlines should not change dramatically. To perform grid space deformation, we consider *individual block expansion*, *neighboring block smoothing*, and *flow-aware smoothing*, inspired by Wang et al. [124]. Individual block expansion allows each block to resize independently based on a global scaling factor and its weighting factor (Section 4.4.1). Neighboring block smoothing preserves the continuity of neighboring blocks while flow-aware smoothing preserves the shape of the streamlines. Please refer to (Section 4.4.2) and (Section 4.4.3) for details. We add *edge flipping constraints* to avoid neighboring block intersection and *volume boundary constraints* to retain the size and shape of the bounding space. We formulate these considerations into energy terms and search for a deformed grid that minimizes the objective function under the edge flipping and volume boundary constraints. To achieve this, we transform the objective function into a linear system and solve for the unknown vertex positions in a least-squares sense.

### 4.4.1 Individual Block Expansion

We introduce this energy term to preserve the cube shape of the blocks. Given a block $\mathbf{b}_i$, ideally, its deformed version $\mathbf{b}'_i$ can be obtained by applying scaling, rotation and translation to the original block, i.e., $\mathbf{b}'_k = s_{\mathbf{b}_k}\mathbf{R}_{\mathbf{b}_k}\mathbf{b}_k + \mathbf{t}_{\mathbf{b}_k}$, where $s_{\mathbf{b}_k}$ is a scaling factor, $\mathbf{R}_{\mathbf{b}_k}$ is a $3 \times 3$ rotation matrix and $\mathbf{t}_{\mathbf{b}_k}$ is a vector indicating the translation. Note that by multiplying the rotation term $\mathbf{R}_{\mathbf{b}_k}$, we allow the block to rotate in order to better utilize the space and incur less distortion. Denoting the set of edges of block $\mathbf{b}_k$ as $E(\mathbf{b}_k)$, we express the energy term of block deformation as

$$D_f(\mathbf{b}_k) = \sum_{\mathbf{e}_{ij} \in E(\mathbf{b}_k)} w_{\mathbf{b}_k} \|\mathbf{e}'_{ij} - s_{\mathbf{b}_k}\mathbf{R}_{\mathbf{b}_k}\mathbf{e}_{ij}\|^2, \tag{4.2}$$

where $w_{\mathbf{b}_k}$ is the normalized importance value of block $\mathbf{b}_k$, $\mathbf{e}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ and $\mathbf{e}'_{ij} = \mathbf{v}'_i - \mathbf{v}'_j$ are the edges before and after the deformation, respectively. The translation $\mathbf{t}_{\mathbf{b}_k}$ is canceled out due to the simple fact

$$\mathbf{e}'_{ij} = \mathbf{v}'_i - \mathbf{v}'_j = (s_{\mathbf{b}_k}\mathbf{R}_{\mathbf{b}_k}\mathbf{v}_i + \mathbf{t}_{\mathbf{b}_k}) - (s_{\mathbf{b}_k}\mathbf{R}_{\mathbf{b}_k}\mathbf{v}_j + \mathbf{t}_{\mathbf{b}_k}) = s_{\mathbf{b}_k}\mathbf{R}_{\mathbf{b}_k}(\mathbf{v}_i - \mathbf{v}_j). \tag{4.3}$$

Initially, we set $s_{\mathbf{b}_k}$ to a user-defined scaling factor $s_f$ and $\mathbf{R}_{\mathbf{b}_k}$ to an identity matrix for all blocks to solve for a new set of vertex positions $\mathbf{V}'$. Then, for each block, we compute $s_{\mathbf{b}_k}$ and $\mathbf{R}_{\mathbf{b}_k}$ from the deformed vertices, and apply the updated $s_{\mathbf{b}_k}$ and $\mathbf{R}_{\mathbf{b}_k}$ to solve for another set of vertex positions. This procedure is repeated for several iterations, until the system converges or a predefined number of iterations is

reached. Although we assign the same scaling factor $s_f$ for all the blocks, only the blocks with larger importance values can get the chance to be enlarged, since they will receive larger penalty (i.e., increase $D_f(\mathbf{b}_k)$) for not approaching the target scaling factor. In contrast, those less important or trivial blocks, with very small or even zero importance values, can be squeezed substantially without receiving much penalty.

### 4.4.2 Neighboring Block Smoothing.

We introduce this smoothing term in order to reduce the size difference between neighboring blocks. As shown in Figure 4.1 (b), such a naïve deformation distorts streamlines that span across multiple blocks and lead to pronounced artifacts along block boundaries. To avoid this, we preserve the Laplacian coordinates [96] of the deformed vertices $\mathbf{v}_i'$ by minimize the following energy term

$$
\begin{aligned}
D_\ell = \sum_{\mathbf{v}_i \in \mathbf{V}} \|\mathbf{L}(\mathbf{v}_i') - s_{\mathbf{v}_i}\mathbf{R}_{\mathbf{v}_i}\mathbf{L}(\mathbf{v}_i)\|^2, \text{ where} \\
L(\mathbf{v}_i) = \frac{1}{|V(\mathbf{v}_i)|}\sum_{\mathbf{v}_j \in V(\mathbf{v}_i)}(\mathbf{v}_i - \mathbf{v}_j),\ s_{\mathbf{v}_i} = \frac{1}{|B(\mathbf{v}_i)|}\sum_{\mathbf{b}_k \in B(\mathbf{v}_i)} s_{\mathbf{b}_k}, \\
\text{and} \quad \mathbf{R}_{\mathbf{v}_i} = \frac{1}{|B(\mathbf{v}_i)|}\sum_{\mathbf{b}_k \in B(\mathbf{v}_i)} \mathbf{R}_{\mathbf{b}_k}.
\end{aligned} \tag{4.4}
$$

In Equation (4.4), $s_{\mathbf{v}_i}$ and $\mathbf{R}_{\mathbf{v}_i}$ are the scaling factor and rotation matrix for the Laplacian coordinates of vertex $\mathbf{v}_i$, respectively; $V(\mathbf{v}_i)$ and $B(\mathbf{v}_i)$ are the sets of neighboring

vertices and blocks of $\mathbf{v}_i$, respectively; and $|V(\mathbf{v}_i)|$ and $|B(\mathbf{v}_i)|$ are the numbers of neighboring vertices and blocks of $\mathbf{v}_i$, respectively. Since the Laplacian coordinates are zero vectors for all the inner vertices, we can actually simplify Equation (4.4) to

$$D_\ell = \sum_{\mathbf{v}_i \in \mathbf{V}} \|\mathbf{L}(\mathbf{v}'_i)\|^2, \tag{4.5}$$

and only add these constraints to the inner vertices. To simplify the calculation for the boundary vertices, we apply a similar approach that only considers the inner vertices on each boundary face. For each vertex, the Laplacian coordinates are computed from the four neighboring vertices that are also located on the boundary face. In this way, the Laplacian coordinates are still zero vectors and therefore we do not need to apply the scaling and rotation.

### 4.4.3 Flow-aware Smoothing

Flow-aware smoothing serves a similar purpose as flow-aware adjustment in the block importance evaluation step. By introducing this term into the deformation process, we are able to reduce the difference between the transformations of two neighboring blocks that share a large number of streamlines. In the left side of Figure 4.2 (b) on page 76, we illustrate an example of severe streamline distortion, where straight streamlines are deformed into polylines due to the different orientations of the two neighboring blocks $\mathbf{b}_1$ and $\mathbf{b}_2$. To reduce this kind of distortion, we drag $\mathbf{v}_1$ shared

by $\mathbf{b}_1$ and $\mathbf{b}_2$ back to the center of the two adjacent vertices $\mathbf{v}'_2$ and $\mathbf{v}'_3$, as shown in the right side of Figure 4.2 (b). The flow-aware smoothing can be achieved by minimizing the energy term

$$D_s(\mathbf{b}_i, \mathbf{b}_j) = \sum_{\mathbf{v}_i \in V(\mathbf{b}_i) \cap V(\mathbf{b}_j)} w_{ij} \|\mathbf{v}'_j + \mathbf{v}'_k - 2\mathbf{v}'_i\|^2, \tag{4.6}$$

where $V(\mathbf{b}_i) \cap V(\mathbf{b}_j)$ is the set of vertices shared by blocks $\mathbf{b}_i$ and $\mathbf{b}_j$, $w_{ij}$ is defined in the same way as in Equation (4.1), and $\mathbf{v}'_j$ and $\mathbf{v}'_k$ are the two neighboring vertices adjacent to $\mathbf{v}'_i$.

### 4.4.4 Edge Flipping Constraints

Although edge flipping can only be found for edges that belong to less important blocks, it is still not desirable. We detect edge flipping by computing the angle formed by the deformed edge and the original one. If the angle is larger than 90°, we consider the edge as flipped. Note that a flipped edge indicates that it has a negative scaling factor. Therefore, we enforce the flipped edge to be aligned with its original direction, but with a very small scaling factor, by adding the following energy term

$$D_{\mathbf{e}_{ij}} = \alpha \|\mathbf{e}'_{ij} - \delta \mathbf{e}_{ij}\|^2, \tag{4.7}$$

where $\mathbf{e}_{ij}$ is a flipped edge, $\alpha$ is a large constant to enforce the constraints, and $\delta$ is a small constant to preserve the block from being shrunk to zero size or even being negatively scaled.

### 4.4.5 Volume Boundary Constraints

In order to retain the size and shape of the volume bounding space, we add the boundary constraints to ensure that the vertices in the grid are always placed within the boundary throughout the deformation process. The following equations are the constraints:

$$\begin{cases} \mathbf{v}'_{i,x} = \mathbf{v}_{i,x} & \text{if } \mathbf{v}_{i,x} \text{ is on the } yz \text{ boundary plane,} \\ \mathbf{v}'_{i,y} = \mathbf{v}_{i,y} & \text{if } \mathbf{v}_{i,y} \text{ is on the } xz \text{ boundary plane,} \\ \mathbf{v}'_{i,z} = \mathbf{v}_{i,z} & \text{if } \mathbf{v}_{i,z} \text{ is on the } xy \text{ boundary plane.} \end{cases} \tag{4.8}$$

### 4.4.6 Solving Linear System

The energy function that we would like to minimize is

$$D = \sum_{\mathbf{b}_k \in \mathbf{B}} D_f(\mathbf{b}_k) + w_\ell D_\ell + w_s \sum_{\mathbf{b}_i, \mathbf{b}_j \in \mathbf{B}} D_s(\mathbf{b}_i, \mathbf{b}_j) + \sum_{\mathbf{e}_{ij} \in \mathbf{E}^*} D_{\mathbf{e}_{ij}}, \tag{4.9}$$

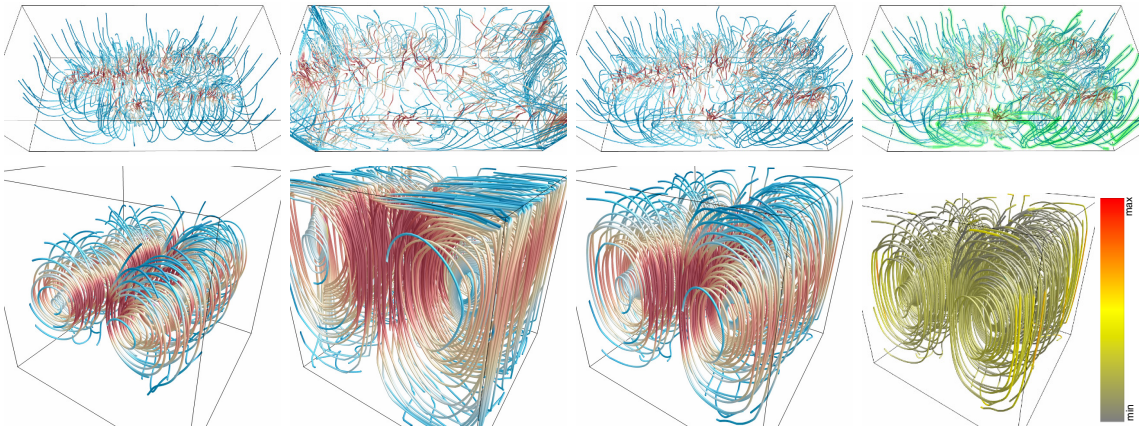
where $\mathbf{b}_i$ and $\mathbf{b}_j$ are neighboring blocks, $\mathbf{E}^*$ is the set of flipped edges, and $w_\ell$ and $w_s$ are parameters to adjust the weights of the two smoothing terms. Each energy term is converted into rows in a linear system, and each dimension of the vertex

coordinates can be solved independently in multiple passes. In each pass, we solve the linear system to obtain a new set of vertex positions $\mathbf{V}'$, and update the scaling factor $s_{\mathbf{b}_k}$ and rotation matrix $\mathbf{R}_{\mathbf{b}_k}$ for each block to better estimate the desired transformation. We use the method described by Horn [41] to calculate the scaling factors and rotation matrices. From the corresponding coordinates of the vertices of one block before and after the deformation, a $4 \times 4$ matrix for that block can be constructed. Then, the rotation matrix is represented by a unit quaternion, which is the eigenvector associated with the most positive eigenvalue of this matrix. To achieve interactive deformation, we leverage a GPU implementation of the concurrent number cruncher (CNC) sparse solver [7] to solve the linear system.

## 4.5 Streamline Repositioning and Error Evaluation

After grid deformation, we reposition the streamlines by computing each point along the line as a linear combination of its corresponding eight block vertices in the deformed grid. To measure block distortion, we transform each original block to have the same size and orientation as the deformed one and compare their difference

$$D_{err}(\mathbf{b}_k) = \sum_{\mathbf{e}_{ij} \in E(\mathbf{b}_k)} \frac{w_{\mathbf{b}_k} \|\mathbf{e}'_{ij} - s_{\mathbf{b}_k} \mathbf{R}_{\mathbf{b}_k} \mathbf{e}_{ij}\|^2}{s^2_{\mathbf{b}_k}}. \tag{4.10}$$




(a) (b) (c) (d)

**Figure 4.4:** F+C visualization results with automatic importance evaluation. First row: the crayfish data set. Second row: the two swirls data set. (a) is the original visualization result. (b) is the naïve deformation that only considers individual block expansion. (c) is the deformation with adding neighboring block smoothing, flow-aware adjustment and flow-aware smoothing. (d) shows deformed streamlines with block errors mapped to colors along the points on each streamline. (© 2014 IEEE. Reprinted by permission.)

Note that Equation (4.10) is the block deformation term (Equation (4.2)) divided by the square of the scale, which normalizes the error for blocks with different sizes. We measure the distortion whenever we update the scaling factor and stop the magnification when increasing the scaling factor does not further magnify the flow features.

Since distortion are inevitable in the deformation process, it will be helpful to inform the users *where* the distortion exists and *how* severe the distortion each block suffers, so that they will not be misled by some abnormal patterns created. We visualize the distortion with two methods. One method is to map error values to streamline colors, as shown in the second row of Figure 4.4 (d). With this method, error values can be better revealed while the information (e.g., velocity magnitude) about the

flow field shown by the original color map is lost. The other method is to draw green semitransparent tubes over the streamlines with higher opacity values indicating larger distortion. With this method, the error values only provide a rough idea about which regions suffer larger distortion. In the first row of Figure 4.4 (d), we observe that the grids with larger distortion are mostly located around the volume boundary. The first method provides more accurate information of the distortion value, while the second method is better to be used as an additional visual hint.

## 4.6 Results and Discussion

To demonstrate the effectiveness of our method, we experiment our method with multiple data set and present both a qualitative analysis on the visualization results (Section 4.6.2), and a quantitative comparison with the traditional fisheye view (Section 4.6.3). The performance and parameter settings are discussed as well.

### 4.6.1 Performance and Parameter Settings

Table 5.1 shows the data sets we experimented with and the timing results for the block importance evaluation, flow-aware adjustment, block deformation and streamline repositioning steps. The timing was collected on a PC with an Intel Core i7-960

CPU running at 3.2GHz, 24GB main memory, and an nVidia GeForce GTX 580 graphics card with 1.5 GB graphics memory. As expected, the deformation time was proportional to the grid dimension, since a grid with larger resolution would have more vertex positions to solve which means the linear system has more variables and equations. Since our framework is designed to visualize the entire flow field in a F+C manner, the grid resolution is not supposed to grow significantly even with larger data sets. Nevertheless, the time cost for streamline repositioning appeared to be similar. Although the total number of streamline points for the supernova data set is much larger than the others, the repositioning time was only slightly longer. Our implementation utilized the CUDA OpenGL interoperability so that the repositioning was performed in parallel on the GPU and there was no need to transfer the rendering data between main memory and graphics memory. For block importance evaluation and flow-aware adjustment, although they took a longer time for the supernova data set, there was no clear pattern in timing among the grid resolution, number of streamlines, and total number of points on streamlines at the scales of other data sets we explored. The timing was dominated by the deformation time. Except for the large supernova data set, the overall time to update the F+C visualization results was less than 0.5 second, which makes our deformation approach interactive.

The parameters used include a user-defined scaling factor $s_f$, a user-specified weight $\beta$ in the flow-aware adjustment (Equation (4.1)), and the two weighting factors $w_\ell$ and $w_s$ for smoothing the energy terms (Equation (4.9)). In our experiments, we

**Table 4.1**
The flow data sets and their timing results. We run ten iterations for the deformation step. All the timing results are calculated by averaging the results gathered from 100 runs. The evaluation time is for block importance evaluation and the adjustment time is for flow-aware adjustment. The evaluation, adjustment, and repositioning time are measured in milliseconds, and the deformation time is measured in seconds.

| data set | grid dimension | eval. time | adj. time | def. time | # lines | # pts. per line | rep. time |
|---|---|---|---|---|---|---|---|
| five critical points | $10 \times 10 \times 10$ | 0.77 | 8.99 | 0.23 | 140 | 58.8 | 0.16 |
| tornado | $12 \times 12 \times 12$ | 1.26 | 12.08 | 0.31 | 60 | 365.4 | 0.17 |
| two swirls | $12 \times 12 \times 12$ | 1.46 | 11.56 | 0.30 | 100 | 236.7 | 0.18 |
| electro | $12 \times 12 \times 12$ | 1.07 | 8.35 | 0.29 | 200 | 52.6 | 0.16 |
| car flow | $36 \times 23 \times 5$ | 2.10 | 11.84 | 0.49 | 140 | 198.7 | 0.18 |
| crayfish | $21 \times 10 \times 7$ | 1.73 | 8.82 | 0.28 | 100 | 248.8 | 0.16 |
| computer room | $27 \times 22 \times 3$ | 2.57 | 9.64 | 0.28 | 200 | 182.7 | 0.17 |
| hurricane | $24 \times 24 \times 4$ | 2.71 | 11.82 | 0.36 | 140 | 346.7 | 0.17 |
| supernova | $20 \times 20 \times 20$ | 9.52 | 23.76 | 0.90 | 200 | 692.4 | 0.24 |

usually used a fixed scaling factor $s_f = 5.0$ and set $\beta = 0.1$, $w_\ell = 2.5$, $w_s = 3.0$. The scaling factor could be a bit larger than the actual scaling that can be achieved, since the blocks would stop growing up to a certain degree. Normally, we found that $\beta = 0.1$ was appropriate for most cases, and $\beta = 0.04$ for a few cases where neighboring blocks sharing many streamlines are of different sizes. For the Laplacian smoothing, we might increase $w_\ell$ to 3.0, if obvious size change between neighboring blocks can be found. We might decrease $w_\ell$ to 2.0 to obtain a larger scaling when the change in size was already smooth. For the flow-aware smoothing, $w_s = 3.0$ was good for most cases. However, if many streamlines were distorted to polylines, we would increase $w_s$ to 6.0. All visualization results we present in this section were generated with these parameter settings.

### 4.6.2 Focus+Context Visualization Results

We experiment multiple data sets using both the automatic importance evaluation and the three types of focuses. Snapshots from streamline animation are also captured, where the focal points move along a selected streamline and a user-drawn path, respectively. The image results are presented for a qualitative discussion.

#### 4.6.2.1 Automatic Importance Evaluation

Figure 4.4 on page 87 shows the F+C visualization results with automatic importance evaluation. As shown in (b), without adding any smoothing term, the regions that are evaluated as the more important ones occupy most of the space, while less important ones are squeezed into thin layers which creates serious distortion. By adding neighboring block smoothing, flow-aware adjustment and flow-aware smoothing terms, we observe from (c), that important regions are still well magnified and the volumes are almost filled with streamlines everywhere. Meanwhile, perceptually, those less important streamlines suffer from less distortion. Error results shown in (d) also demonstrate that important regions almost keep their original shapes and less important ones are not seriously distorted either. Although the measured block distortion seems to be large for the two swirls data set, the swirl patterns are still

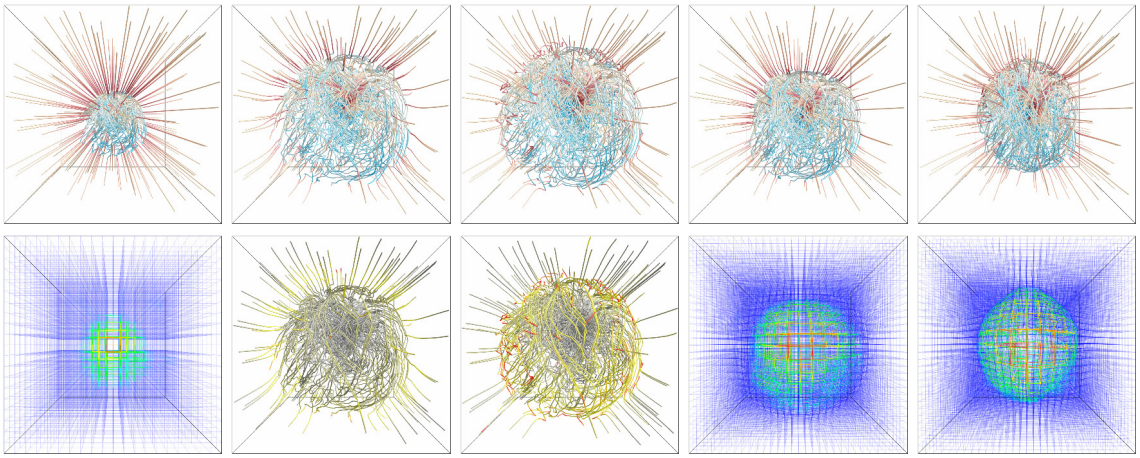
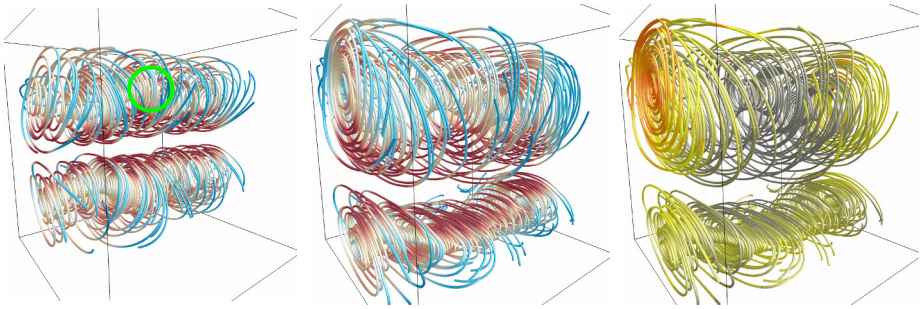



**Figure 4.5:** F+C visualization results with automatic importance evaluation for different scaling factors $s_f$ and grid resolutions. (a) is the original visualization result of the supernova data set. (b), (c), (d) and (e) are deformation results produced with $s_f$ and grid resolution as 5.0 and $20 \times 20 \times 20$, 10.0 and $20 \times 20 \times 20$, 5.0 and $30 \times 30 \times 30$, and 10.0 and $30 \times 30 \times 30$, respectively. The second row shows the corresponding grid, block distortions mapped to streamline colors, and deformed grids. (© 2014 IEEE. Reprinted by permission.)

clear in the deformation result. This is due to the fact that the distortion mainly comes from block stretching, which only slightly changes the perceived shape of the streamlines.

In Figure 4.5, we compare the F+C results with different grid resolutions and scaling factors using the supernova data set. Compared with using $30 \times 30 \times 30$ grid resolution, the important region at the center expands more using $20 \times 20 \times 20$ grid resolution. This holds for both cases under different scaling factors: 5.0 and 10.0. Even though less important blocks would shrink due to the individual block expansion term, the smoothing terms still maintain the shape of those blocks to some degree. With a higher grid resolution, the number of less important blocks surrounding the important



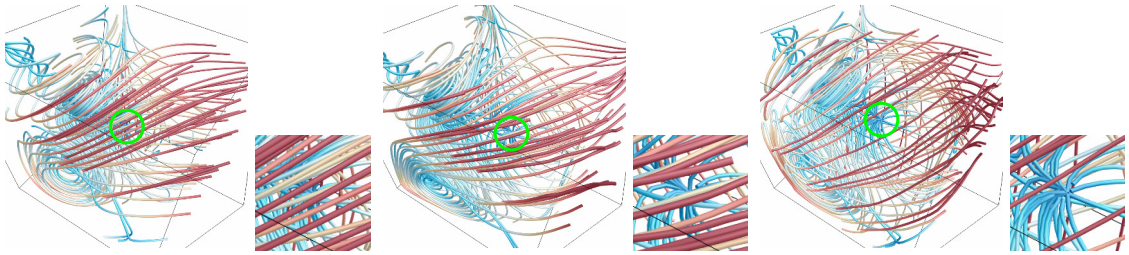
(a) (b) (c)

**Figure 4.6:** F+C visualization results with the user-specified spherical block focus for the two swirls data set . (a) the original streamlines. (b) the deformed streamlines. (c) the deformed streamlines with block errors mapped to colors along the points on each streamline. (© 2014 IEEE. Reprinted by permission.)

region becomes larger, which increases the resistance and prevents important blocks from growing further. Meanwhile, using a scaling factor of 10.0 does not further magnify the important region, because the smoothing terms stop less important blocks from being further squeezed. From deformed grids and evaluated error results, we observe that using a scaling factor of 10.0 shrinks the blocks around the boundary of the important regions, leading to larger distortion. Therefore, in practice, we need to carefully select the appropriate grid resolution and scaling factor instead of simply aiming for higher grid resolutions and larger scaling factors.

#### 4.6.2.2 Spherical Block Focus

In Figure 4.6, we show the F+C visualization results with the user-specified spherical block focus. For the two swirls data set, the focus is at the center of the upper



(a) (b) (c)

**Figure 4.7:** F+C visualization results that reveal hidden features for the five critical points. The user-specified regions are highlighted in the green circle, together with the enlarged images to the right side of each visualization result. (a) shows the original streamlines. (b) shows the deformed streamlines with the spherical block focus. (c) shows the deformed streamlines with the hourglass block focus. (© 2014 IEEE. Reprinted by permission.)

swirl. Before the deformation, the two swirls occupy similar space in the volume. By applying the spherical block focus on the upper swirl, the focal region grows, pushing the lower swirl and the two ends of the upper one to the boundary. From the error image, we observe that the focal region does not suffer much distortion, as the distortion is mainly distributed to the squeezed regions. The five critical points data set also shows a similar result, where the focus is the spiral located at the upper right conner. After the deformation, that spiral is magnified and shifted closer to the center, while the other regions shrink and absorb most of the distortion.

#### 4.6.2.3 Hourglass Block Focus

Figure 4.7 demonstrates the effectiveness of our hourglass block focus. For the five critical points data set, the source located at the center of the volume is occluded by some less interesting streamlines with a similar pattern, as shown in Figure 4.7

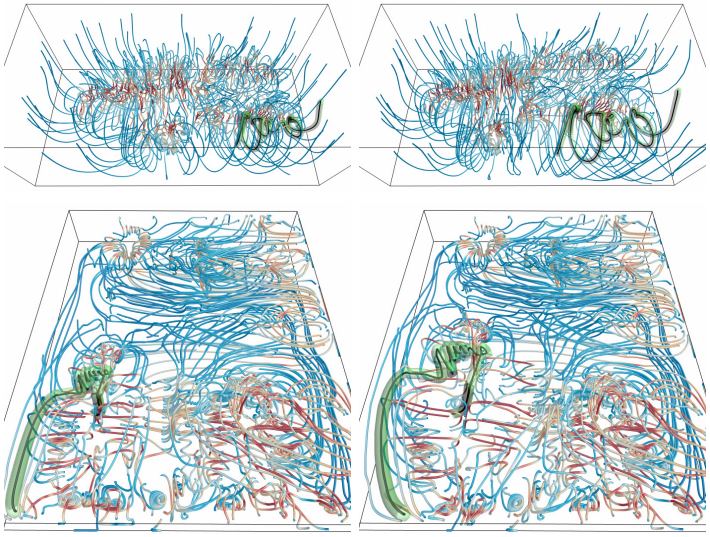
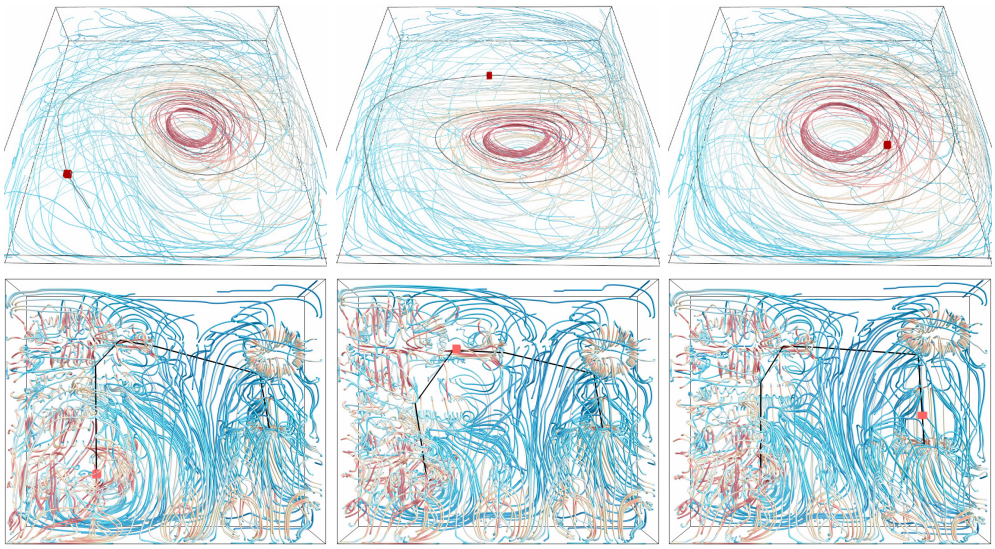


**Figure 4.8:** F+C visualization results with the user-specified streamline focus for the crayfish (top) and computer room (bottom) data sets, where the focal streamline is highlighted in black and surrounded by green semi-transparent tubes. (© 2014 IEEE. Reprinted by permission.)

(a). After the spherical block focus is applied, the source still cannot be observed clearly as shown in Figure 4.7 (b), although the source itself has been magnified. This is because the density of streamlines in the front does not change significantly. The hourglass block focus is used to specify the same source as the focus. In Figure 4.7 (c), the source becomes clearly visible since the streamlines at the outer ring are much sparser. A similar result can be found for the electro data set as the streamlines occluding the source become sparser with the hourglass block focus.



(a) (b) (c)

**Figure 4.9:** Snapshots of F+C animation results. The first row shows a user-specified streamline for the hurricane data set and the second row shows a user-drawn path for the computer room data set. The focal streamline/-path is highlighted in black and the current focal point is marked with a red cube. (© 2014 IEEE. Reprinted by permission.)

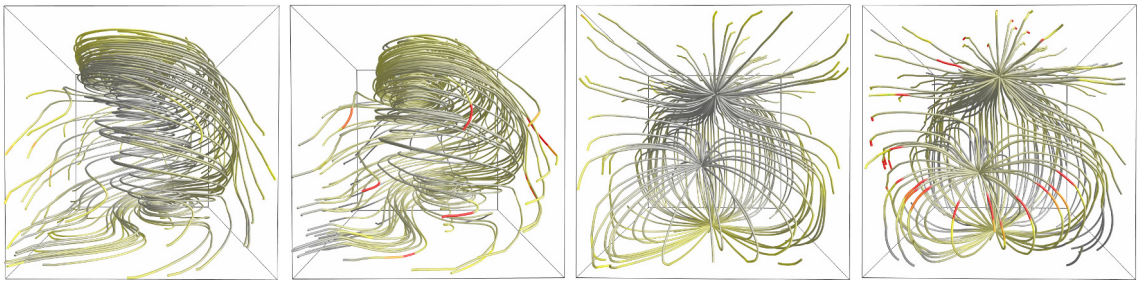
#### 4.6.2.4 Streamline Focus

In Figure 4.8, we show the streamline focus results. Unlike the block focus, the streamline focus treats all the blocks into two different categories: the blocks that the focal streamline does and does not pass through, respectively. Blocks in the same category are assigned the same importance value, so that the transformation for the blocks that contain the focal streamline will be similar and the distortion will be distributed to the rest of blocks more evenly. From the visualization results for both data sets, we see that the shape of the focal streamline is almost the same as the original one after the deformation, while no obvious change can be observed for other

surrounding areas. This implies that the streamlines in the context are more stable and the relationships between the focal streamline and the rest of streamlines are easier to interpret.

#### 4.6.2.5 Streamline Animation

In Figure 4.9, we show selected snapshots of streamline animation results where the focal point moves along a user-specified streamline or a user-drawn path. As shown in the first row of Figure 4.9, the animation helps the user explore the regions that a streamline passes through. This is different from our previous approach that magnifies the entire streamline simultaneously. Since the space in the volume is limited, enlarging multiple regions could either decrease the scaling factor that can be achieved or result in more serious distortion. Using the animation to move the focal point will be more efficient to magnify the regions consecutively. As such, we should select those long streamlines that pass through different regions over short ones. The second row of Figure 4.9 shows another example where we explore F+C animation using the user-drawn path.



(a) (b) (c) (d)

**Figure 4.10:** Mapping block errors to streamline colors for the tornado and electro data sets. (a) and (c) are with our spherical block focus. (b) and (d) are with the fisheye focus. (© 2014 IEEE. Reprinted by permission.)

### 4.6.3 Comparison with Fisheye View

For comparison, we implemented the fisheye view F+C technique presented by Sarkar and Brown [89]. For each vertex within the user-specified focal region with radius $r_{focus}$, we transform the vertex based on the polar coordinate system originated at the center of focus. This maps a vertex with the original coordinates $(r_{ori}, \theta, \gamma)$ to the fisheye coordinates $(r_{feye}, \theta, \gamma)$, where $r_{feye}$ is given by

$$r_{feye} = r_{focus} \frac{(d+1)\frac{r_{ori}}{r_{focus}}}{d\frac{r_{ori}}{r_{focus}} + 1} = r_{focus} \frac{d+1}{d + \frac{r_{focus}}{r_{ori}}}. \tag{4.11}$$

Here, $d$ is a constant distortion factor and a larger value of $d$ results in a higher degree of magnification. In this paper, we set $d = 3.0$.

In Table 5.2 and Figure 4.10, we show quantitative and qualitative results comparing

**Table 4.2**
Block distortion evaluated using Equation (4.10). The blocks are grouped by the distances (in voxels) from their center to the focus.

| dist. to focus | # blocks | spherical focus | | fisheye focus | |
|---|---|---|---|---|---|
| | | avg. | max | avg. | max |
| tornado | | | | | |
| $< 10$ | 27 | 1.641 | 2.665 | 13.572 | 15.987 |
| $< 20$ | 251 | 17.145 | 37.969 | 88.950 | 843.750 |
| $< 40$ | 1283 | 49.445 | 505.412 | 66.443 | 843.750 |
| all | 1728 | 49.434 | 505.412 | 49.333 | 843.750 |
| electro | | | | | |
| $< 10$ | 27 | 2.332 | 4.219 | 18.382 | 43.927 |
| $< 20$ | 230 | 19.749 | 55.511 | 95.026 | 844.575 |
| $< 40$ | 1283 | 42.603 | 191.895 | 64.261 | 844.575 |
| all | 1728 | 40.833 | 434.848 | 42.841 | 844.575 |

block distortions for our spherical grid focus and the fisheye focus. For the tornado data set, the focus is set at the center of the volume. We observe from Figure 4.10 that the focal region has much less distortion using our method. For the surrounding regions, although both methods lead to some distortion, our method does not gather the distortion around the boundary of the focus. The measured errors shown in Table 5.2 are consistent with the image results. The radius of focal region is 20 voxels for our spherical focus, and 35 voxels for the fisheye focus to achieve a similar scaling effect. The overall errors are close for both methods, but our method has much smaller average and maximum errors near the focus. For our spherical block focus, the errors are mainly distributed to the blocks that are at least 20 voxels away. However, for the fisheye focus, the blocks located outside of the focal region will not deform at all. All the errors are accumulated within the focal region, especially for the blocks around the boundary of the focal region, which results in the undesired ringing artifact.

## 4.7 Evaluation

To evaluate our deformation framework, we collaborated with a domain expert in fluid mechanics (Professor Seung Hyun Kim) for an empirical expert evaluation. His research focuses on the modeling of multiscale and multiphysics problems in relation to energy science and technology. In addition, we performed a user study involving five graduate student with fluid mechanics background. They were required to complete six tasks and provide comments based on six criteria.

### 4.7.1 Empirical Expert Evaluation

After learning the framework and using our program multiple times with various data sets, Dr. Kim provided his feedback. We organize and present his feedback as the following. In general, the use of deformation for F+C visualization in flow field exploration is novel and effective. Having multiple methods developed for users to select the focus is a significant advantage. This allows users to determine the best method in their respective cases or even apply multiple methods in a certain order to achieve more desired results. In terms of distortion, both the spherical focus and streamline focus provide better F+C visualization effects than the fisheye view. When the interesting region is at the corner, focusing on a streamline in that region might

be more effective than the spherical focus, since the spherical region might go out of bound and lead to inevitable distortion on the boundary. In many cases, he found it very useful to explore the data sets with multiple focus selection methods. For example, the users could first use the streamline focus to enlarge the regions that a streamline goes through. Based on the deformation result, the users might be able to find some small features in those regions and apply the spherical focus to further enlarge the small features. It is also beneficial to use the hourglass focus on the features that are hidden in the central region of volume, e.g., the source around the center in the five critical points data set. That is, the users can use the hourglass focus to push away the streamlines that occlude the interesting features, and then apply the streamline focus or spherical focus to further explore that region of interest.

With the GPU implementation, our program is fairly interactive which allows the users to fine tune the parameters to achieve satisfactory deformation results on the fly. According to the suggestions of Dr. Kim, we also modified our initial single view interface to support multiple views, which benefits parameter tuning by eliminating the need to switching between different views back and forth. We allow the users to freely select any three items for simultaneous display. Our experience shows that visualizing the deformed streamlines, original streamlines and deformed grid simultaneously is particularly useful. The connection among these three views is helpful for the users to determine the actual scaling obtained in focused regions and fine tune the distortions accordingly. Even if some distortions are inevitable, they could

be easily identified under multiple views and with error indications (introduced in Section 4.5). In addition, as the users get familiar with each parameter, they can predict the changes due to parameter tuning accordingly.

Since one of the main current areas of interest of the expert is the modeling of turbulent combustion, he further commented that the application of this deformation framework to flame visualization could be valuable for investigation. The combustion reactions can be confined into a relatively thin region and be substantially influenced by a flow field, strain or vorticity. Using our deformation framework to emphasize the species concentrations or temperature in the region of high strain rate or vorticity would provide very useful information. The current deformation framework could also be extended to be of further use in two aspects. First, in addition to the vector field, the deformed blocks could also guide the deformation of scalar fields for a mix rendering to provide more context information or apply to F+C visualization of time-varying data sets. Second, a diverse choice for automatic evaluation could be applied to enlarge the regions with any other desired properties, e.g., high vorticity.

### 4.7.2 User Study

We also recruited five unpaid researchers for a user study: two postdoctoral scholars, one PhD student, and two master students. All of them are researchers majoring

in mechanical engineering with at least one year of research experience in fluid dynamics. The user study was conducted in a lab using the same PC. The PC has a 27-inch monitor with $1920 \times 1080$ resolution, where the visualization result occupied an area of $1200 \times 800$. The users were first introduced to the concepts of automatic importance evaluation, spherical block focus, hourglass block focus, streamline focus, and error indication. Then they were given the crayfish data set for free exploration to get familiar with the system. They could perform the tasks whenever they felt comfortable. Each study took about two hours to two and a half hours to complete. Although each task could be performed in a few minutes, the users frequently returned to the interface for further verification when writing their comments, which occupied most of the time.

We designed six tasks (**T1** to **T6**). **T1** and **T2** asked the user to select a deformation method and the viewing direction to best observe the source for the five critical points data set and the flow pattern at the center for the supernova data set, respectively. **T3** and **T4** asked the user to evaluate the distortion given a deformation result using the crayfish and two swirls data sets, respectively. **T5** and **T6** asked the user to select a deformation method and reproduce the deformation result given an image of deformation result using the car flow and computer room data sets, respectively. The users were informed that the tasks were not timed and their comments were of crucial importance.

Six criteria (**C1** to **C6**) were given as guidelines for users to evaluate the deformation framework: **C1** effectiveness to magnify features; **C2** effectiveness to reduce the occlusion over a feature; **C3** ease to notice the distortion; **C4** ease to understand the relationship between deformed and original streamlines; **C5** ease to estimate the original pattern from the deformed one; and **C6** ease to reproduce a deformed visualization result. We asked the users to comment on the effectiveness of the proposed F+C techniques for **T1**, **T2**, **T5** and **T6**, the distortion evaluation for **T3** and **T4**, and the ease of reproducing a deformation result for **T5** and **T6**. Finally, a set of open questions were also presented which ask the users to provide general impression of the methods for each of the criteria. The deformation result for **T3** and **T4** can be found in Figure 4.4 (d). The images presented to the users for **T5** and **T6** are the bottom right image in Figure 4.8 and Figure 1.7 (c), respectively.

In terms of **C1**, the feedback was very positive. Yet the selection of methods varied due to the different foci between disciplines or personal preference. The only exception was **T6**, where the deformation result to be reproduced was apparently generated using streamline focus. Other than that, each user selected the deformation method to complete the tasks in a consistent way. User 1 used a combination of automatic importance evaluation and spherical block focus for **T1**, **T2**, and **T5**. He commented that *"automatic importance evaluation and spherical block focus combined could provide more magnifying features compared to using any single choice"*. He would finally select spherical block focus to produce the result, since *"spherical block focus does it*

*better because the feature is within a local area"*. User 2 also preferred a combination of multiple methods. He applied automatic importance evaluation and then hourglass block focus for **T1**, because of *"secluding the source that is obscured by other streamlines"*. For **T2** and **T5**, he used automatic importance evaluation to gain an overview and spherical block focus to magnify a specific region. The other three users would like to apply a single method to perform the task, User 3 preferred hourglass block focus and User 4 and User 5 mostly used spherical block focus. User 5 also used streamline focus for **T2**, which asked the users to observe the pattern of the supernova data set. The central region was complex and difficult to understand even if it was magnified. He said that the streamline focus *"allowed to better understand the features"*, since most streamlines shared a similar pattern and the analysis should start from understanding one of them.

Although the selections differed among the users, their comments on each method were similar. All of them rated spherical block focus to be the most effective one to magnify features, followed by automatic importance evaluation and streamline focus. They indicated that *"automatic importance evaluation method could find the region of importance"*, but spherical block focus was better, since *"user interaction is involved"* and *"in most fluid flow problems, we are interested in a region in space"*. Spherical block focus was favored over streamline focus mostly because streamline focus only had *"limited ability to magnify the adjacent fluid flow"*. But a user also mentioned that this might be discipline dependent, since *"if the particle tracing is*

*the concern, maybe the streamline focus could do a better job".* Hourglass block focus was considered to be the least effective one, since it did not magnify a feature.

In terms of **C2**, most users selected spherical block focus or hourglass block focus. Although spherical block focus was not designed to reduce the occlusion, users found that when the influence region was large enough, the outer layer could still get sparser. This would definitely sacrifice the quality of context streamlines, but a user stated that he *"chose to focus more in the point"* rather than *"worry much about the distortion on the boundary".* In our observation, the users usually used automatic importance evaluation to gain an overall impression of the field, and applied spherical or hourglass block focus for further analysis, since the interaction allow them to specify a region for detail observation.

The error estimation corresponds to our criteria **C3**, **C4**, and **C5**. All users agreed that with error indication, it was easy to notice the distortion. A user even mentioned that he could notice the distortion by just moving the scale slider, and *"the error indication helped me to quantify".* The ease of understanding the relationship between deformed and original streamlines depended on the complexity of the deformation. For the deformation result using the two swirls data set, a user stated that *"if the deformed style is a regular shape, it is easy to relate to the original streamlines, even when the fluid flow is complex".* Most users believed it was easy to understand this relationship by moving the scale slider and observing the deformation process, even for

relatively complex deformation. But a user also mentioned that the absence of error indication with a small scale value (close to 1.0) *"makes tracking all the distorted streamlines impossible"*, although he admitted that *"it is able and relative easy to track just one streamline"*. Once the relationship between deformed and original streamlines was built, the estimation of original shape of streamlines would not be an issue. Only one user mentioned that *"it is able to make relatively rough estimations, but not into details"*. This is acceptable since the distorted streamlines are mostly in the context.

In terms of **C6**, all users stated it was easy to produce a similar result with comments such as *"it is not difficult to reproduce the results, since the feature of each function is clear and easily recognized"*. The selection of parameters seemed not to be a problem. A user mentioned that *"it may take some time to select the correct cell, and select the correct parameters, but it can be done within a short time"*. Two users also commented that it might be more difficult to reproduce a result generated with hourglass block focus. This was probably because hourglass block focus is view-dependent and has more parameters to adjust.

# Chapter 5

# FlowString: Partial Streamline Matching Using Shape Invariant Similarity Measure for Exploratory Flow Visualization

## 5.1 Terminology and Notation

Before describing the overview of our algorithm[1] , we first define the following terms that will be frequently used in this section:

[1]The material contained in this chapter was published in *Proceedings of IEEE Pacific Visualization Symposium 2014* [104] and *IEEE Transactions on Visualization and Computer Graphics 2016* [106].

- *Character:* A character is a unique local shape primitive extracted from streamlines which is invariant to its geometric position and orientation. Characters are the low-level feature descriptors for categorizing different streamline shape features.

- *Alphabet:* The alphabet consists of a set of characters that describe various local shape features of streamlines traced from a given flow data set.

- *Word:* A word is a sequence of characters expressing a meaningful streamline shape pattern. Words are the high-level feature descriptors for differentiating regional streamline shape features.

- *Vocabulary:* The vocabulary consists of a set of words describe various regional shape features of streamlines traced from a given flow data set.

- *String/Substring:* A string is the mapping of a global streamline to a sequence of characters. A substring encodes a portion of the corresponding streamline. A substring could match with a word in the vocabulary.

The notations for string operation are mostly consistent with the convention. However, some minor changes are also introduced to adapt to this specific context, which are listed as follows:

- *Character:* The shape primitive represented by a character is formed by an ordered set of points. A character is denoted as a single lowercase letter `a` if

the sample points on the streamline to be matched is in the same order of the shape primitive. It is denoted as a single lowercase letter followed by a prime symbol `a'` if the sample points is in the reversed order of the shape primitive. We use the uppercase letter `A` to indicate that the sample points could be in both directions.

- *Multiple characters with common features:* "`|`" This notation denotes multiple characters that share some common properties, e.g., $(\mathtt{a}_1|\mathtt{a}_2|\ldots|\mathtt{a}_l)$ denotes a local shape represented by any one character appearing in the parenthesis.

- *Single character repetition* "$^+$"*:* The repetition of a single character usually indicates the appearance of a pattern formed by repeating a local shape, e.g., spirals. In this case, the repetition number mainly depends on the length of the corresponding segment, which does not change human perception. For example, if character `a` represents a circle, $\mathtt{a}^+$ will match any spiral which corresponds to the concatenation of a number of `a`.

- *Wildcard characters* "`?`" *and* "`*`"*:* We enable querying using wildcard characters: the question mark `?` for substituting zero or one character and the asterisk `*` for zero or more characters. The wildcard characters are usually used to connect two patterns. The asterisk could be useful to search for any streamline that contains both patterns, while the question mark could further constrain the distance between the two patterns.

- *Word concatenation* "`|`" *and* "`&`"*:* We use two shortcuts `|` (`or`) and `&` (`and`) to concatenate two words with the square brackets "`[ ]`" for distinguishing word boundaries. For example, `[aaa]|[bbb]` returns the segments that matches either `aaa` or `bbb`, while `[aaa]&[bbb]` finds the segments that contain both `aaa` and `bbb` within some distance apart.

## 5.2 Overview

Our FlowString algorithm consists of two major components: *alphabet generation* and *string operation*. Alphabet generation is to generate the alphabet that describes unique local shape features of streamlines traced from a given flow data set. String operation refers to the matching, querying and pattern recognition of the strings based on this alphabet. We consider each character in the alphabet as a local shape descriptor, which is invariant to its geometric position and orientation. Concatenating the characters defines unique shape features in a larger scope. In order to be invariant to the local feature size, we first resample the streamlines, so that the number of sample points will be similar for the local features with the same shape but different scales. For each sample point, its local shape will be represented by a set of sample points in its neighborhood with a size of $r$, i.e., the sample point itself and the $(r-1)/2$ nearest neighbors in both the forward and backward directions along the streamline. The dissimilarity between the local shapes of any two sample points is

given by the *Procrustes distance* between the two sample point sets centering on these two points. Using the pairwise dissimilarity as the distance matrix, we apply *affinity propagation* to perform a two-level bottom-up clustering on these local shapes and treat each cluster at the higher level as a character. Each sample point will be assigned the character corresponding to the cluster in which it resides. With this treatment, we can consider the streamlines as *strings* that are concatenation of characters assigned for each of their sample points. A *suffix tree* is constructed to represent all the strings to enable efficient search and pattern recognition. In the following, alphabet generation is discussed in 5.3, string operation is discussed in 5.4, the user interface and interactions are introduced in 5.6, and some additional considerations are discussed in 5.5. Finally, we present the visualization results in Section 5.7 and findings in an empirical expert evaluation 5.8.

## 5.3 Alphabet Generation

Alphabet generation is performed in three steps: streamline resampling to generate the sample points; dissimilarity measure to evaluate the difference of local shapes between every pair of sample points; and affinity propagation clustering to group the sample points according to the dissimilarity matrix.

### 5.3.1 Streamline Resampling

We first resample the streamlines, so that the number of sample points is similar for the local features with the same shape but different scales. For each sample point, its local shape is represented by a set of sample points in its neighborhood with a size of $r$, i.e., the sample point itself and the $(r-1)/2$ nearest neighbors in both the forward and backward directions along the streamline. Our streamline resampling should meet two crucial requirements. First, a streamline segment between two sample points should be simple enough, so that no feature is ignored due to under-sampling. Second, since we use a neighborhood of size $r$ to represent the local shape, the density of sample points should be related to the local feature size. That is, for a meaningful comparison, the local features with the same shape should contain the same number of sample points.

Let us consider a continuous 3D curve $C$ and another curve $C'$ which is the result of uniformly scaling $C$ by a factor $s$. Let $p_1$ and $p_2$ be two points on $C$, and $p'_1$ and $p'_2$ be two points on $C'$ which correspond to $p_1$ and $p_2$, respectively. The curvature $\kappa'$ of each point on $C'$ is $\kappa/s$, where $\kappa$ is the curvature of the corresponding point on $C$. Since the arc length $l'$ between $p'_1$ and $p'_2$ is $s \times l$, where $l$ is the arc length between $p_1$ and $p_2$, the *accumulative curvature* between $p'_1$ and $p'_2$ is the same as that between $p_1$ and $p_2$. This implies that keeping a constant accumulative curvature between two

neighboring sample points will produce similar resampling for features with the same shape but of different scales.

For a streamline which is often represented as a polyline, the curvature is not immediately available. In our approach, the discrete curvature at a point could be approximated by the angle between its two neighboring line segment, and the accumulative curvature becomes the *winding angle* of a streamline segment. Although the winding angle might be affected by the density of points along a polyline, it is very stable if the points traced along the streamline are dense enough.

Our resampling starts from selecting one end of a streamline as the first sample point, and iterates over the other traced points along the streamline. During the iterations, we accumulate the winding angle from the last sample point to the current point. Once the winding angle is larger than a given threshold $\alpha$, the current point is saved as a new sample point and the winding angle is reset to zero. Note that the neighborhood size $r$ is closely related to the selection of $\alpha$. That is, when $\alpha$ is smaller, $r$ should be larger to cover the same range of the streamlines in order to capture the shape of local features. In our experiments, we find that setting $\alpha = 1$ (in radian) and $r = 7$ works well for all our test cases. This is because when $\alpha = 1$, the pattern of a streamline segment between two neighboring sample points is relatively simple, and seven consecutive points cover mostly the range of a circle, which is enough to describe a local shape and yet not too complex. In Figure 5.1, we show an example

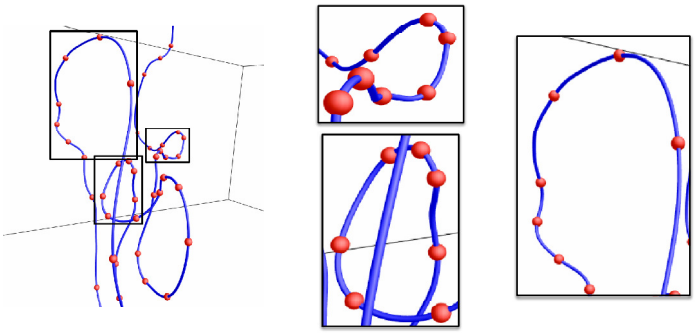
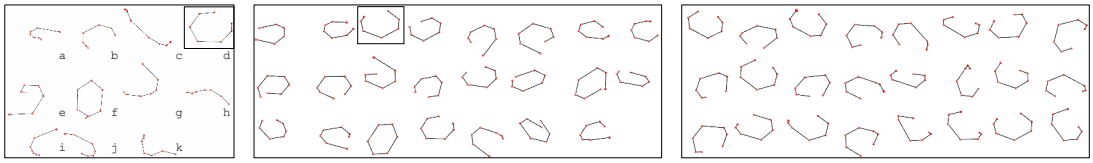


**Figure 5.1:** Resampling a streamline traced from the crayfish data set. The red dots are resampled points. Three regions selected are highlighted in the right. (© 2014 IEEE. Reprinted by permission.)

of our resampling result. The three highlighted regions are with three different local scales, which all contain a similar number of sample points after resampling.

### 5.3.2 Dissimilarity Measure

We compute the dissimilarity between the local shapes of two sample points as the *Procrustes distance* between their neighborhoods, where each neighborhood is a sample point set of size $r$. This distance only considers the shape of objects and ignores their geometric positions and orientations. The two point sets must first be superimposed before shape comparison, which calls for a registration to obtain the optimal translation, rotation and uniform scaling. This registration is often referred to as the *Procrustes superimposition*. After the superimposition, the two point sets representing the same shape will exactly coincide and thus have the distance of zero. The optimal translation $\mathbf{T}$, rotation $\mathbf{R}$, and uniform scaling $s$ from one point set




(a) (b) (c)

**Figure 5.2:** Characters generated from a two-level bottom-up affinity propagation clustering of the crayfish data set. (a) shows the 11 high-level cluster centers, which are assigned to characters `a` to `k` in order. (b) shows the 23 members in the cluster highlighted with a box in (a), which are low-level cluster centers. (c) shows the 24 members in the cluster highlighted with a box in (b). (© 2014 IEEE. Reprinted by permission.)

$P_a = \{p_{a1}, p_{a2}, \ldots, p_{ar}\}$ to another point set $P_b = \{p_{b1}, p_{b2}, \ldots, p_{br}\}$ are the ones that minimize the summation of the pairwise point distances [41]

$$d = \sum_{i=1}^{r} |p_{bi} - {p'}_{ai}|^2, \text{ where } {p'}_{ai} = s\mathbf{R}p_{ai} + \mathbf{T}. \tag{5.1}$$

Note that the minimized $d$ is the Procrustes distance between $P_a$ and $P_b$. However, in Equation (5.1), we assume that $p_{ai}$ should be paired with $p_{bi}$, which might not always be the case for two streamline segments, since two segments with the same shape might be indexed in the opposite directions. Therefore, instead of accepting $d$ as their dissimilarity between $P_a$ and $P_b$ immediately, we also compute the distance $d'$ with points being paired in a reversed order, and use the minimum of $d$ and $d'$ as the final dissimilarity value.

### 5.3.3 Affinity Propagation Clustering

Given the dissimilarity measure, we compute the pairwise dissimilarity among all sample points and apply *affinity propagation* for clustering. The similarity values are then obtained as the negative of the dissimilarity values, as suggested by Frey and Dueck [29]. Unlike k-means and k-medoids clustering algorithms, affinity propagation simultaneously considers all data points as potential exemplars and automatically determines the best number of clusters, with the preference values for each data point as the only parameters. The preference value indicates the probability of selecting the corresponding data point as a cluster center. Using a uniform preference value indicates that all the data points are considered with an equal chance to be cluster centers, and a smaller preference value (i.e., a more negative value in our case) produces a smaller number of clusters. In our scenario, affinity propagation usually generates a fine level of clustering result (with hundreds of clusters). Therefore, we use the minimum of the similarity values as the preference. Although affinity propagation generates high-quality clusters for all the sample points, it is unnecessary to keep the clusters at such a fine level. To support pattern query and recognition at a coarser level, the cluster centers at the first level are then clustered by applying affinity propagation again to generate the second-level clusters. In our experiments, the second-level cluster indices serve as the characters, and we find that they already have enough discriminating powers. Figure 5.2 shows an example of the clustering

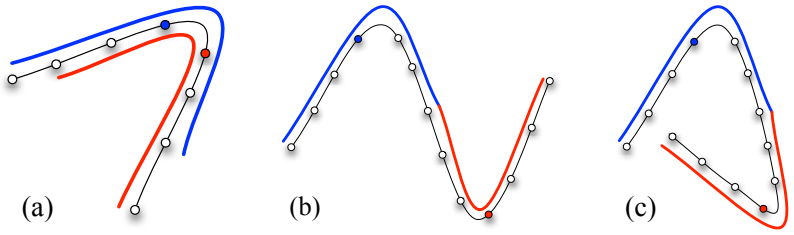



**Figure 5.3:** Character concatenation. The blue and red lines indicate the neighborhoods of blue and red sample points, respectively. (a) characters are assigned to all sample points. $r-1$ sample points are shared by the neighborhoods of blue and red sample points, which produce a deterministic shape. (b) and (c) characters are assigned to every $r-1$ sample points. Only one point is shared by the neighborhoods of blue and red sample points, which could produce different shapes. (© 2014 IEEE. Reprinted by permission.)

results. We see that the members in the same clusters are usually similar to each other.

### 5.3.4 Character Concatenation

In our work, a character corresponding to a sample point determines the local shape of its neighborhood of size $r$. If the characters are assigned to every sample point, a concatenation of two characters represents the shape of a neighborhood of size $r+1$. As shown in Figure 5.3 (a), this shape is uniquely determined by the two characters centering on the blue and red sample points. However, if the characters are only assigned to every $r-1$ points, even if the two characters are exactly the same, the resulting shape of $2r-1$ points could vary significantly. This is because the relative orientation of the two local shapes is undetermined, as shown in Figure 5.3 (b) and

(c). Moreover, since these $r$ points in a neighborhood might not be evenly spaced, the overlapping region of two neighborhoods also decides their relative scale. Also notice that the order of points for each character might affect the shape represented by a string. Certainly, if $r$ is large, we might not need to assign characters to every sample point to maintain the overlapping region of size $r-1$. In practice, since we opt to use a small value for $r$ to avoid too complex local shapes, assigning a character to every sample point seems to be necessary in order to produce deterministic shapes for a string.

## 5.4 String Operation

Our string operation is based on *suffix trees* [110]. In this section, we give a brief introduction of suffix trees, and explain how to construct vocabularies and perform searches on a suffix tree.

### 5.4.1 Streamline Suffix Tree

After we convert each streamline to a string, we construct a suffix tree in linear time and space to enable efficient operations on these strings. A suffix tree is a special kind of tree that presents all the suffixes of the given strings. Each edge of the suffix

tree is labeled with a substring in the given strings. For a path starting from the root to any of the leaf nodes, the concatenation of these substrings along this path is a suffix of the given strings.

The problem of search for a string then becomes the search for a node in the suffix tree. Considering that the size of the alphabet is constant, the decision on which edge to visit could be made in constant time, and the search of a string with length $m$ can be performed in $O(m)$ time. Assuming the number of appearance of a string to be searched is $z$, reporting all the positions of that string takes $O(z)$ time. As a result, with the suffix tree, an exact match of a substring that appears in the given string multiple times only takes $O(m + z)$ time.

### 5.4.2 Vocabulary Construction

Given a pool of traced streamlines, one interesting yet challenging problem is to automatically identify meaningful words in these streamlines to construct the vocabulary. Since the words are depicted by a sequence of characters, we need to not only select representative streamlines, but also extract important segments from them for word identification. With our streamline suffix tree, this could be efficiently solved as we select the most common patterns from the streamlines. In other words, streamline segments that appear most frequently could be identified as words.

We implement our approach on the streamline suffix tree by a simple tree traversal scheme. Since the shape of each streamline segment is captured by a substring in our suffix tree, selecting the common patterns of streamline segments could be considered as the detection of the most frequently appeared substrings. Considering that each potential substring is associated with a node in the suffix tree, the number of appearance for a substring can be efficiently counted with the following two cases:

- If the substring corresponds to a leaf node, its number of appearance is the number of position labels attached to that node;
- If the substring corresponds to an internal node, its number of appearance is the summation of the counts for all the children of that node.

This information could be gathered by a traversal of the tree in the depth-first search manner. Then, all substrings with the length and number of appearance larger than certain thresholds could be reported by another tree traversal. Therefore, identifying words to form the vocabulary can be performed in $O(n)$ time, where $n$ is the total length of the original strings, since the number of nodes is linear to $n$.

### 5.4.3 Exact vs. Approximate Search

Since the string is used to represent the shape of streamline segments, exact string matching normally does not provide enough flexibility to capture streamline segments

with similar shapes. First, the similarities among the shapes represented by different characters are different, e.g., a portion of spiral with large torsion is more similar to that with small torsion than other shapes. But, exact match only produces a binary result, which is either the same or different. Second, with respect to human perception, different numbers of repetition of a certain shape often seem to be similar. For instance, a spiral that contains three circles and another one contains five circles are usually considered to be similar. Assuming a shape similar to a circle is represented by character `a`, then strings `aaa` and `aaaaa` should be matched in our search. To enable these approximate searches, we first introduce a straightforward dynamic programming approach to detect $k$-approximate match on the suffix tree, where $k$ is a threshold used in the edit distance. Then, this approach is extended to support the repetition of a single character.

In our scheme, computing the edit distance of two strings $P = P_1P_2 \dots P_{n_p}$ and $T = T_1T_2 \dots T_{n_t}$ is the same as the traditional approach by filling a table DP of size $n_p \times n_t$, where DP$[i, j]$ is the edit distance for substrings $P_1 \dots P_i$ and $T_1 \dots T_j$. Our straightforward $k$-approximate match on a suffix tree traverses the tree in the depth-first search manner and expands the table column by column.

During the traversal, each time we expand the string $T$ by one character along the edge being visited and fill the column DP$[*, n_t]$. If DP$[n_p, n_t]$ is smaller than $k$, then we find a match $T$ whose edit distance to $P$ is within $k$. Note that we only

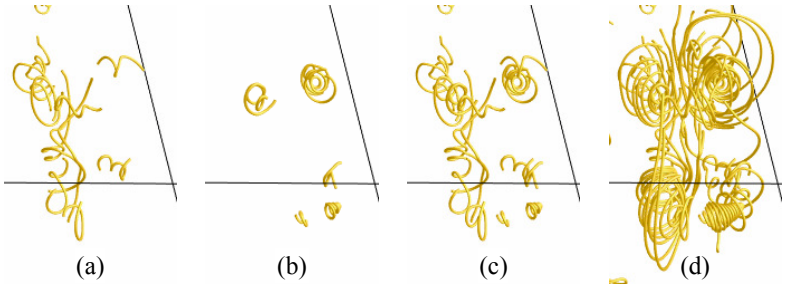
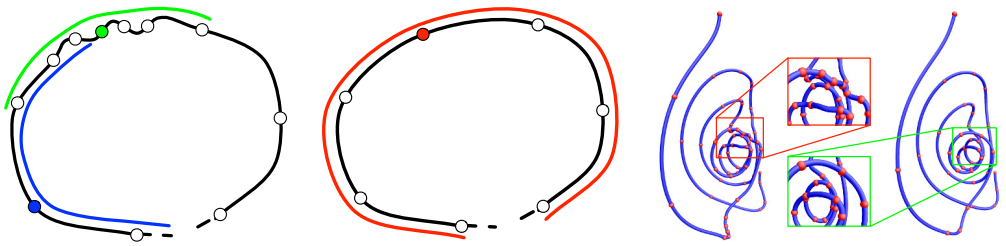



**Figure 5.4:** Matching results using the crayfish data set. A zoomed-in view is used to show a partial volume for clearer observation. (a) and (b) show respectively, exact match results for patterns `EE` and `FF`, where `E` (`F`) is a spiral pattern with large (small) torsion (Figure 5.2 (a)). (c) and (d) show respectively, exact and approximate ($k = 15$) match results for pattern `(E|F)(E|F)`. (© 2014 IEEE. Reprinted by permission.)

need to traverse to a certain depth whose corresponding label string is shorter than $n_p + k/\text{cost}_i$, where $\text{cost}_i$ is a constant for insertion cost, since otherwise we would have an edit distance larger than $k$. The benefit of implementing approximate search on the suffix tree is that we only need to compute the edit distance once between $P$ and all the appearances of the same substring. Furthermore, if the traversal finishes exploring a branch under a node $u$ and starts to traverse another branch, the columns in the table DP representing the edit distance between $P$ and the label string on the path from the root to $u$ could also be reused.

The single character repetition symbol "+" and the multiple characters with common features symbol "|" are implemented by extending the traditional edit distance. For the single character repetition, a minimum number of repetition $q$ is used to guarantee that the pattern is significant enough for human perception. For example, if $q = 3$, `aaa` is considered to be the same as `aaaaa` but not `aa`. For the multiple characters



(a) (b) (c)

**Figure 5.5:** Using smoothed streamlines to capture high-level features. (a) illustration of resampling on original streamlines. (b) illustration of resampling on smoothed streamlines. (c) resampling on a streamline before and after smoothing. The original streamline is shown on the left with a segment highlighted in a red rectangle. The smoothed streamline is shown on the right with the corresponding segment highlighted in a green rectangle. (© 2016 IEEE. Reprinted by permission.)

with common features, Figure 5.4 shows some search results using the crayfish data set, where `E` is a character representing a spiral pattern with large torsion and `F` represents a spiral pattern with small torsion. `E` and `f` correspond to `e` and `f` as shown in Figure 5.2 (a) respectively. We observe from Figure 5.4 (a) that streamline segments matched with `EE` are mostly spirals with large torsion, and those matched with `FF` in (b) are mostly spirals with small torsion. In (c), streamline segments include the results from both (a) and (b). If we enable approximate search, more swirling streamline segments are detected, as shown in (d).

## 5.5 Further Consideration

In this section, we discuss two points to further improve our FlowString approach: first, we smooth streamlines to remove small scale features and capture high-level ones; second, we introduce the universal alphabet to support queries and comparisons across multiple data sets.

### 5.5.1 High-Level Features

Large-scale shapes or features at higher levels that contain small-scale features could be challenging to detect, due to the extra characters created for small-scale features. For example, in Figure 5.5 (a), the streamline segment forms a circle, but with a small turbulent portion. Our resampling strategy will densely sample this portion to capture the turbulent feature. This hinders the overall circular shape to be captured. As shown in the Figure 5.5 (a), neither the neighborhood of the green sample point nor the blue sample point can cover the entire circle. The corresponding shapes to these two sample points are most likely to be identified as a turbulent segment and a hook shape, respectively.

To allow the overall shape of a streamline segment to be correctly understood, we first

smooth the streamlines, which removes small-scale features. As shown in Figure 5.5 (b), the turbulent portion of streamline is smoothed out, so that the circular pattern can be captured at the red sample point. In Figure 5.5 (c), we demonstrate this with a streamline traced in the crayfish data set. On the left, we observe that the small features create denser sample points on the original streamline. The right one has the streamline smoothed, and the sample points distribute more evenly along the streamline.

In our implementation, we apply a simple Laplacian smoothing for several iterations. In each iteration, we move a point on a streamline towards the center of its two neighbors. More precisely, in each iteration, we update the position of a point $p_i$ with

$$\lambda p'_i + (1 - \lambda) \left( \frac{p'_{i-1} + p'_{i+1}}{2} \right), \tag{5.2}$$

where $p'_i$, $p'_{i-1}$ and $p'_{i+1}$ are the positions of points $p_i$, $p_{i-1}$ and $p_{i+1}$, respectively, and $\lambda$ is a factor that controls the smoothing speed. The smoothing speed is maximized when $\lambda = 0$, which means that we update the position of $p_i$ with the center of its two neighbors. In this paper, we use a moderate smoothing speed with $\lambda = 0.5$. Once the smoothed streamlines are available, users can choose to include the smoothed streamlines in a query. The query will then be performed on both the original and the smoothed streamlines. Streamline segments matched on the smoothed streamlines

will be mapped back to the original streamlines. The matched segments will always be shown in the form of the original streamlines.

### 5.5.2 Universal Alphabet

The approach described above can be naturally extended to multiple data sets by applying the alphabet generation procedure on multiple streamline sets produced from different data sets. As a result, a universal alphabet will be generated for all the data sets. This is possible because for different data sets or streamline sets, most characters will still be similar, although certain characters might be absent in some data sets due to the lack of corresponding features. In practice, we find that most flow patterns can be captured by a limited set of characters, and the universal alphabet can be generated using a moderate number of data sets that contain various flow features. The universal alphabet is beneficial in two aspects. First, it will eliminate the need to generate an alphabet for each data set, if the basic shape primitives presented in one data set are already captured by the universal alphabet. Second, it will provide a more natural way to compare flow patterns across multiple data sets.

Our universal alphabet is also generated using affinity propagation. Similar to the alphabet generation procedure for a single data set, the universal alphabet is generated in two steps. The first step still computes the first-level cluster centers for each

data set independently. Then, we simultaneously consider all the first-level cluster centers as the candidates for the universal alphabet, by computing the dissimilarity values among them and applying affinity propagation for the second-level clustering. Note that we can also generate the universal alphabet in an incremental way using *leveraged affinity propagation*. Assume the data point set is $P$ and the range of possible dissimilarity values is $S$, the entire dissimilarity matrix can be considered as a mapping $M : P \times P \to S$. Affinity propagation considers all data points at the same time and computes the best exemplars (i.e., clustering centers) from $M$. Unlike affinity propagation, leveraged affinity propagation samples a subset of data points $P' \in P$ and computes the best exemplars from the mapping $M' : P \times P' \to S$. In each iteration, leveraged affinity propagation keeps the best exemplars from the previous sample points and replaces the other data points with new sample points. This iterative scheme could be applied to extend our alphabet to include extra features from a new data set by a simple modification: in each iteration, we consider the previous universal alphabet and a subset of data points from the new data set as the candidates for the second-level cluster centers, i.e., characters.

The benefit of incremental clustering is mostly on the performance side. However, a data set normally has hundreds of first-level cluster centers. This implies that affinity propagation should be able to handle tens of data sets, which is enough to generate the universal alphabet. Therefore, we prefer affinity propagation that considers all first-level cluster centers at the same time, since it usually yields better clustering

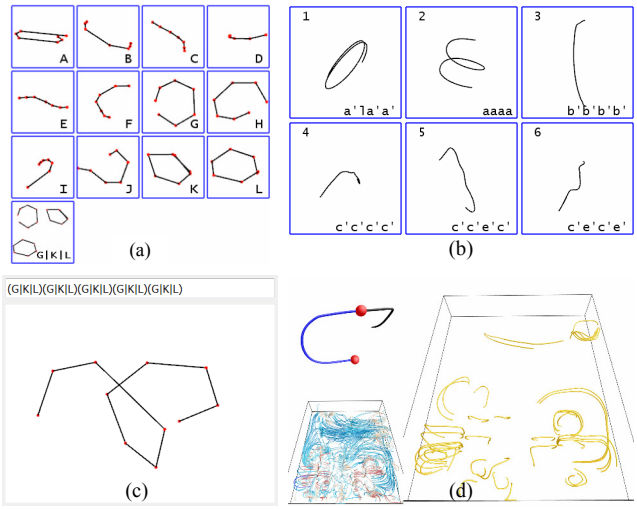



**Figure 5.6:** Alphabet widget (a), vocabulary widget (b), and query string widget (c) with the solar plume data set. Streamline widget (d) with the computer room data set. (a) shows the alphabet visualization where the last character is created by the user to match either `G`, `K` or `L`. (b) shows the first page of the vocabulary widget. (c) shows a query string in the forms of text and polyline. (d) shows the user-selected query segment on the upper-left subwindow (where two red spheres are used to delimitate the blue segment as the query pattern), all streamlines on the lower-left subwindow, and the query result on the right subwindow. (© 2014 IEEE. Reprinted by permission.)

results.

## 5.6 User Interface and Interactions

To make our FlowString a useful tool to support exploratory flow field analysis and visualization, we design a user interface for intuitive and convenient streamline feature querying and matching. Our interface includes four major components: the *alphabet*, *vocabulary*, *query string*, and *streamline* widgets, as shown in Figure 5.6. These

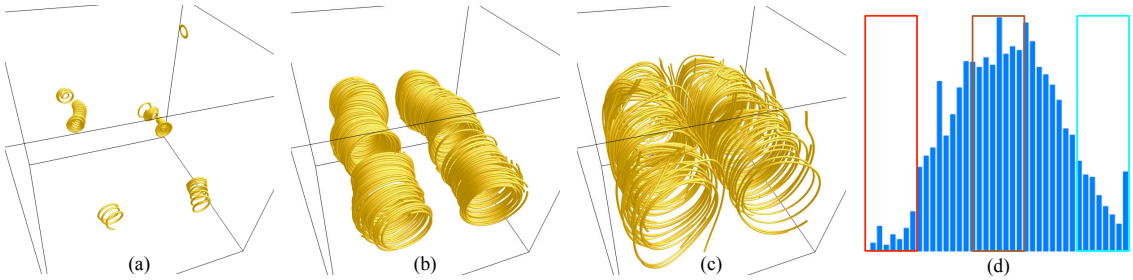



**Figure 5.7:** Circular patterns queried by `MMM` in "universal 1" alphabet (Figure 5.11 (a)) at the user-specified scale using the two swirls data set. (a) small-scale features. (b) medium-scale features. (c) large-scale features. (d) histogram of feature scales. The red, brown and cyan rectangles represent the selected scale ranges corresponding to (a), (b) and (c), respectively. (© 2016 IEEE. Reprinted by permission.)

components support visual query and result retrieval.

The alphabet widget visually displays all the characters, as shown in Figure 5.6 (a). Users can construct a query string from this widget by clicking on the displayed characters or typing in an input box. After clicking on each character, the query string and the query result will be updated on the fly. Users can also select multiple existing characters to create a new character, which can match with either of the selected characters. For example, they can select `G`, `K` and `L` to create the character `G|K|L`, as shown in the bottom of (a). Clicking on this new character, the query string widget will append `(G|K|L)` to the current query string. The query string generated from clicking on `(G|K|L)` five times is shown in (c). The vocabulary widget visualizes all the words automatically detected from the streamline suffix tree, as shown in Figure 5.6 (b). Users can click on a word to retrieve the corresponding pattern in the flow field. They can also select multiple words in sequence to search streamline

segments matching with the concatenation of those words. In the first row of Figure 5.8 on page 135, we show the selected words in the vocabulary widget and their corresponding query result in the streamline widget. The query string widget displays the query in both textual and visual forms, as shown in Figure 5.6 (c). Users can freely change the textual query string and its visual form will be updated accordingly. Several sliders are provided to adjust the parameter for $k$-approximate search, and the thresholds of frequency and length for word generation. In Figure 5.6 (d), the streamline widget shows the input streamlines at the bottom left, from which users select a streamline. Users can then specify a segment of the streamline to query by moving the two end points, which are shown as the red balls. In this example, a "U"-shape segment is selected, and the query result is shown on the right of this widget.

In addition, we provide a bar chart histogram for users to see the scales of matched segments and specify a desired range of scales to further refine the query result. In Figure 5.7, the circular pattern is queried by `MMM`, and the histogram of scales is plotted in the bar chart as shown in (d). Then, users can brush the histogram to select a range of scales for query. In (a), (b) and (c), we show the matched segments at small, medium and large scales, respectively. The brushed ranges are indicated in (d) by the red, brown and cyan rectangles, respectively. To compute the scale of a matched segment, we follow the optimal scale computation in the registration of two point sets [41]. Formally, if $P = \{p_1, p_2, \ldots, p_n\}$ is the set of sample points on the

**Table 5.1**
The ten flow data sets and their parameter values. The timing for matrix is the running time for computing pairwise distance among the neighborhoods of sample points. The timing for affinity propagation clustering includes both the first- and second-level clustering. The column "max dist." shows the maximum dissimilarity between any two sample point in that data set. The timing results are measured in seconds.

| data set | # lines | # points | # sample points | # cluster 1st level | # char. | max dist. | timing | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | matrix | cluster | setup |
| vessel | 100 | 25606 | 1338 | 56 | 5 | 31.11 | 1.45 | 2.0 | 0.06 |
| five c. p. | 150 | 18618 | 1720 | 75 | 6 | 31.06 | 2.39 | 3.6 | 0.07 |
| electron | 200 | 24191 | 1415 | 38 | 4 | 13.98 | 1.62 | 3.3 | 0.05 |
| tornado | 200 | 200735 | 12363 | 141 | 6 | 29.80 | 126.06 | 115.0 | 1.77 |
| two swirls | 200 | 209289 | 13508 | 156 | 6 | 36.05 | 150.39 | 247.1 | 2.07 |
| supernova | 200 | 56210 | 8542 | 150 | 7 | 35.28 | 35.28 | 139.0 | 1.08 |
| crayfish | 150 | 164605 | 7590 | 178 | 11 | 33.77 | 33.77 | 89.6 | 0.97 |
| solar plume | 200 | 257087 | 12484 | 247 | 12 | 36.63 | 128.47 | 221.6 | 2.09 |
| comp. room | 400 | 361258 | 9772 | 262 | 11 | 35.91 | 78.94 | 75.2 | 1.43 |
| hurricane | 200 | 293572 | 4766 | 98 | 7 | 35.13 | 18.42 | 16.5 | 0.28 |

matched segment, the scale of this segment is given by

$$s = \sqrt{\sum_{i=1}^{n} (p_i - c)^2}, \tag{5.3}$$

where $s$ is the computed scale and $c = (p_1 + p_2 + \cdots + p_n)/n$ is the center of points in $P$.

## 5.7 Results and Discussion

We experiment FlowString with multiple data sets. The performance and image results are presented in this section. We also discuss the impact of using universal

alphabets and smoothed streamlines.

### 5.7.1 Performance and Parameters

Table 5.1 shows the configurations of ten data sets, the timing for the first- and second-level affinity propagation clustering, and launching the program. For each of the data sets, we randomly placed seeds to trace the pool of streamlines. All the timing results were collected on a PC with an Intel Core i7-960 CPU running at 3.2GHz, 24GB main memory, and an nVidia Geforce 670 graphics card with 2GB graphics memory.

Affinity propagation clustering can be time-consuming when performed on GPU. We leveraged GPU CUDA to speed up this procedure. For most of the data sets, we performed the clustering by affinity propagation using the GPU. For the solar plume and two swirls data sets, a GPU implementation of *leveraged affinity propagation* was used, since the memory needed to perform affinity propagation exceeds the limit of graphics memory. Unlike affinity propagation which considers all the data points (and the similarities among them) at the same time, leveraged affinity propagation samples from the full set of potential similarities and performs several rounds of sparse affinity propagation, iteratively refining the samples. Thus, the required memory space is reduced with leveraged affinity propagation. For most of the data sets, the clustering

step only took around one minute. For the two swirls data set which contains the most number of sample points, it still could be completed in five minutes. We believe that the timing for clustering is acceptable, since it only needs to run once for a pool of streamlines.

The dissimilarity matrix computation can be performed in reasonable time using the GPU. For the two swirls data set, it took 150 seconds to complete, and the costs for other data sets were even less. Other than these preprocessing steps, the other steps could be finished on the fly. It only took seconds to setup the program for a new run, which includes the time for resampling, computing the dissimilarities between each sample point and each character, and constructing the suffix tree.

Parameter setting is straightforward. The approximation threshold $k$, minimum number of repetition $q$, and minimum length and frequency for generating the vocabulary are four parameters that users can configure. They could be adjusted to update the query result in real time. The insertion and deletion costs are automatically decided for each data set. To avoid frequent insertion and deletion, they are both assigned twice the value of maximum dissimilarity between any two sample points in that data set. This rule applies to all the following case studies.

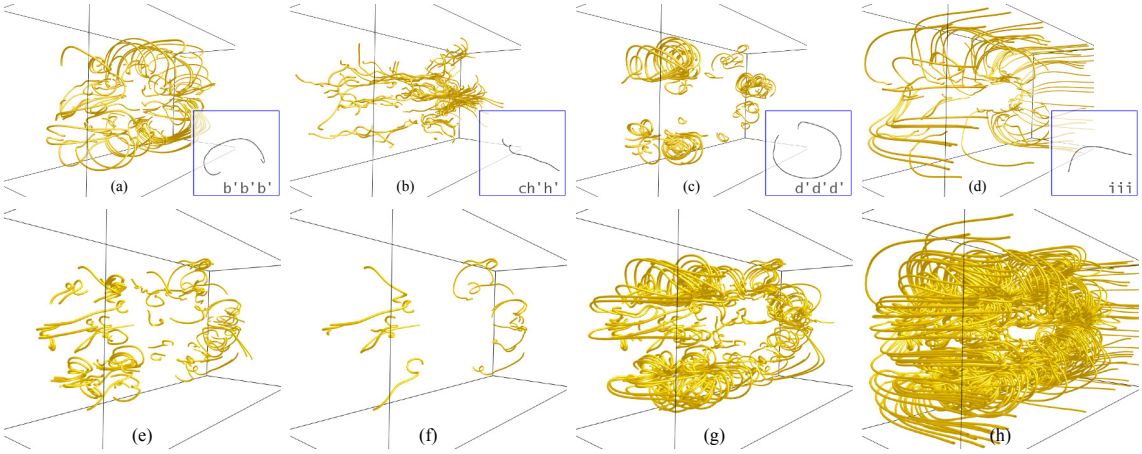



**Figure 5.8:** Case study for the crayfish data set. (a) to (d) show streamline segments matched by four automatically generated words. (e) to (h) show query results of `(A|I)`$^+$`(D|E|F|K)(D|E|F|K)`, `(A|I)(A|I)`$^+$`(D|E|F|K)(D|E|F|K)`, `(A|I)???(D|E|F|K)(D|E|F|K)`, and `(A|I)*(D|E|F|K)(D|E|F|K)`, respectively. (© 2014 IEEE. Reprinted by permission.)

## 5.7.2 Case Studies

### 5.7.2.1 Crayfish

Figure 5.8 demonstrates query results of both automatically generated words and user inputs using the crayfish data set. In the first row of Figure 5.8, the four words are selected from a vocabulary of seven words, which are generated with the minimum number of appearance and the minimum length set to 100 and 3, respectively. We see that the word `b'b'b'` mostly corresponds to streamline segments of "C"-shape. The word `ch'h'` finds those turbulent segments inside. The word `d'd'd'` matches segments with swirling patterns. Unlike those in Figure 5.4 (b), `d'd'd'` is usually

an elliptical spiral instead of a circular one. Finally, the word `iii` corresponds to streamline segments of "L"-shape on the outer layer along the boundary. We find that most of words with clear patterns are repetitions of a single character. A word with multiple characters often indicates a streamline segment that connects multiple patterns, which is less distinguishable by human observers.

In the second row of Figure 5.8, we demonstrate an example of using user input to search for a combined pattern that contains a straight segment followed by a spiral pattern. As shown in Figure 5.2, characters `A` and `I` represent shapes that start with straight segments and `D`, `E`, `F` and `K` are mostly swirling patterns. (e) shows the query result for user input `(A|I)`$^+$`(D|E|F|K)(D|E|F|K)`, where $^+$ indicates that character `A|I` could repeat multiple times. We then further refine the query result by repeating character `A|I`, which ensures that the straight segment is obvious enough for human perception. As shown in (f), the refined query matches less streamline segments, but the straight segment can be better observed in most of the matched segments. The query `(A|I)???(D|E|F|K)(D|E|F|K)` allows any pattern represented by less than three characters to be inserted between the straight pattern and the swirling pattern, which makes the resulting segments in (g) contain more complex patterns. Finally, if we allow any pattern with arbitrary length to be inserted by querying `(A|I)*(D|E|F|K)(D|E|F|K)`, almost all the input streamlines could be matched, since most of the streamlines contain a straight portion on the outer layer and spirals inside the volume.

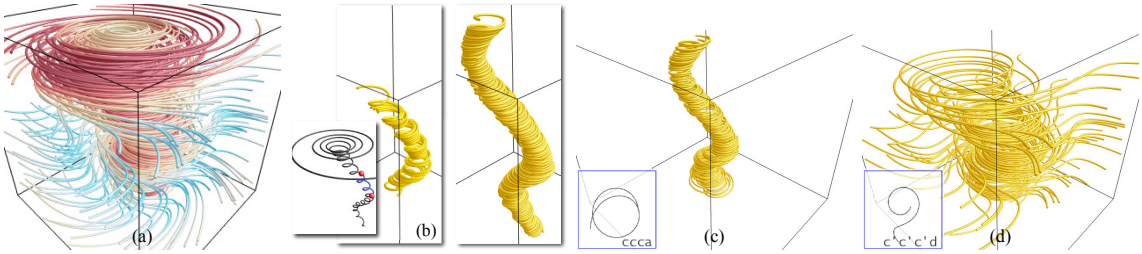



**Figure 5.9:** Case study for the tornado data set. (a) shows all streamlines. (b) shows query results for a user-selected streamline segment with different settings. (c) and (d) show streamline segments matched by two automatically generated words. (© 2014 IEEE. Reprinted by permission.)

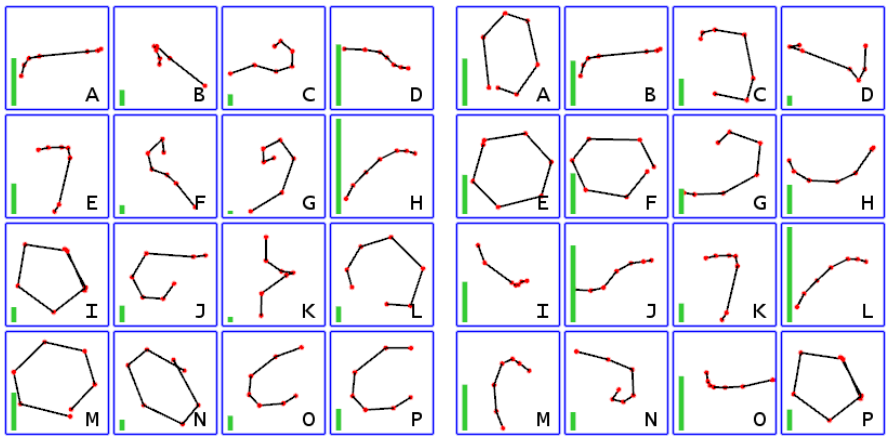
#### 5.7.2.2 Tornado

In Figure 5.9 (b), the query result on the left is matched by using the exact string `a'bbba'a'bbba'a'a'b`, which corresponds to the user-selected segment. The query result on the right is found by replacing each of the characters with a user-defined character `(A|B|E)`, since these three characters are similar. The exact string matches only the segments that are almost the same as the query segment, while the modified query matches more segments in the core of the tornado. Figure 5.9 (c) and (d) show the segments corresponding to the words `ccca` and `c'c'c'd`, respectively. Characters `a` and `c` are mostly circles, and character `d` matches the segments with "S"-shape on the outer layer of the tornado. We observe that when `c` concatenates with `a`, it corresponds to the small-scale circles. When `c` connects with `d`, it matches the large-scale circles. This demonstrates that the scale of a character in a streamline depends on its context, which ensures that the shape for a string is mostly determined.


**Figure 5.10:** Case study for the two swirls data set. (a) shows all streamlines. (b) shows the query result for a user-selected streamline segment with the minimum number of repetition $q = 1$. (c) and (d) show query results for a user-selected streamline segment with $q = 0$ and $q = 1$, respectively. (© 2014 IEEE. Reprinted by permission.)

#### 5.7.2.3 Two Swirls

Figure 5.10 demonstrates query results of two user-selected streamline segments. In (b), the query segment is one that connects a small spiral pattern and a large swirling pattern. The corresponding query string is `d'd'c'c'c'e'e'a'a'c'd'd'd'`, which matches only the query string itself. The reason is that the query string is somewhat complicated, and even the very similar segments might vary for one or two characters, especially in terms of the number of repetition. We then change the minimum number of repetition $q$ to one, and the query string is modified to $\texttt{D}^+\texttt{C}^+\texttt{E}^+\texttt{A}^+\texttt{C}^+\texttt{D}^+$. Note that $\texttt{D}^+$ at the beginning and the end allows the spirals to be displayed in the query result. This query finds two more similar patterns, as shown in (b). In (c) and (d), the query segment is one that connects two large swirling patterns. The query using the exact string `d'c'c'a'aba'a'c'd'd'd'd'd` on that segment finds itself and another very similar one. For the same reason as the previous example, we set $q = 1$. The query




(a) (b)

**Figure 5.11:** Universal alphabets when matching with the solar plume data set. The green bar on the left side of each character indicates its number of appearance in the data set. The alphabets are generated using (a) all the ten data sets and (b) five of the ten data sets, respectively. (© 2016 IEEE. Reprinted by permission.)

string is changed to $\mathtt{D^+C^+A^+B^+A^+C^+D^+}$ accordingly. It matches more segments with the same pattern. In (d), we manually change the query string to $\mathtt{DC^+A^+B^+A^+C^+D}$, which ignores the swirling pattern at the two ends for a clearer observation.

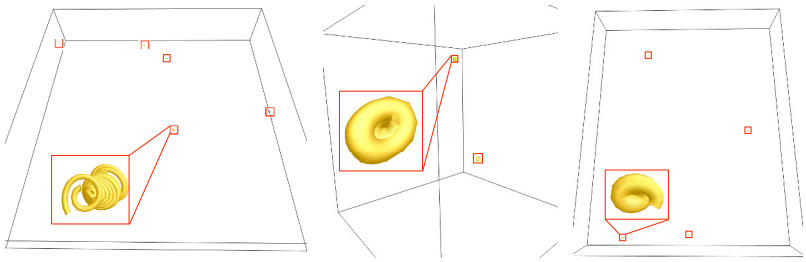
### 5.7.3 Universal Alphabet

In this section, we compare the results of using alphabets generated from single data set and universal alphabet. The comparison is performed both qualitatively and quantitatively, followed by a detailed discussion on the discriminative power of using universal alphabet.

#### 5.7.3.1 Qualitative Comparison

Figure 5.11 demonstrates two examples of universal alphabet when matching with the solar plume data set. Figure 5.11 (a) shows the universal alphabet generated with all the ten data sets, and Figure 5.11 (b) shows the universal alphabet generated from five data sets (namely, crayfish, computer room, solar plume, supernova, and two swirl data sets). Data sets used to generate the second universal alphabet are selected according to their coverage of flow patterns. The five selected data sets are more complicated and contain various kinds of flow pattern, so that they are more likely to generate a meaningful universal alphabet. By comparing the shape and frequency of appearance in the solar plume data set for each character, we observe that the two universal alphabets are actually quite similar. The most common character is `H` in the first alphabet and `L` in the second one. The shapes of these two characters are almost the same. This relationship can be found between `I` and `P`, `M` and `E`, `A` and `B`, `E` and `K`, and `D` and `J`, where the first characters in these pairs are from the first universal alphabet and the second characters are from the second universal alphabet.

In Figure 5.12, we demonstrate the matching result for `III` using the first universal alphabet with different data sets. The character `I` mostly corresponds to the small-scale spirals. Figure 5.12 shows that in the hurricane, supernova, and computer room data sets, string `III` finds us the small-scale spirals, which are difficult to notice and



(a) (b) (c)

**Figure 5.12:** The small-scale spirals matched by the character `I` in the universal alphabet. The data sets used are (a) hurricane, (b) supernova, and (c) computer room. (© 2016 IEEE. Reprinted by permission.)

**Table 5.2**
The ten flow data sets and their average and standard deviation of errors. "uni 1" and "uni 2" correspond to the universal alphabets generated using all the ten data sets and five of the ten data sets, respectively. "single" indicates the alphabet for each data set generated using only that data set. "uni 1 vs. single" and "uni 2 vs. single" show the differences of average errors between the corresponding alphabets.

| | | vessel | 5cp | elec. | tor. | two swirls | super. | cray. | solar plume | comp. room | hurr. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| uni 1 | average | 7.61 | 6.90 | 4.52 | 5.35 | 4.48 | 6.31 | 7.49 | 6.35 | 7.19 | 5.61 |
| | std. dev. | 1.70 | 2.05 | 1.62 | 2.01 | 2.32 | 2.36 | 1.94 | 1.98 | 2.57 | 2.19 |
| uni 2 | average | 7.59 | 8.14 | 5.32 | 6.65 | 3.60 | 6.04 | 7.50 | 5.98 | 7.18 | 5.35 |
| | std. dev. | 1.97 | 3.28 | 2.28 | 2.64 | 2.53 | 2.31 | 2.56 | 1.98 | 2.71 | 2.45 |
| single | average | 6.63 | 6.15 | 2.56 | 3.93 | 3.84 | 5.1 | 6.89 | 5.56 | 7.61 | 5.01 |
| | std. dev. | 2.54 | 2.86 | 1.84 | 1.94 | 2.77 | 2.43 | 1.92 | 1.98 | 2.49 | 2.54 |
| uni 1 vs. single | | 0.97 | 0.75 | 1.96 | 1.42 | 0.64 | 1.25 | 0.60 | 0.79 | -0.42 | 0.60 |
| uni 2 vs. single | | 0.96 | 1.99 | 2.76 | 2.71 | -0.25 | 0.99 | 0.61 | 0.42 | -0.43 | 0.34 |

locate. However, if the alphabet from a single data set is used, the comparison of the same flow feature across multiple data sets needs to start from the very first step for every data set. Moreover, users will have to make the connection of strings or words across data sets by themselves. Using the universal alphabet, users can simply apply the previous query on the later data sets. Thus, the universal alphabet makes it convenient to compare and explore flow patterns for multiple data sets.

#### 5.7.3.2 Quantitative Comparison

In addition to qualitative comparison, we also evaluate the effectiveness of different alphabets through quantitative comparison. Table 5.2 shows the average and standard deviation of errors for each data set using different alphabets. The error is given by the Procrustes distance between the neighborhood of a sample point and the corresponding character. We observe that the errors using the universal alphabet are usually slightly larger, which is expected. Note that the errors using the alphabet from a single data set is not always smaller due to the fact that affinity propagation is not specifically designed to reduce the within-group variance.

We apply $t$-tests to evaluate differences between errors using the alphabet "universal 1" and "single", and using the alphabet "universal 2" and "single" for each data set. All the $p$-values are smaller than $1 \times 10^{-26}$, indicating that for each data set significant difference is found between the universal alphabets and the alphabet from a single data set. This is not surprising with the large sample size and relatively small standard deviation. According to the central limit theorem, the estimated variance of sample averages is very small in this case. Thus, even small difference between the sample averages could be significant. However, even if the errors are unlikely to come from the same distribution, the differences between the averages are not large. We observe in Table 5.2 that most of the differences are smaller than one, and the

largest difference is 2.76 (between "universal 2" and "single" for the electron data set). These value are relatively small compared to the maximum distance between the neighborhoods of any two sample points, which is usually larger than 30 (Table 5.1). In addition, for the more complicated data sets, e.g., the crayfish, solar plume, computer room, and two swirls data sets, we find that although the errors are usually larger, the differences between errors from universal and single alphabets are actually smaller. For example, the computer room data set has the largest average error with the alphabet from a single data set, but the average errors even decrease with the universal alphabets. In contrast, the data sets that have simple flow patterns, e.g., the electron and tornado data sets, suffer from larger error differences. This is probably due to the fact that the streamlines in these data sets are similar and better captured by a small alphabet generated from a single data set. Thus, they are more likely to be affected when we consider other data sets with different features.

#### 5.7.3.3 Discriminative Power

Our results show that although the universal alphabet usually produces comparable results for those complicated data sets, the discriminative power of the universal alphabet to simple data sets often reduces. The first row of Figure 5.13 shows that the alphabet generated from only the electron data set produces a high-quality vocabulary. The word `aaa` finds the mostly straight streamlines; `bbb` matches the

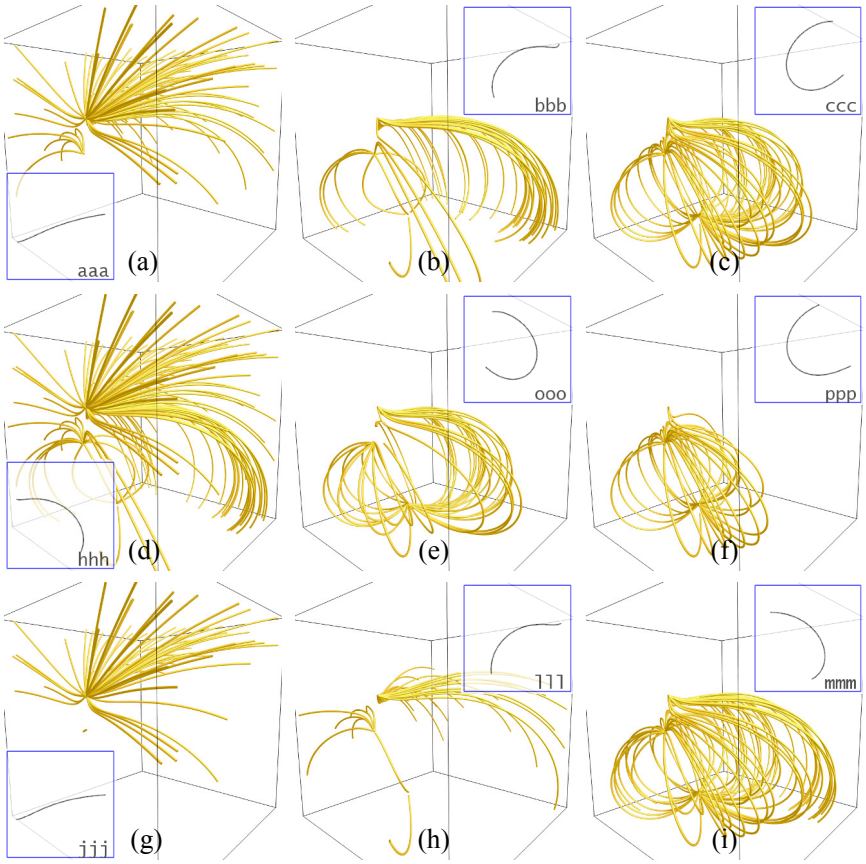
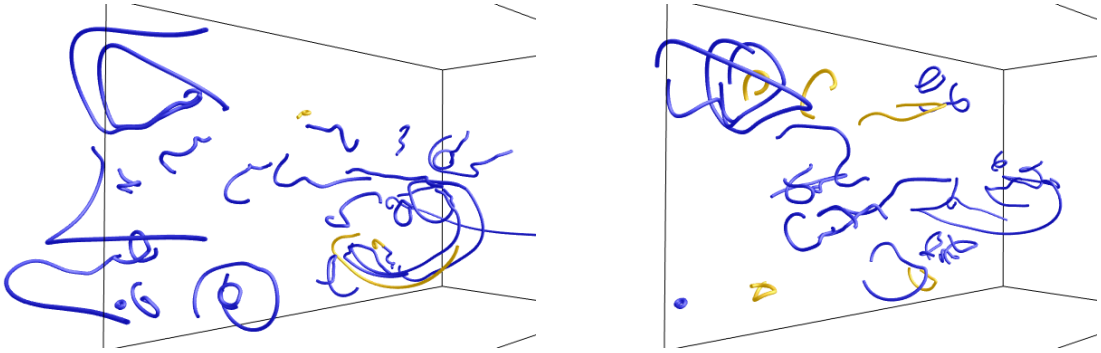



**Figure 5.13:** The words of electron data set using the alphabets generated from itself (first row), from all the ten data sets (second row), and from five of the ten data sets (third row), respectively. (© 2016 IEEE. Reprinted by permission.)

dissymmetric curvy streamlines; `ccc` corresponds to the symmetric curvy streamlines with different winding angles. The second and third rows of Figure 5.13 shows six words based on the universal alphabet generated from all the ten data sets and five of the ten data sets, respectively. Although the words still distinguish streamlines with different winding angles to some degree, the discriminative power apparently decreases compared to the alphabet generated from a single data set. This is due to the fact that the streamlines in the electron data set contain only simple patterns. If



(a) (b)

**Figure 5.14:** Patterns matched by `GGG` (a) and `OOO` (b) in "universal 1" alphabet using both the original and smoothed streamlines of the crayfish data set. The yellow segments are matched on the original streamlines. The blue segments are only matched on the smoothed streamlines. (© 2016 IEEE. Reprinted by permission.)

the electron data set is the only one to generate the alphabet, these patterns could be well captured. However, when other data sets are considered, these clusters have to compromise with other data sets. As a result, the discriminative power for this data set is traded to enhance the overall effectiveness. As we observe in Figure 5.13, the features shown in (a) and (b) are merged in (d), when all the data sets are considered. On the other hand, since more complicated data sets already contain various kinds of flow pattern, including other data sets might not introduce new patterns or increase the in-group variance. Therefore, applying our approach on these kinds of data set seems to produce more stable results.

### 5.7.4 Smoothed Streamlines

In Figure 5.14, we query the crayfish data set using "universal 1" alphabet with both the original and smoothed streamlines. The segments that are only matched on the smoothed streamlines are highlighted in blue. In (a), the queried pattern corresponds to `GGG`, which is a hook shape with a small circle-like pattern at one end. We observe that the yellow segments matched on the original streamlines have exactly the queried shape, but the blue segments are more likely to contain some turbulent portion of the hook. In (b), a "U"-shape pattern is queried. The segments matched on the smoothed streamlines seem to contain even more diversified shapes. But overall, they are either in the "U"-shape or elongated ellipse shape, which can be considered as the concatenation of two "U"-shape patterns. Actually, this query result depends on the degree that we smooth the streamlines, because this degree determines which features will be smoothed out. If the degree is large, more features will be removed, and the query result will be more diversified when mapped back to the original streamlines.

## 5.8 Empirical Expert Evaluation

To evaluate the effectiveness and learning difficulty of FlowString, we collaborate with a domain expert in turbulent flow (Dr. Raymond Shaw), whose research focuses on

understanding the influence of turbulence on cloud particle growth through condensation and collision. Although three tasks were designed, this study aimed at providing comprehensive reasoning on the effectiveness instead of quantitative results. Dr. Shaw was informed that the comments on reasons behind his rating and selection were more important than the accuracy of tasks. The three tasks are:

- Task 1: In the solar plume data set, find the streamline segments of the small spiral pattern and those of the turbulent flow pattern.
- Task 2: In the crayfish data set, find the streamline segments corresponding to the pattern of a hook connecting with a spiral, and those corresponding to the pattern of small repeated spirals.
- Task 3: In the two swirls data set, find all the common flow patterns.

For the first two tasks, images of the specific flow patterns to find are provided along with text description. For each task, similar questions are asked. These questions can be summarized in three categories:

- Rate the effectiveness in the five-point Likert scale of the vocabulary, approximate search, multiple characters with common features, single character repetition, and wildcard characters.
- Select the most helpful functions to accomplish the tasks.
- Provide detailed comments on the rating and selection.

### 5.8.1 Comments

After learning the features and interface of the program and practicing on various data sets, Dr. Shaw performed the tasks and provided his feedback. We organize and present his feedback as the following. In general, FlowString is novel and effective. It provides multiple searching features to identify and locate flow features. The characters successfully capture the basic pattern of flow features. The overlapping of six sample points for two neighboring characters enforces their unique shapes, which is powerful for identifying specific features of interest. In many cases, the repeated use of a single character is very useful for narrowing down the matched results to a specific pattern. In addition, users can define a feature that combines multiple characters which appear similarly. This greatly enhances the ability to locate specific types of flow features. The ability to work with the alphabet, including wildcard characters, allows for great flexibility. Even early in the evaluation the question mark was found to be especially useful in matching a set of characters in a more flexible or general way. For example, if a set of characters was combined so as to search for a specific flow structure of interest, but had become too narrowly defined, inclusion of one or more question marks efficiently allowed the query to become more general. As experience was gained with the range of alphabet capabilities, aspects such as the "prefer user alphabet" proved to be powerful in identifying specific types of features. In particular, the ability to impose directionality on the ordering of the characters is

very useful in finding specific geometries of interest.

Visual aspects of FlowString were found to be very effective. Specifically, the interface visualizes characters and words effectively. In this regard, the ability to rotate individual characters in 3D is very important, e.g., for observing the torsion of a spiral. Certain characters, when viewed as 2D projections, initially look like minor variations on a theme, but when viewed in 3D their differences become much clearer. The user interface tools for rotating and viewing shapes are easy to learn. The streamline widget was another graphical interaction tool that proved to be very simple to use and efficient in its ability to allow interaction between the streamlines themselves and the alphabet. In fact it was found that this widget was very useful in "teaching" users how to use the alphabet, e.g., the important aspects like repeated characters.

The vocabulary was found to be one of the most powerful tools, especially for a new user. In effect, the vocabulary has already identified dominant flow structures, even when these features were complex, varying widely in shape and across scales. For example, a search was initiated for what physically could be described as an entrainment event in the crayfish pattern, specifically a long, straight steamline near the outside of the flow, that ends in a tight swirl as it enters the more complex central flow region (e.g., see outer flow features in Figure 5.8 (h) on page 135). Such entrainment events would be typical of a flow pattern of interest in exploring a physical system. Initially the pattern was searched for by using the streamline widget to

select a specific example, and then the resulting word was generalized by including wildcard characters, etc. Subsequently, when moving to the vocabulary approach, it was found that a variety of complex but similar streamline patterns were quickly identified, including the same pattern that was originally selected using the streamline widget. Ultimately, the vocabulary proved not only more efficient, but more effective in generalizing the query.

Dr. Shaw further indicated that the scale independence of the method is powerful, once fully appreciated. This would be similar to the concept of a wavelet display, in which correlation is shown as a function of position and scale. In terms of learning difficulty, FlowString has sufficient basic features that a user can achieve an impressive range of tasks even after minimal training. FlowString has a range of powerful but more subtle capabilities and benefiting from the full range of these features requires practice and development of experience. Furthermore, it is important to discuss specific features of this tool with an expert for full understanding. The biggest challenge for a scientific user, in his opinion, is the mental picture originally brought to the problem of scale dependence of the flow features and its relationship to streamline sampling resolution. It is crucial to understand that the character matching involves a resampling of seven points, i.e., that the identification of features through cumulative curvature results in the ability to identify similar shapes or features across a wide range of scales. With around an hour of experimenting with FlowString alphabet and vocabulary options the ability to find specific types of features increases rapidly.

### 5.8.2 Rating and Multiple Selection Questions

Each of the five query features, i.e., the vocabulary, approximate search, multiple characters with common features, single character repetition, and wildcard characters, was rated for each of the tasks. The rating scores of the effectiveness of these features echo these comments. Two features were not rated because they were not used in the tasks. For the thirteen scores in the five-point Likert scale we received, ten of them were rated four points, two were rated five points, and one was rated three points. In the first task, Dr. Shaw felt that every query feature is useful in some aspect, and rated each of them four points. Among these query features, he selected approximate search and single character repetition as the most helpful ones. This might due to the fact that these two features require less experience to use, since the approximate search can reduce the difficulty in composing the exact query string, which is convenient for beginners and the concept of single character repetition is straightforward.

In the second task, he rated the approximate search five points and the single character repetition three points. In addition, he selected the alphabet widget to be the most helpful one to compose the query string, and multiple characters with common features and wildcard characters to be the most effective ones to refine the query results. This selection is consistent with the characteristics of the crayfish data set,

where each streamline might cross multiple flow features, with less single character repetition. In this case, the multiple characters with common features can group similar flow features, and the wildcard characters can deal with the somewhat turbulent segments connecting the query patterns. This indicates that with only tens of minutes of experience with the tool, users will be able to determine the most effective features to use, even if the use of those features is not trivial.

In the third task, Dr. Shaw rated the vocabulary widget to be very useful to find all flow patterns with five points. He did not use the approximate search and wildcard characters in this task, since he was confident to choose which features to use. He selected the vocabulary widget, streamline widget, and multiple characters with common features to be the most helpful features. The streamline widget was used to determine the encoding of a segment when composing the query string. Overall, from these observations, we feel that although users might experience some difficulties in using the tool at the very beginning, they should be able to understand the use of different features and determine the appropriate features according to the given task within one or two hours.

# Chapter 6

# VesselMap: A Web Interface to Explore Multivariate Vascular Data

## 6.1 Overview

Visualization of vascular data sets, especially revealing blood flow patterns in the aneurysm regions and relationship among the blood flow and multiple scalar properties, is critically important to understand the formation and ruptures of aneurysms and develop comprehensive treatment plans.

Toward this end, we develop VesselMap, a novel web-based solution to assist medical experts in exploring vessel data and analyze the relationships among different properties. VesselMap is centered on a scheme that enables 2D illustrative visualization of parameters for true 3D or 4D data. Using this scheme we first flatten the 3D vessel structure and corresponding parameters into a 2D plane. Then, all subsequent interactions can be operated in 2D. Brushing is supported on VesselMap, so that users can simply drag the mouse to select a set of blocks. This provides a much easier way to specify and label regions of interest and eliminates the occlusion of vessel branches. It also serves as a clearer and paper-friendly overview, since rotation and zooming are not available on a printed report. Multiple types of queries are supported by this 2D visualization together with a histogram visualization. Users can query the property distribution over a region, or query regions by specifying a set of value ranges of the scalar properties. Furthermore, a segmentation scheme is provided to investigate local characteristics for any parameter. The groups generated from segmentation can be effectively displayed on the 2D illustration, and their differences are evaluated and displayed in a difference matrix. Users can easily discover the relationships among groups and properties. Finally, the web-based environment places a minimum amount of effort to setup and requires only the displayed information to be transferred.

Our application consists of a web front-end client for visualization and user interface, and a back-end host for computation. The host and client talk to each other with message passing. The host loads the data and performs the histogram computation and

segmentation based on query messages from the web front-end, which is described in Section 6.2. The web front-end helps visualize the computed information in multiple forms and interacts with users, as described in 6.4.

## 6.2 Data Processing

### 6.2.1 Data Reduction

The 3D vessel flow field data set we deal with contains multiple time steps and each time step might correspond to multiple vector/scalar fields. However, due to the inherent structure of vessels, they only occupy a small portion of space in the entire volume. Furthermore, it might not be necessary to load all time steps and properties into memory at the very beginning. Instead, the volumes could be loaded on demand when the program runs. For each volume, to reduce the data to be loaded, we first evenly partition it into data blocks, and then the 3D vessel flow is analyzed. For efficiency, data blocks that do not contain any vessel structure will not be loaded.

Two look-up tables are used to record the data loading status. Figure 6.1 illustrates this data structure. The first look-up table is a mapping from a time and property name pair to an index indicating the starting position of memory which contains the corresponding volume. The second look-up table indicates the memory position of

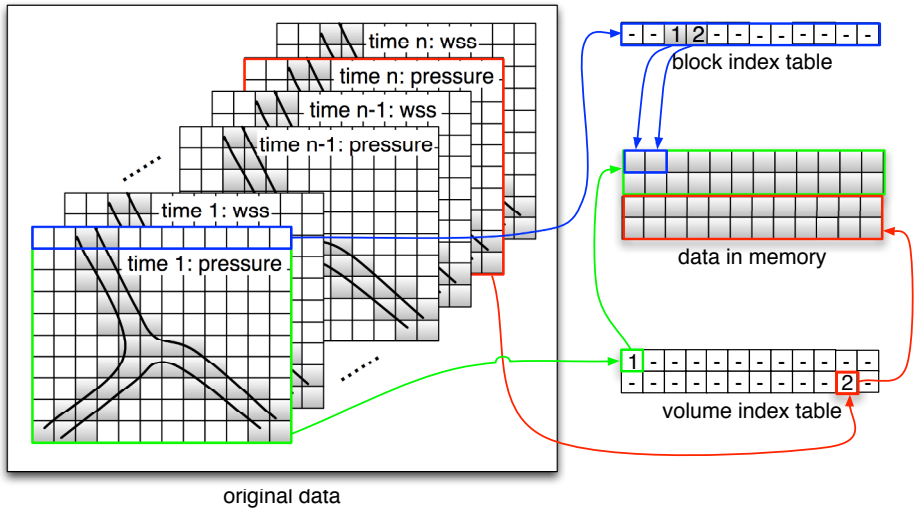



**Figure 6.1:** Two levels of look-up tables for data reduction.

each data block within a volume. Note that we only need one look-up table for the data blocks, since the vessel structure is the same for all the time steps and properties. If the look-up table indicates the volume being visited does not reside in memory, it will be loaded. If the look-up table indicates the block being visited is not loaded, the visit will just be ignored, since this means that the data block does not contain any vessel structure and need not to be studied. The blocks that contain the vessel structure are marked in gray color. Taking the first row of a volume as an example, all blocks are empty except for the third and fourth block. An index in the block index table (first look-up table) contains a single integer indicating the position of this data block in a volume, e.g., the third block is the first non-empty block and marked by "1". If an index corresponds to an empty blocks, it is marked by "–". The volume index table (second look-up table) has a similar structure. Assuming that we only load the "pressure" volume for the first and $n$-th time step, the indices corresponding

to these two volumes indicate the positions of these two volumes in the memory. For the volumes that are not loaded yet, they are marked by "–".

### 6.2.2 Histogram Computation

The computation of statistical information of voxels/particles is implemented using OpenCL in order to utilize the parallelism of GPU/CPU, where each thread deals with one voxel/particle. In this way, the removal of empty data blocks also reduces the number of threads needed, since a less number of voxels are counted. For simplicity, the number of bins for each histogram is defined by a single value. The range of values being studied is first initialized as the entire value range of a property, and can be later modified by users manually. The value of each voxel/particle is quantized uniformly in that range to decide which bin this voxel/particle belongs to. The host will compose the results in JSON format and send it as a text string to the web front.

## 6.3 Algorithm

Our exploration is guided by statistical information, mostly histograms of particles with different properties. The property values of particles are either interpolated in scalar volumes (e.g., pressure and WSS) or derived from particle tracing (e.g., age).

Users can freely interact with the histograms to select a combination of bins in one or more histograms to filter particles and highlight the corresponding region. In addition, we propose an approach to map the 3D vessels to a 2D *VesselMap*, while preserving its perceptual structure. Users can brush the 2D VesselMap or select a group from our segmentation to specify a region of interest, and the histograms computation will be constrained in the user-specified region. They can further interact with histograms or select regions to gradually refine their query.

We provide a segmentation of vessel structure to compare the histograms in different regions, so that local behavior can be observed. The segmentation can be guided by different mapping functions to provide a variety of results. The differences of regions are derived from their histograms. In addition to the traditional measures which usually compute a single value to represent the difference, we visualize the difference using the quantile-quantile plot (Q-Q plot) [8]. An interface to show all differences between every pair of groups is developed for global investigation.

Our system consists of a web front-end client for visualization and user interface, and a back-end server for computation. The server is responsible for histogram computation, VesselMap generation, and flow visualization. While the web front-end displays visualization results, it also interacts with users and requests the server to update visualization results through messages. Figure 6.2 shows the interface of VesselMap. The interface consists of three regions. The 2D VesselMap representation and the 3D

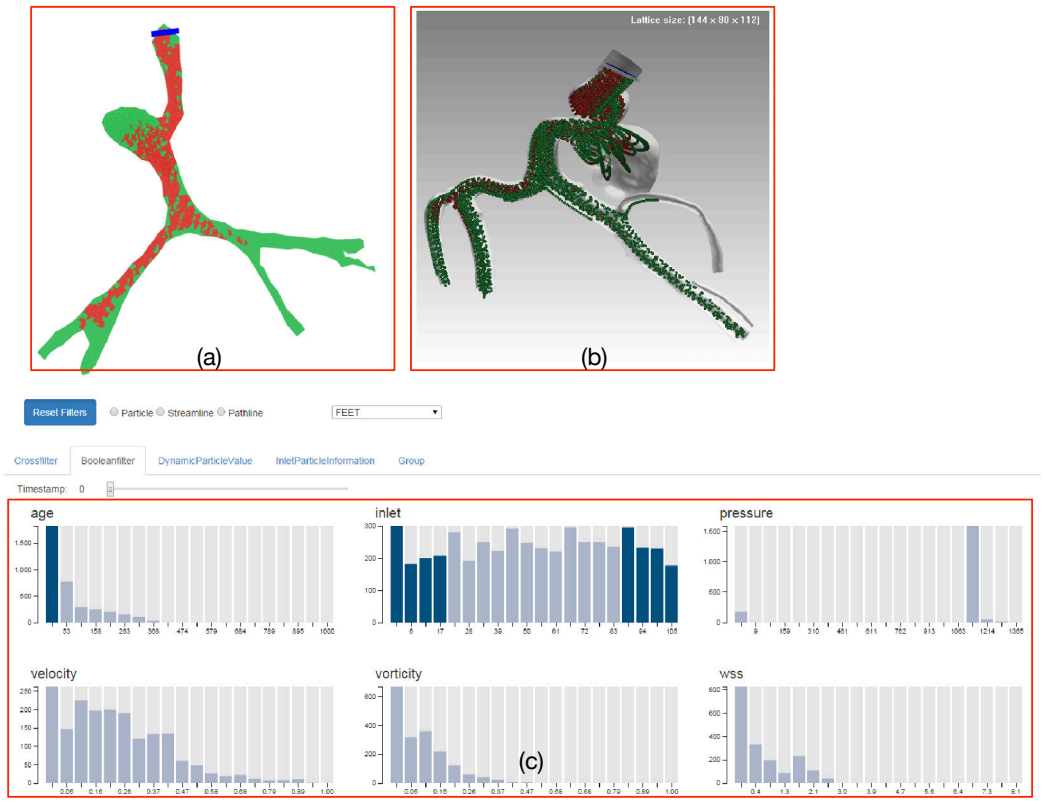



**Figure 6.2:** VesselMap Interface. (a) 2D VesselMap. (b) 3D particle rendering. (c) Histograms.

particle/streamline rendering are shown at the top-left and top-right regions of the web page. The bottom region contains multiple tabs, including histograms, particle inlet information, and group comparison.

### 6.3.1 VesselMap: a 2D Representation

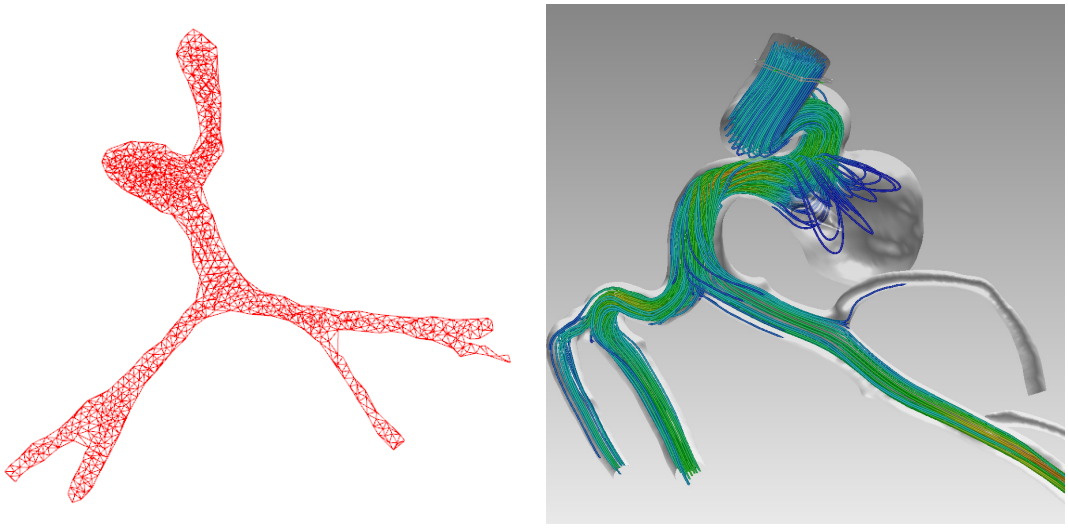
The entire volume is evenly partitioned into small blocks (e.g., $3 \times 3 \times 3$ in this paper), and only blocks containing the vessel structure are loaded. These 3D blocks approximate the vessel structure. We map the blocks to points in 2D, so that they

can be displayed on a webpage with lower overhead and easier interaction. The mapped points are then triangulated to form a mesh structure representing the vessel structure. To build the connection between the original 3D volume and the 2D image, the mapping should preserve the local shape of vessels. We formulate this as a 2D graph layout problem, and achieve the desired layout using a minimization approach. Assuming the blocks that contain some of the vessel structure are $B = \{b_1, b_2, \cdots, b_n\}$, the neighboring blocks of $b_i$ are those whose distances to $b_i$ are smaller than or equal to a given threshold $\delta$, and the non-neighboring blocks are those whose distances to $b_i$ are larger than $\delta$. The energy of this mapping is defined as follows:

$$\sum_{b_i \in B} \sum_{b_j \in B} c_{ij}, \text{ where}$$

$$c_{ij} = \begin{cases} w_{\leq} \left| d_{ij} - e_{ij} \right|, & e_{ij} \leq \delta, \\ w_{>} \left| d_{ij} - g_{ij} \right|, & e_{ij} > \delta. \end{cases} \tag{6.1}$$

where $c_{ij}$ is the cost between $b_i$ and $b_j$, $w_{\leq}$ and $w_{>}$ are weights for neighboring and non-neighboring blocks, respectively, $d_{ij}$ is the distance between $b_i$ and $b_j$ in the 2D image, $e_{ij}$ is their Euclidean distance in the original 3D volume and $g_{ij}$ is their geodesic distance in the 3D vessel structure. Intuitively, for neighboring blocks, we preserve their original distances to maintain the local shape of vessels. For non-neighboring ones, using the geodesic distance will separate two blocks in different vessel branches to avoid occlusion. Figure 6.3 (a) shows the mapped points with triangulation, we observe that the shape is still similar to that in 3D, but occlusion is removed.



(a) (b)

**Figure 6.3:** Mapping from the 3D volume space to a 2D VesselMap representation using the VDS1. (a) the mesh structure of VesselMap. (b) streamline visualization with a semi-transparent vessel structure.

Considering block centers as point clouds, the geodesic distance is approximated by the shortest path between two points. Given a threshold $\epsilon$, we initialize the geodesic distance matrix as follows:

$$g_{ij} = \begin{cases} e_{ij}, \ e_{ij} \leq \epsilon, \\ \infty, \ \text{otherwise.} \end{cases} \tag{6.2}$$

where $e_{ij}$ is the Euclidean distance between the centers of blocks $b_i$ and $b_j$. Running an all-pair shortest distance algorithm (e.g., Floyd-Warshall algorithm) on this distance matrix, we can obtain the approximated geodesic distance in $O(n^3)$ time, where $n$ is the number of blocks. In our implementation, we use $\epsilon = 1.5$, which is slightly larger than the distance between two neighboring blocks. This means that at the initial

stage, only the distances between neighbors are known, and the distances between non-neighboring blocks are then approximated by the shortest paths.

To obtain the positions of blocks $V = \{v_1, v_2, \cdots, v_n\}$, where $v_i$ is the position of block $b_i$, Equation (6.1) is represented as a matrix, so that the minimization is performed by solving a linear system in a least-squares sense. Since in these equations $d_{ij} = \sqrt{(v_i - v_j)^2}$ is not linear, we propose a scheme to solve them in multiple iterations. Let $V = \{v_1, v_2, \cdots, v_n\}$ be the desired positions and $V' = \{v'_1, v'_2, \cdots, v'_n\}$ be the current positions. Then a target distance $t_{ij}$ is achieved by moving the two blocks along the line segment connecting them. Note that the target distance $t_{ij}$ is $e_{ij}$ for neighboring blocks and $g_{ij}$ for non-neighboring blocks. In this way, the equation $|d_{ij} - t_{ij}|$ is rewritten in the form of $|(v_i - v_j) - t_{ij}(v'_i - v'_j)/(|v'_i - v'_j|)|$, where $t_{ij}(v'_i - v'_j)/(|v'_i - v'_j|)$ indicates that current positions $v'_i$ and $v'_j$ are moved along the line passing them to achieve the target distance $t_{ij}$. Applying the least square method will provide us a new set of positions based on the current positions. We will repeat this procedure until the positions do not change or some predefined number of iterations is achieved.

### 6.3.2 VesselMap Segmentation

In the next stage, VesselMap is segmented into regions. The connection among the segmented regions form what is known as the nerve of VesselMap. Inspired by Singh

et al. [95], we develop this segmentation scheme to divide the vessel structure at the block level based on a mapping function that assigns each block a scalar value. The regions generated from segmentation serve as a starting point to investigate their local behaviors. The nerve of VesselMap provides an abstract overview of the vessel data. Depending on the selection of mapping function, the nerve of VesselMap conveys different information. For example, if the geodesic distance from a block to the inlet is used, it represents the shape of the vessel structure; while if a scalar property in a block is used, the nerve of VesselMap may represent the topological structure of that property.

#### 6.3.2.1 Background

The construction described in this section is motivated by the concept of the nerve of a covering in the field of topology. We refer the readers to [37] for background on topological space and simplicial complex, and only provide a brief description here. A finite *covering* $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of a topological space $X$ is a collection of subsets $U_\alpha$ of $X$ whose union is the entire space $X$, where $A$ is a finite index set. The *nerve* $N(\mathcal{U})$ of a finite covering $\mathcal{U}$ is a simplicial complex whose vertex set is the index set $A$, and a $k$-simplex $\{\alpha_0, \alpha_1, ..., \alpha_k\} \in N(\mathcal{U})$, if and only if $U_{\alpha_0} \cap U_{\alpha_1} \cap \cdots \cap U_{\alpha_k} \neq \emptyset$. However, this requires a meaningful covering to construct the nerve, which is usually not immediately available. In practice, we often obtain the covering of the space $X$

through a parameter space $Z$ that is equipped with a covering. Let $X$ and $Z$ be two topological spaces and $f : X \to Z$ be a continuous map. If $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ is a covering of $Z$, then $\{f^{-1}(U_\alpha)\}_{\alpha \in A}$ is a covering of $X$. We denote $f^{-1}(U_\alpha) = \bigcup_{i=1}^{N_\alpha} f^{-1}(U_\alpha)_i$, where each $f^{-1}(U_\alpha)_i$ is a connected component of $f^{-1}(U_\alpha)_i$ and $N_\alpha$ is the number of connected components.

#### 6.3.2.2 Our construction

We consider the space of all blocks $B$ and a map $f : B \to \mathbb{R}$, which assigns each block a real number. Assuming the range of $f$ is $[r_{\min}, r_{\max}]$, the subsets $\{[r_{\min}, r_1], [r_2, r_3], ..., [r_k, r_{\max}]\}$ is a covering of $[r_{\min}, r_{\max}]$, where $r_{i+1} < r_i < r_{i+2}$, and $[r_{i-1}, r_i]$ and $[r_{i+1}, r_{i+2}]$ are two overlapping subsets. Considering the interval $[0, 10]$ as an example, $\{[0, 4], [3, 7], [6, 10]\}$ is such a covering of $[0, 10]$. Let $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ be a covering of $\mathbb{R}$, then the set $\bar{\mathcal{U}} = \{\bigcup_{i=1}^{N_\alpha} f^{-1}(U_\alpha)_i\}_{\alpha \in A}$ forms a covering of $B$. Note that each element $f^{-1}(U_\alpha)_i \in \bar{\mathcal{U}}$ is represented as a vertex in the nerve of VesselMap. Simply, given a mapping function $f : B \to \mathbb{R}$, we first determine the minimum and maximum of mapped values, i.e., $r_{\min}, r_{\max}$. Then, we evenly divide the range into overlapping sub-ranges, and each block corresponds to one or more sub-ranges according to its mapped value. Finally, connecting the neighboring blocks that correspond to the same sub-range forms a set of connected components, which serve as the vertices in the nerve of VesselMap, denoted as $V = \{v_0, v_1, \ldots, v_k\}$, where $k$ is

(a) (b)

**Figure 6.4:** The segmentation and the nerve of VesselMap using the geodesic distance to the inlet as the mapping function. The result is colored by groups of blocks. The rectangle nodes and the edges connecting them show the nerve of VesselMap. (a) Blocks are grouped by their distances to the inlet. (b) Blocks are grouped by the vorticity values of the particles. In this case, the blocks that do not contain any particle are not grouped and are colored in white.

the number of connected components. We name each connected components a group of blocks.

For the nerve of VesselMap, we only consider simplices up to dimension one in the nerve of a covering, i.e., vertices and edges. Therefore, only the connections between two vertices are taken into consideration. Following the previous definition, an edge (1-simplex) $e_{ij} = (v_i, v_j)$ is in the nerve of a covering, if and only if $U_i \cap U_j \neq \emptyset$, where $U_i$ and $U_j$ are subsets in the covering corresponding to $v_i$ and $v_j$, respectively. In our context, if two groups of blocks share some blocks in common, an edge will be added between them in the nerve of VesselMap.

However, this procedure generates segmentation results with overlapping. We further

simplify the procedure and divide the range of $f$ into non-overlapping sub-ranges, and therefore, the groups are non-overlapping as well. In this case, an edge $e_{ij} = (v_i, v_j)$ is in the nerve of VesselMap, if a group represented by $v_i$ contains a block $b_p$ that is a neighbor of some block $b_q \in v_j$. Figure 6.4 shows two examples of the segmentation and the nerve of VesselMap. In Figure 6.4 (a), the neighboring blocks whose geodesic distances to the inlet fall into the same range are grouped together. This segmentation produces similar results as those approaches that consider the shape of structures for segmentation. In addition, the corresponding nerve demonstrates the topological structure of the vessel.

### 6.3.3 Comparing Regions and Properties

One of the major goals of our application is to guide users to explore the relationships among properties. However, we feel that the commonly used approaches (e.g., correlation coefficients) only provide one single value indicating the difference, which fails to answer how they differ from each other. In our approach, in addition to the traditional difference values, we use Q-Q plot to capture the details. Quantiles are points taken at regular intervals from the cumulative density function (CDF) of a random variable. The concept of quantiles differs from percentiles in the sense that quantiles are indexed by sample fractions instead of sample percentages. Generally, the $p$-th quantile $Q(p)$ of a random variable $Z$ is the value $z$ such that

(a) (b) (c)

**Figure 6.5:** Quantile-quantile plot examples. The correlation of quantiles is mapped to the background color of the plot. The ranges of quantiles are mapped by two blue bars. The two distributions in (a) are very similar; those in (b) are less similar; and those in (c) are least similar.

$P(Z \leq z) = p$. The Q-Q plot for two random variables $Z_1$ and $Z_2$ plots the quantile pairs $< Q_{Z_1}(p_1), Q_{Z_2}(p_1) >, < Q_{Z_1}(p_2), Q_{Z_2}(p_2) >, \cdots, < Q_{Z_1}(p_n), Q_{Z_2}(p_n) >$ on a 2D plane, where $p_1 < p_2 < \cdots < p_n$ are equally spaced.

Figure 6.5 shows three examples of the Q-Q plot. For a clearer view, the plotted points always occupy the entire space, i.e., they start from the bottom-left corner and end at the top-right corner. In addition, a blue bar is used to indicate the local value range in a group with respect to the global value range in the volume. In (a), the two distributions are almost identical and the points are mostly aligned along the diagonal. In (b), the two distributions are less similar. The slope of the Q-Q plot is first steep and then becomes flat. This indicates that the distribution corresponding to the $x$-axis has slightly higher probabilities for the smaller values while the distribution corresponding to the $y$-axis has slightly higher probabilities for the larger values. In (c), the two distributions are least similar. The slope of the Q-Q plot is almost flat for a large portion of quantiles. This indicates that the distribution corresponding to the $y$-axis has very high probabilities for the smallest values. Thus,

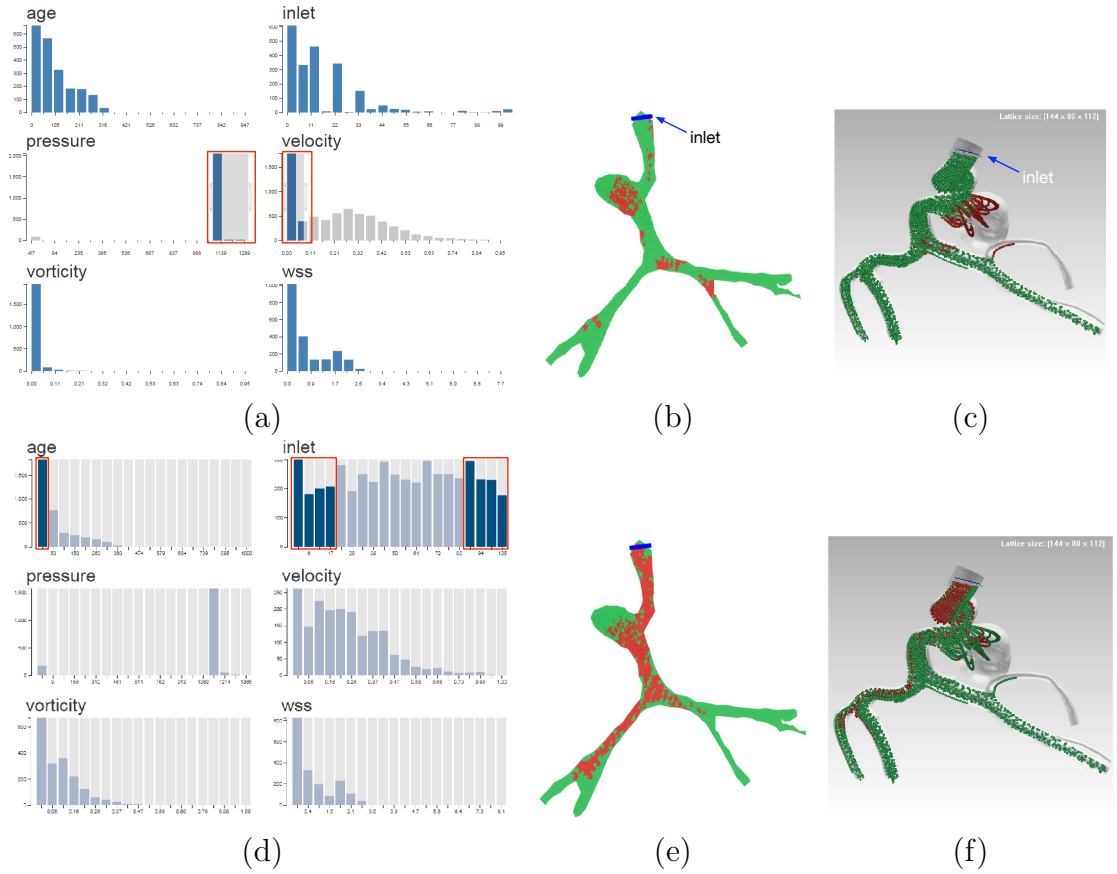



**Figure 6.6:** Cross filter and boolean filter for refining query on histograms. The first row shows the cross filter and the second row shows the boolean filter. The selected ranges in histograms are highlighted in red rectangles. (a) and (d) show the interface of visualizing histograms and selection of bins. (b) and (e) show the highlighting results on VesselMap, where red blocks contain particles fulfilling the selection criteria and green ones do not. (c) and (f) show the highlighting results of particles in the original 3D volume.

a Q-Q plot provides more information on how the two distributions differ from each other.

## 6.4 User Interface and Interaction

Our interface consists of three components: VesselMap, flow visualization, and information interface, as shown in Figure 6.2 on page 159. VesselMap displays the 2D representation of vessel structure. A blue bar is shown to indicate the inlet where particles are released, as shown in Figure 6.6 (b). When histogram or particle inlet information is displayed, the triangles are colored in red or green: red indicates blocks in the triangle containing queried particles, and green indicates blocks in the triangle containing no queried particles. When group comparison is displayed, the triangles are colored according to the groups they belong to. Flow visualization displays the vessel surface as a semi-transparent mesh and the flow inside the vessel, as shown in Figure 6.6 (c) and (d). Flow can be visualized as particles, streamlines or pathlines. Flow visualization is computed by the server, and only the resulting images are sent to the webpage. Users can select six orthogonal directions (i.e., head, feet, anterior, posterior, left and right) to view the data, or a predefined direction specified on the server. Information interface contains four tabs: cross filter, boolean filter, particle inlet visualization, and group comparison. Cross and boolean filters both contain histogram visualization, where users can interact with the histograms to filter the particles and highlight regions in the 2D VesselMap. Particle inlet visualization shows the particle distribution over the inlet plane that feeds the user-specified region.

In this application, all particles are released from an inlet plane specified by a user defined plane that cuts through a vessel branch. Note that, when referring to a property of particle, we use the term "particle inlet", or simply "inlet", to indicate the position where a particle is released in the seeding plane, instead of a specific inlet among multiple inlets. Group comparison demonstrates the differences among groups of blocks generated from the segmentation.

### 6.4.1 Histogram Visualization and Filtering

Our method visualizes statistical information of different properties, such as velocity, vorticity, WSS, particle inlet and pressure, in real time. Users can specify a region of interest so that the histogram will be generated from particles in that region. If it is not specified, the entire volume will be taken into consideration for histogram generation. As shown in Figure 6.6 (a) and (d), the histograms are shown as bar charts and the selection of a certain range of property values to further refine the histogram computation can be directly performed on those bar charts. This will be very helpful for domain experts, since finding the regions that meet certain criteria is a common task, e.g., finding the region with high pressure and low WSS. We provide two methods to specify the refining query criteria: *cross filter* and *boolean filter*.

For cross filter, users can mouse over multiple bins of a histogram, so the computation

of all other histograms will be restricted to these selected bins, meaning that only the particles falling into the selected ones are used to generate the histograms. If multiple histograms are selected, the computation of all non-selected histograms will be restricted by the selected ones; and any selected histogram will be restricted by other selected ones but not by itself. The first row of Figure 6.6 shows an example of cross filter. In (a), the bins corresponding to lower velocity and higher pressure are selected using cross filter. The blocks and particles are also highlighted according to the selected bins, as shown in (b) and (c). Those particles that belong to the selected bins are highlighted in red, while other particles are displayed in green.

Boolean filter behaves similarly. Nevertheless, instead of brushing across the bins, users need to click on multiple bins to specify the refining criteria. Considering the selection status of each bin as a boolean, the selected bins in the same histogram are connected by the *or* operator, while those in different histograms are connected by the *and* operator. The constraints on different histograms are the same as cross filter. The second row of Figure 6.6 demonstrates an example of boolean filter. Eight bins of inlet are selected, which means only particles released from these eight positions will be counted. In addition, a bin of age is selected to further constrain that only the particles with small age are counted.

Note that using cross filter, users can select a portion a bin, and the corresponding range will be interpolated. But they are not able to select two disjoint ranges for

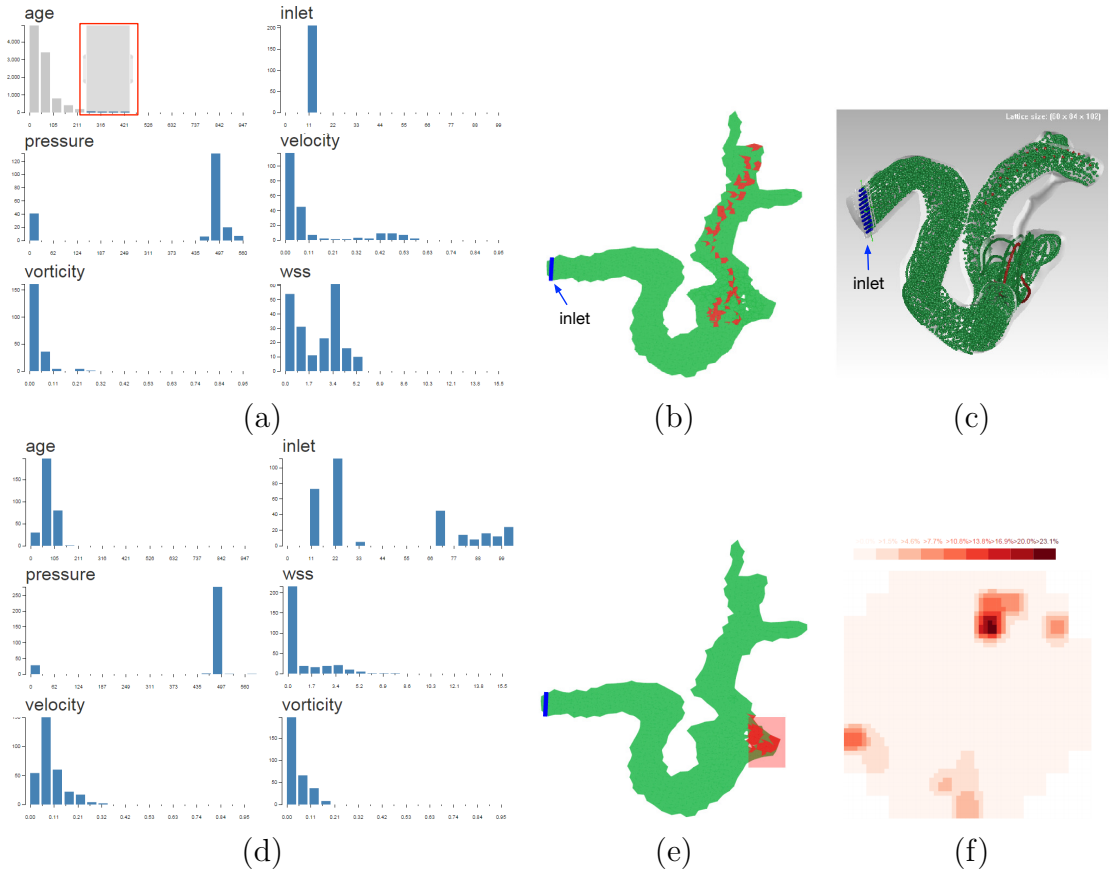
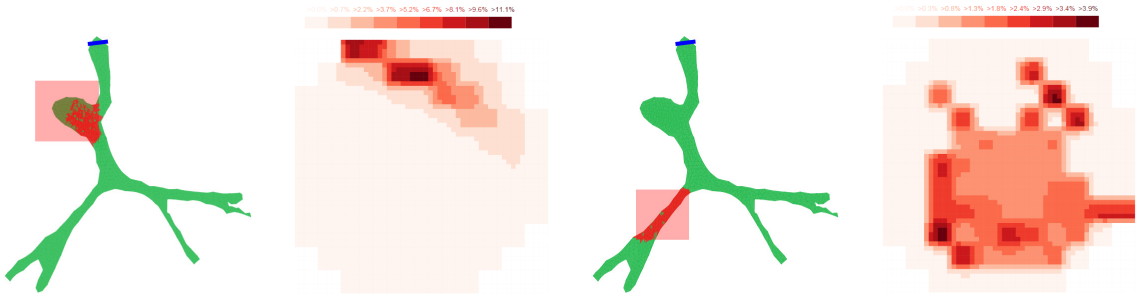



**Figure 6.7:** Visualization result using the VDS2. The first row shows the query of particles with large age. The second row shows the query regarding the aneurysm. (a) and (d) are histogram visualizations. (b) and (e) are VesselMap representations. (c) shows the queried particles with large age in the original 3D volume. (f) shows the inlet heat map associated with the aneurysm.

query. Using boolean filter, users can select disjoint ranges. But for each selected bin, the entire range of the bin is used.

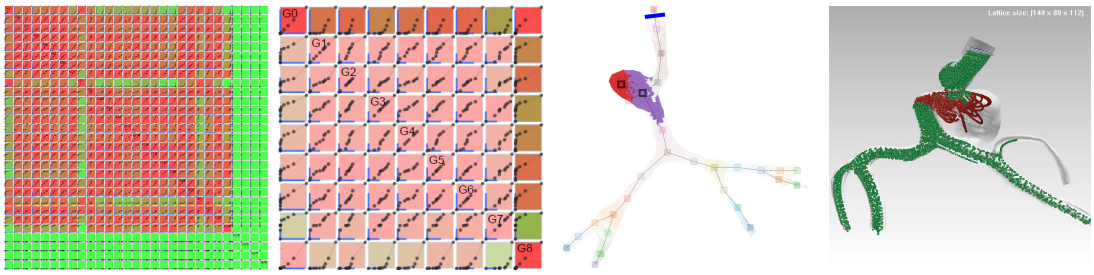


(a) (b) (c) (d)

**Figure 6.8:** Contributing particle distribution at a flow inlet. (a) shows the user-selected region on VesselMap. (b) is the heat map that indicates the number of contributing particles released from each position on the seeding plane. In the first row, the region of interest is the aneurysm. In the second row, the region of interest is a segment of a vessel branch highlighted in red.

### 6.4.2 Particle Inlet Visualization

Particle inlet visualization displays the number of contributing particles in a user-specified region for each inlet position using a heat map. The heat map is a commonly used graphical representation for a matrix, where each entry in the matrix is represented using color. In our case, each small color block on the heat map corresponds to a position on the inlet plane. This plane is displayed as a blue bar in VesselMap, as shown in Figure 6.7 (b). In flow visualization, the inlet plane is shown as a transparent box with white boundaries, and the particles on the inlet plane is colored in blue, as shown in Figure 6.7 (c). The color in the heat map indicates the number of particles originated from that specific region on the seeding plane which are currently in the region of interest. Users can interact with VesselMap to specify the region of interest, as shown in Figure 6.8. Note that some blocks in the selected region are




(a) (b) (c) (d)

**Figure 6.9:** The relationships among velocity distributions of groups of blocks segmented according to their distances to the inlet. (a) the difference matrix. (b) a zoom-in view focusing on the groups from $G0$ to $G8$. Cell $(G0, G8)$ is clicked, and row $G0$ and column $G8$ are highlighted. (c) and (d) the segmentation result and particle visualization with groups $G0$ and $G8$ highlighted.

still colored in green because these blocks do not contain any particle. Figure 6.8 shows two heat maps with respect to the aneurysm and a segment of vessel branch, respectively. This information might help medical experts identify the vessel branches that feed the aneurysm and the locations to release drugs that will be delivered to the desired location.

### 6.4.3 Group Comparison

In the segmentation, the vessel structure is divided into several groups of blocks guided by some property. The group comparison interface is designed to investigate into the differences of histograms among these groups. Note that the property for histogram computation is specified by users, and could be different from the one that guides the segmentation. In this way, the local behaviors and relationship between

properties can be better studied. The differences are organized in an $n \times n$ matrix, where each cell shows a difference value together with a Q-Q plot indicating how the two distributions differ. All groups are sorted by decreasing numbers of particles in them, and arranged left to right and top to bottom in the matrix. In Figure 6.9, (a) shows an example of the difference matrix interface. Users can use this interface to narrow down to some region of interest. By clicking a cell in the difference matrix, the row and column containing that cell will be highlighted, as shown in (b). In addition, the corresponding groups and particles will be highlighted in VesselMap and particle visualization as well, as shown in (c) and (d), respectively.

## 6.5 Results

The current segmentation is based on the average values computed at a certain time step. Otherwise, segmentation may change over time and the relationships among groups can be difficult to perceive. Thus, we only present the results using steady fields in this section, although our approach can be used on unsteady data. The two data sets we used are *vascular data set 1* (VDS1) and *vascular data set 2* (VDS2), with dimension of $108 \times 60 \times 84$ and $45 \times 63 \times 78$, respectively. The timing results were collected on a PC with an Intel Core i7-3820 CPU running at 3.6GHz, 16GB main memory, and an nVidia Geforce 670 graphics card with 2GB graphics memory. Even leveraging the power of GPU, the generation of VesselMap dominates the timing

cost, as it requires to solve multiple linear systems. For the VDS1 with 1095 blocks, the corresponding VesselMap took 55.8 seconds to compute. For the VDS2 with 1482 blocks, it took 137.6 seconds. We feel that this performance is still acceptable, since it only needs to be computed once for each data set. The histograms are updated using GPU, and can be performed in real time.

### 6.5.1 Case study: VDS1

We query for the particles with large pressure and small velocity values, as shown in the first row of Figure 6.6 on page 168. VesselMap and particle visualization both show that the queried particles mostly reside in the aneurysm and those regions where vessel branches bifurcate. We observe from the inlet histogram that most of these particles are released from the positions corresponding to four bins. In the first row of Figure 6.8, we investigate the contribution of each position on the seeding plane to the aneurysm. Note that some blocks in the selected region are still in green because they do not contain any particle. Particle inlet visualization indicates that most particles are released in a small region, which is consistent with the inlet histogram. In the second row of Figure 6.8, a segment of a vessel branch is selected, and the contribution of each position in the inlet plane is somewhat uniform. In the second row of Figure 6.6, a bin with small age and eight bins of inlet are selected. We observe the similar trend that the inlet positions corresponding to the particles with

small age values are mostly evenly distributed. It is also obvious that the velocities of these particles are more uniformly distributed and more particles with higher vorticity values are selected, compared with the histograms in the first row of Figure 6.6. From VesselMap, we also observe that most queried particles are in one vessel branch. This indicates that particles in this branch spend less time in the aneurysm.

In Figure 6.4 (b) on page 6.4, the segmentation result of the VDS1 guided by the vorticity shows that more groups are generated around the neck of the aneurysm and around an outlet of a vessel branch. This indicates that the average vorticity values of blocks in these regions vary more dramatically.

The segmentation guided by the distances to the inlet better preserves the shape of the vessel structure, as shown in Figure 6.4 (a). We use the difference matrix to analyze the differences of velocity distributions among groups. Note that the groups that do not contain any particle will be considered as entirely different from other groups, and the corresponding cells in the difference matrix will all be colored in green. In Figure 6.9 (a), we observe that most green cells correspond to groups $G0$, $G8$ and $G21$, except the empty groups. This indicates that the velocity in these three groups might be different from other groups. Clicking at cell $(G0, G8)$ highlights row $G0$ and column $G8$, as shown in Figure 6.9 (b). In cells $(G0, G0)$ and $(G8, G8)$, points in the Q-Q plot are denser in the smaller value range. Moreover, cells in row $G0$ and $G8$ show that the Q-Q plot lines lean to horizontal when comparing these

two groups with other groups. These facts imply that particles velocities in these two groups are smaller than the other groups. The corresponding groups and particles are highlighted accordingly when a cell in the difference matrix is selected. We can find that $G0$ and $G8$ cover the aneurysm region.

### 6.5.2 Case study: VDS2

The vessel structure of the VDS2 is shown in Figure 6.7 (b) and (c) on page 172. The first row of Figure 6.7 demonstrates the query of particles with large age. The inlet histogram shows that all particles with large age actually correspond to one bin, which means that they are released at a very small region on the seeding plane. From VesselMap and particle visualization, we observe that these particles go through the aneurysm region, and their paths are close to the center of the vessel branch. In addition, these particles mostly have small velocity values as expected. In the second row of Figure 6.7, we select the aneurysm region for query. We find that most particles in this region have moderate age values. This confirms that the particles with large age are mostly those that have already left the aneurysm. The inlet histogram shows that the particles that fill this region actually come from multiple inlet positions. This is observed in Figure 6.7 (f). However, comparing Figure 6.7 (a) and (d), we observe that the particles that stay in the aneurysm for a long time are mostly released from a specific location. From the velocity histograms, we find that the aneurysm region

even contains more particles with slightly higher velocity values than the particles with large age. It is likely that for the path near the centerline of the branch that leaves the aneurysm, the velocity is actually lower than other paths. The pressure and especially the WSS values mostly concentrate in a few histogram bins.

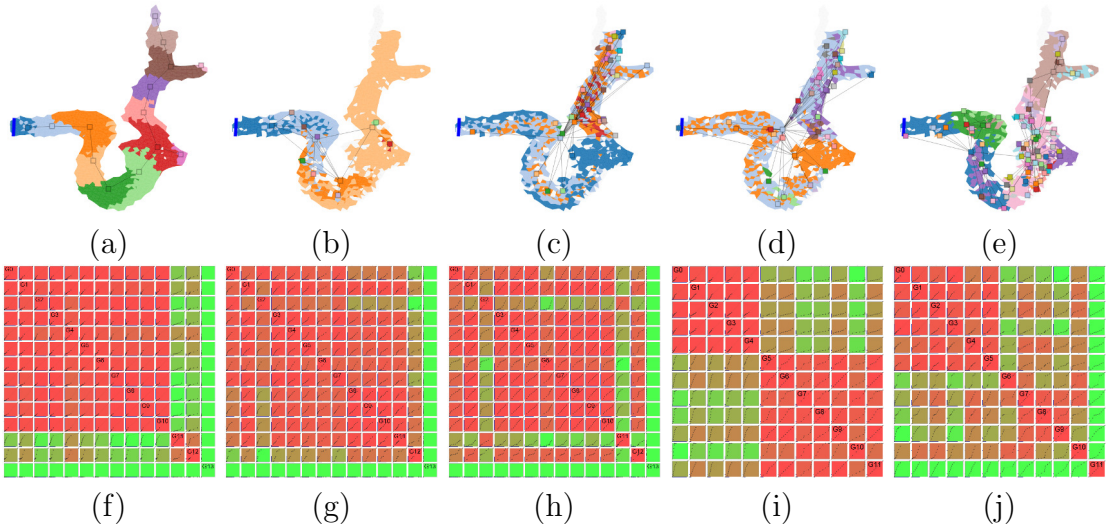



**Figure 6.10:** The segmentation result and group comparison using the VDS2. First row: segmentation result guided by (a) distances to the inlet, (b) age, (c) velocity, (d) vorticity, and (e) WSS. Second row: group comparison of (f) age, (g) velocity, and (h) vorticity among groups segmented by distance to the inlet; and of (i) age and (j) velocity among groups segmented by age.

In the first row of Figure 6.10, we show the segmentation result guided by different properties. In (a), using the distances to the inlet, we obtain the skeleton of the vessel structure. In (b), we see four large groups segmented by age, which are arranged similarly as the distances to the inlet. But there are also some small groups, where the particles are probably stuck and stay for a long time. In (c), we see that three large groups align along the main vessel branch. This indicates that there might be

several major paths with different velocities along the vessel. Note that this finding is consistent with our discussion of the histograms in Figure 6.7 on page 172. In (d), we notice that the average vorticity value varies in most of the vessel structure, but they are somewhat stable in the aneurysm and its neighborhood. In (e), we find that the WSS varies in the region near the aneurysm. However, it is very stable inside the aneurysm, since the aneurysm forms a single group. The histograms in Figure 6.7 also show the same results.

We then investigate the relationship among groups, as shown in the second row of Figure 6.10. For the groups segmented by the distance to the inlet, we find that the age distribution of groups $G11$ and $G12$ are different from other groups, as shown in (f). The age for most groups cover a large range, with their maximum values much larger than the other values. But the age in $G11$ is distributed somewhat uniformly in a smaller range. Note that $G11$ corresponds to the top of aneurysm region. This indicates that the particles in this region follow similar path before, and have similiar age. (g) shows that the velocity in $G2$ and $G12$ differs from other groups. We observe that $G2$ contains more particles with small velocity, which is common for an aneurysm. (h) shows that the aneurysm region corresponding to $G2$ and $G11$ contains more particles with small vorticity. For the groups segmented by age, we find that the groups form basically two clusters, considering the age distributions or the velocity distributions, as shown in (i) and (j), respectively. Groups $G0$ to $G4$ form one cluster, and the other groups form the other. Two groups from the same cluster have similar

age and velocity distributions, but two groups from different clusters are less similar. Note that the first cluster that consists of groups with more particles contains the four major groups in (b). It is likely that the isolated small groups have different average age values from their neighbors due to velocity difference.

## 6.6 Empirical Evaluation

To evaluate the effectiveness and learning difficulty of our approach, we collaborate with Dr. Jingfeng Jiang, a domain expert in biomechanics and biomedical imaging. Dr. Jiang research interests include predicting and analyzing bio-flows for cardiovascular diseases. This evaluation aimed at providing initial assessment on the usefulness of the proposed system instead of quantitative results. The tasks were only designed to guide the expert through the workflow of our approach. During the evaluation, Professor Jiang was informed to freely interact with our interface to explore the VDS1.

After learning the features of our interface, Professor Jiang explored the VDS1 using our interface and provided his comments. We organize and present his comments as the following. Overall, this is a great application with potential to impact flow visualization in a clinical setting. In addition, this application does not significantly differ from typical web applications. Thus, for ordinary users who are comfortable with web browsing, it should be easy to learn.

In terms of effectiveness, VesselMap allows users to quickly select a region of interest in 2D, because the flattening process will "evenly" space out the geometry. While in original 3D volumes, a user may need to rotate the geometry to find a position where only a small region on the projected plane is in the background or foreground. A technical concern is that the flattening process of VesselMap may cause unrelated 3D points to look like spatial neighbors. In terms of building connection between the 2D VesselMap and 3D vessel structure, users can brush across several branches on VesselMap, and observe the connection from highlighted region in the corresponding 3D particle visualization. However, since users may not be immediately aware of this, they should be forced to perform this task in their learning stage. In addition, some visual landmarks can be placed on both VesselMap and particle visualization, so that the connection can be perceived more easily.

Cross and boolean filters are very useful. Currently, everyone looks at flow data based on his or her own experience. Thus, there is a good chance to overlook some important features or correlations. In this application, once the queries are set up, users can use certain combinations to quickly identify areas of interest that meet all criteria. This helps to examine the flow characteristics within and around a cerebral aneurysm in a more comprehensive way.

The segmentation based on one criterion (e.g., distance, velocity, etc.) will enable the user to quickly grasp spatial distributions of the flow characteristics. Based on that,

it should help the user select some important regions of interest for further analysis. Some work may be needed to better organize or cluster the segmented regions. The difference matrix provides the overall correlation between different regions as well as a Q-Q plot. With the color-coded matrix, it is easy to see the correlation between two parameters such as age (particle residence time) and velocity. This along with the respective Q-Q plot, it will potentially help users identify regions which contain interesting flow features. For further development, it is possible that the combination of segmentation and difference matrix could be used as thumbnails to show more detailed flow features.

# Chapter 7

# Pedagogical Visualization Tools for Cryptographic Algorithms

## 7.1 Overview

Other than flow visualization, we also work on a variety of other visualization topics, among them is a series of pedagogical visualization tools for cryptographic algorithms[1]. *Cryptography* is fundamental to information security. Various aspects in information security heavily depend on crytography, including data confidentiality,

[1]The material contained in this chapter was previously presented in *Association for Computing Machinery, Inc. Reprinted by permission. Technical Symposium on Computer Science Education* [100, 101] and *Journal of Computing Science in Colleges* [102].

authentication and non-repudiation, etc. Many applications of cryptography are critically important nowadays, such as ATM cards, computer passwords, and electronic commerce. Due to the increasing needs in network/data security, the computer science (CS) education community started to add cryptography into CS curricula. However, modern cryptography resides at the intersection of multiple disciplines such as mathematics and computer science. Sophisticated mathematical theories are usually involved in cryptographic algorithms, which are challenging for CS students to understand.

To help students understand cryptographic algorithms in an intuitive way, we developed a suit of pedagogical visualization tools. These tools include DESvisual [102] for Data Standard Encryption algorithm, ECvisual [100] for Elliptic Curve based ciphers, RSAvisual [101] for the RSA algorithm, SHAvisual [69] for the Secure Hash Algorithm, VIGvisual [58] for the Vigenére cipher, and AESvisual for the Advanced Encryption Standard algorithm. All tools support Windows, MacOS and Linux. Each of them provides a Demo mode and a Practice mode for the corresponding algorithm. The Demo mode is useful for instructors to demonstrate important operations in the classroom, and the Practice mode is designed for self-study, where students can fill in the important intermediate results step by step.

In this chapter, we will discuss the designs and evaluations for DESvisual, ECvisual, and RSAvisual, since they are developed and evaluated by the author. Among these

three, ECvisual will be covered in more details, since the elliptic curve based ciphers are considered to be the currently most advanced one and involve more advanced mathematics. A brief introduction will be provided for the other projects.

### 7.1.1 SHAvisual, VIGvisual and AESvisual

#### 7.1.1.1 SHA and SHAvisual

Secure Hash Algorithm (SHA) is a family of cryptographic hash functions published by the National Institute of Standards and Technology. It was first introduced in 1993 and four series was developed over the past twenty years. SHAvisual was designed to help students learn the SHA-512 algorithm. The Demo mode of SHAvisual only provides a simplified SHA-512 visualization, which uses smaller size of data blocks for clearer demonstration. In addition to the Demo mode and the Practice mode, SHAvisual provides an additional Full mode, which shows the full version SHA-512 algorithm. A global view is displayed in a separate window to highlight the current procedure in the algorithm pipeline. The pipeline is organized in five subpages: Message Generation, Workflow Overview, Words Generation, Compression Function, and Round Detail.

#### 7.1.1.2 Vigenére Cipher and VIGvisual

The Vigenére cipher was named after Blaise de Vigenére, who described this cipher in 1586. It encrypts the text by using a series of Caesar ciphers based on the letters of a keyword agreed upon before communication. VIGvisual demonstrates the encryption and decryption using the Vigenére cipher and the attacks against it. The tabula recta, which is used to look up the ciphertext letter during encryption, is shown in three intuitive manners: table, ruler and disk. Animation can be enabled by users to go through the encryption or decryption letter by letter, with the current step highlighted in the tabula recta. The attacks against the Vigenére cipher are performed by keyword length estimation and recovery. VIGvisual uses Kasiski's method and the Index of Coincidence method for keyword length estimation, and the $\chi^2$ method for keyword recovery.

#### 7.1.1.3 AES and AESvisual

Advanced Encryption Standard (AES) is a specification for data encryption established by the National Institute of Standards and Technology. AES is based on the Rijndael cipher, a family of ciphers with different key and block sizes. The Demo mode consists of four subpages: Overview, Encryption, Decryption and Key Expansion, and the Practice mode is organized similarly. Each of the Encryption, Decryption and

Key Expansion is further divided into subpages for step-by-step computation results. The critical operations, including $\mathtt{GF}(2^8)$ addition and multiplication, can be further expanded and visualized in separate windows for more details.

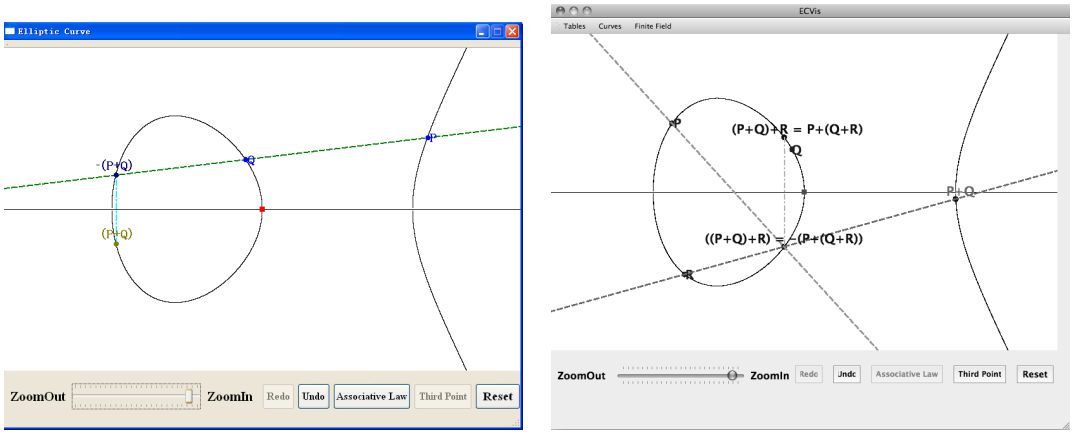
## 7.2 ECvisual: A Visualization Tool for Elliptic Curve Based Ciphers

Elliptic Curve Cryptography is built upon the algebraic structure of elliptic curves over finite fields. ECvisual was developed to facilitate the intuitive understanding of this cryptosystem, which involves many mathematical concepts related to elliptic curve. It demonstrates the addition law and the associative law of an elliptic curve over the real field and a finite field. Given any point in a finite field, it can show the subgroup of that point with a step-by-step demonstration on how each point in the subgroup is obtained. When an instructor introduces the elliptic curve version of the ElGamal cryptosystem, ECvisual can also be used to demonstrate the procedure to covert plaintext to a point on the elliptic curve. In addition to the benefits for demonstration, ECvisual has practice components built in to allow students to work with elliptic curves on their own. With ECvisual, students are able to practice adding points on the curve, converting plaintext to a point on the curve, encrypting, and decrypting points.

### 7.2.1 Software Overview

ECvisual has two subsystems, one over the real field and the other over a finite field of order $p$. Due to screen space limitation, $p$ is restricted to no more than 17. ECvisual has two operation modes: the demo mode and the practice mode. The demo mode may be used by instructors for classroom demonstration and by students to visualize the detail of computations. The practice mode is designed to help students go through the computations step-by-step and perform self-study. Thus, a student may use the practice mode to step through a computation procedure, fill in the answers, and check for correctness.

ECvisual has three pages, the Table page, the Curves page, and the Finite Field page. When ECvisual starts, the Table page is shown, and this is where the elliptic curve formula is set. The user defines a particular curve to work with by choosing appropriate parameters on this page (Section 7.2.3). The continuous elliptic curve is shown on the Curves page (Section 7.2.2). The Finite Field page illustrates the finite field over which an elliptic curve is defined (Section 7.2.3). This page also includes encryption and decryption (Section 7.2.4) and plaintext to elliptic curve point conversion algorithms (Section 7.2.5).




(a) (b)

**Figure 7.1:** The elliptic curve (a) group addition operator and (b) associative law (© 2012 Association for Computing Machinery, Inc. Reprinted by permission.)

### 7.2.2 The Elliptic Curve Group over Reals

This component provides the user with an opportunity to practice and visualize the elliptic curve group over the real number field. The user selects an elliptic curve by supplying the $a$ and $b$ in $y^2 = x^3 + ax + b$. Then, ECvisual draws the curve, allows the user to zoom in and out, selects two points P and Q, computes the intersection point of the line PQ and the curve (*i.e.*, $-$(P+Q)), and shows P+Q (Figure 7.1 (a)).

To visualize the associative law, the user clicks on the Associative Law button, picks three points P, Q and R, and ECvisual displays intermediate computations showing P+Q+R = (P+Q)+R = P+(Q+R) (Figure 7.1 (b)). Thus, the user should be able to easily learn the abstract idea via visualization.

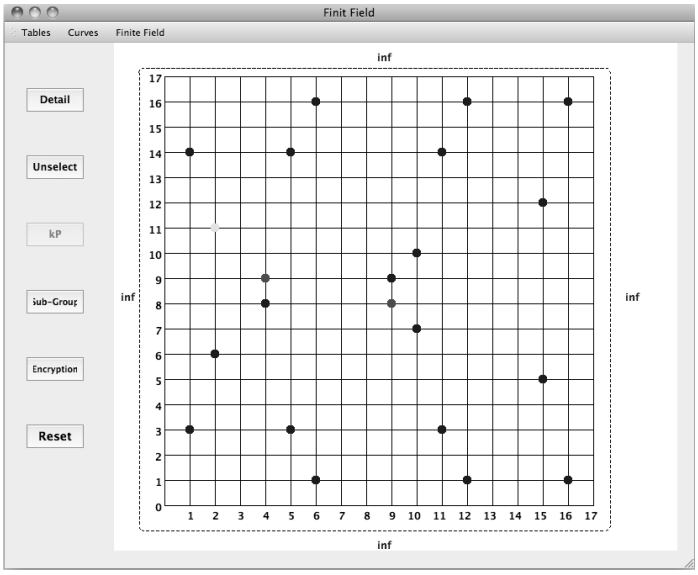



**Figure 7.2:** An elliptic curve over a finite field: $y^2 = x^3 + 3x + 5 \pmod{17}$ (© 2012 Association for Computing Machinery, Inc. Reprinted by permission.)

### 7.2.3 The Elliptic Curve Group over a Finite Field

The elliptic curve group over finite field $\mathbf{Z}_p$ of prime order $p$ component helps students visualize and practice elliptic curve computations over a finite field. The user supplies a prime number $p > 3$ and the parameters $a$ and $b$ in $y^2 = x^3 + ax + b$ (`mod` $p$), where $4a^3 + 27b^2 \neq 0$ (`mod` $p$) must hold. Then, the system displays a grid and all points on the curve with the identity element marked as inf at the center of each edge of the grid (Figure 7.2).

The user may click on the Table button to show the additive, multiplicative, and additive and multiplicative inverse tables of order $p$ (Figure 7.3).

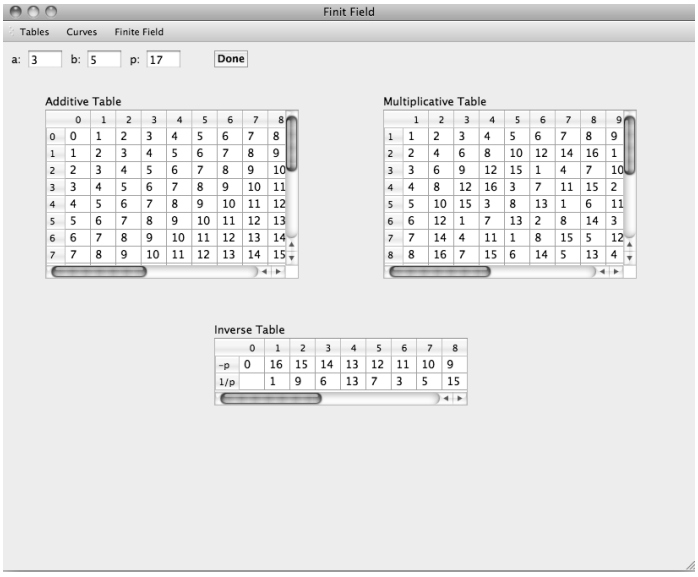



**Figure 7.3:** Addition, multiplication, and inverse tables over a finite field: $y^2 = x^3 + 3x + 5 \pmod{17}$ (© 2012 Association for Computing Machinery, Inc. Reprinted by permission.)

The Detail button (Figure 7.2) on the left panel brings up the **Detail Computation** window with which the user can practice computations on an elliptic curve. For example, the user may click on two points, which are shown in red and whose values are shown in the **Detail Computation** window as P and Q in the order of selection (Figure 7.4). Initially, all fields other than P and Q are blank. The user may choose Run, Step or Practice. The Run button asks the system to compute P + Q and displays all intermediate results such as $y_2 - y_1$, $x_2 - x_1$, the multiplicative inverse of $x_2 - x_1$ (*i.e.*, $(x_2 - x_1)^{-1}$), the slope $\lambda$ of the line PQ, the $x$-coordinate of the intersection point of line PQ and the elliptic curve, and the corresponding $y$-coordinate. The point P + Q is shown in yellow.

The user may select Step to step through the computation. In this case, the user fills

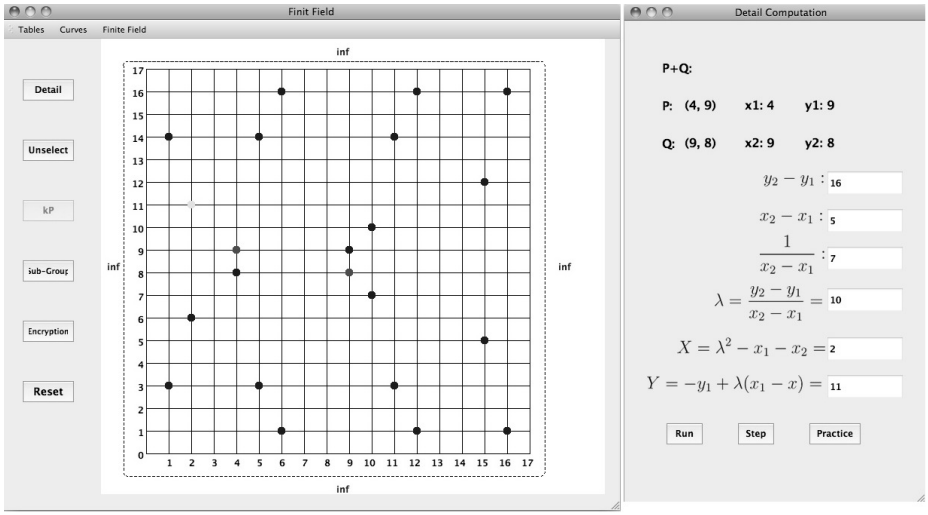



**Figure 7.4:** Compute $(2, 11) = (4, 9) + (9, 8)$ (© 2012 Association for Computing Machinery, Inc. Reprinted by permission.)

in the result one-by-one with the help of the computation tables (Figure 7.3), and the system verifies the input and displays **Correct!** if the answer is a correct one. The user may also select Practice and fill in *all* answers. The system then verifies all input and displays **Correct!** if all of them are correct. Incorrect answers are highlighted and **Wrong!** is displayed.

With this environment, the user may try to find a subgroup of prime order by repeatedly computing P, 2P = P+P, etc until $(n-1)$P is the identity. This can be performed by clicking on the kP button on the left panel. ECvisual is also able to find and display *all* subgroups of prime order by clicking on the Sub-Group button. Each subsequent click on the Sub-Group button will cycle through the prime order subgroups one-by-one. Thus, the user may step through these subgroups to choose an appropriate one for encryption. The preferable subgroup is the one with maximum

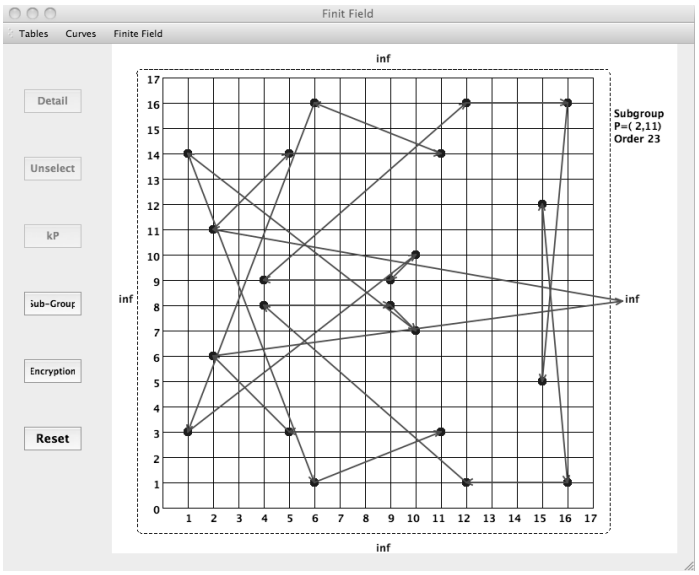



**Figure 7.5:** A subgroup of order 23 starting with $(2, 11)$: $y^2 = x^3 + 3x + 5 \pmod{17}$ (© 2012 Association for Computing Machinery, Inc. Reprinted by permission.)

prime order. Figure 7.5 shows a subgroup of order 23 with the edges showing the generation order of this subgroup.

### 7.2.4 Encryption and Decryption

Once a maximum prime order subgroup is found, the $p$ in $\mathbf{Z}_p$, the equation of the chosen elliptic curve, and the point $\mathsf{P}$ and its order $n$ are the public domain parameters. After this, the user may practice encryption and decryption easily by clicking on the Encryption button. This brings up the **Encryption & Decryption** window. For example, the user may select a private key $d$ randomly in the interval $[0, n-1]$ and compute the public key $\mathsf{Q} = d\mathsf{P}$. The sender represents the text by a point $\mathsf{M}$

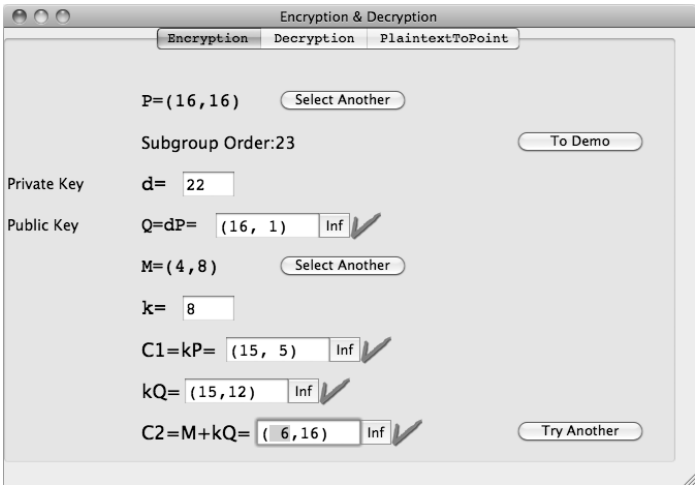



**Figure 7.6:** Encryption practice: $y^2 = x^3 + 3x + 5 \pmod{17}$ (© 2012 Association for Computing Machinery, Inc. Reprinted by permission.)

on the elliptic curve, selects randomly a number $k$ in $[1, n-1]$, computes $C_1 = k\mathsf{P}$ and $C_2 = \mathsf{M} + k\mathsf{Q}$, and sends $(C_1, C_2)$ to the recipient. The recipient uses her private key $d$ to compute $dC_1 = d(k\mathsf{P}) = k(d\mathsf{P}) = k\mathsf{Q}$, and, hence, recovers $\mathsf{M} = C_2 - k\mathsf{Q}$. In this way, elliptic curve encryption and decryption can be practiced easily with the visualization/practice system.

Figure 7.6 shows an encryption practice session. The system selects point $\mathsf{P}$ and a subgroup of maximum prime order, and allows the user to select a private key and fill in intermediate results. Again, the system will tell the user whether his/her computation is correct or wrong.

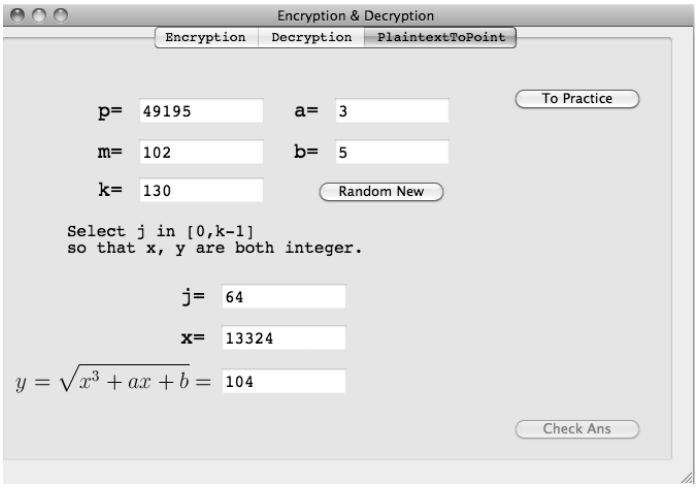



**Figure 7.7:** Plaintext to elliptic curve point (© 2012 Association for Computing Machinery, Inc. Reprinted by permission.)

### 7.2.5 Plaintext to Elliptic Curve Point

Converting a plaintext to a point on an elliptic curve is not very trivial and requires a larger $p$ to be "practical." Hence, this component is independent of the remaining components because large $p$ is impractical for visualization. ECvisual uses the Koblitz method [49]. Figure 7.7 shows a demonstration session of the Koblitz technique.

### 7.2.6 Evaluation

The ECvisual survey consists of two components, a set of nine questions and 13 write-in comments. The nine questions are listed in Table 7.1. Choices available are 5:strongly agree, 4:agree, 3:neutral, 2:disagree, and 1:strongly disagree. Because we intend to

study the impact of ECvisual on multiple disciplines, students were asked to fill in their disciplines. We collected 31 survey forms of which two were invalid. The distribution of majors is as follows: 3 in computer network and system administration (CNSA), 5 in computer and electrical engineering (CpE), 13 in computer science (CS), 5 in mathematics (Math), 1 in materials engineering, 1 in biological science, and 1 undeclared. The last three are grouped into the Other category.

**Table 7.1**
Survey questions for ECvisual

| Number | Question |
|---|---|
| Q1 | ECvisual's demo mode helped me understand what an elliptic curve is |
| Q2 | ECvisual's demo mode helped me understand how to represent plaintext as a point on an elliptic curve |
| Q3 | ECvisual's demo mode helped me understand how to encrypt and decrypt using elliptic curve version of the ElGamal cryptosystem |
| Q4 | ECvisual's demo mode was helpful for my self-study |
| Q5 | ECvisual's practice mode helped me understand how to add points on an elliptic curve |
| Q6 | ECvisual's practice mode helped me understand how to represent plaintext as a point on an elliptic curve |
| Q7 | I understand the elliptic curve version of the ElGamal cryptosystem more after I was able to use ECvisual |
| Q8 | By using ECvisual I was able to identify the parts of the elliptic curve version of the ElGamal cryptosystem that I do not understand |
| Q9 | ECvisual enhanced the course. |

#### 7.2.6.1 General Discussion

Table 7.2 and Figure 7.8 shows the mean and standard deviation of each question. In general, reactions to ECvisual are positive. The highest score of 4.2 with a small standard deviation of 0.5 was given to Q1, indicating that students agreed highly that ECvisual helped them understand what an elliptic curve is. Q4, Q5 and Q9 received the same score of 3.9, suggesting that ECvisual enhanced self-study and the course, and helped students understand the arithmetic on an elliptic curve. The remaining five questions were rated approximately the same (*i.e.*, 3.6 and 3.7) with slightly larger standard deviation. Thus, student reactions are mixed although the general trend is still in the positive side.

**Table 7.2**
Mean and standard deviation of survey questions for ECvisual

| | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 |
|---|---|---|---|---|---|---|---|---|---|
| Mean | 4.2 | 3.6 | 3.6 | 3.9 | 3.9 | 3.6 | 3.7 | 3.6 | 3.9 |
| S.Dev | 0.5 | 0.9 | 0.7 | 0.7 | 0.8 | 0.9 | 0.8 | 0.7 | 0.7 |

Correlations among student responses are high. The highest correlation is between Q2 and Q6 (0.87), which means students learned elliptic curve representations with the demo mode and practice mode. The lowest correlations 0.61 are between Q1 and Q8, and between Q1 and Q9. Overall, students answered questions in a rather similar pattern.

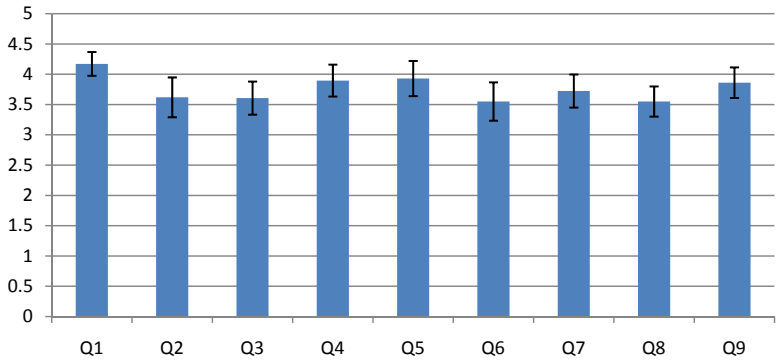



**Figure 7.8:** Mean and 95% confidence interval of survey questions for ECvisual

Table 7.3 shows the effect sizes (*i.e.*, Cohen's $d$ [3, 14]) among questions. We noted that the mean of Q1 (4.2) is very different from those of Q2, Q3, Q6 and Q8 (3.6) with effect sizes no less than 0.8. Additionally, effect sizes among Q2, Q3, Q6 and Q8 are very small. Therefore, student responses to Q2, Q3, Q6 and Q8 are nearly identical, and significantly different from responses to Q1. The effect sizes between Q1 and Q4 (0.44) and Q1 and Q5 (0.36) are moderate, indicating responses to Q1 and Q4, and those to Q1 and Q5 are moderately different. Moreover, the effect size between Q4 and Q5 is zero, suggesting students answered these two questions nearly identically. Consequently, students liked ECvisual for elliptic curve arithmetic and for self-study. It is interesting to point out that, except for Q1, effect sizes of Q8 and other questions are very small (0) to moderate (0.52). This indicates that except for Q1, students ratings of Q8 and other questions are not very different. The effect sizes between Q9

and Q1 and between Q9 and Q8 are moderate (0.5); but, effect sizes between Q9 and other questions are very small (0.05) to moderate (0.4). Hence, except for Q1 and Q8, students rated other questions similar to Q9.

**Table 7.3**
Effect sizes among questions for ECvisual

| | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 |
|---|---|---|---|---|---|---|---|---|
| Q1 | 0.80 | 0.89 | 0.44 | 0.36 | 0.87 | 0.70 | 1.02 | 0.51 |
| Q2 | | 0.02 | -0.30 | -0.40 | 0.08 | -0.10 | 0.09 | -0.30 |
| Q3 | | | -0.40 | -0.40 | 0.07 | -0.20 | 0.08 | -0.40 |
| Q4 | | | | -0.00 | 0.44 | 0.24 | 0.50 | 0.05 |
| Q5 | | | | | 0.46 | 0.27 | 0.52 | 0.09 |
| Q6 | | | | | | -0.20 | 0.00 | -0.40 |
| Q7 | | | | | | | 0.24 | -0.20 |
| Q8 | | | | | | | | -0.50 |

In summary, we found students felt that ECvisual helped them understand elliptic curves and their arithmetic, and also helped self-study.

#### 7.2.6.2 Discipline Specific Discussion

Because the class has students from more than five disciplines, it is very helpful to understand the differences among these groups, namely, CNSA, CpE, CS, Math, and Other. For each question, we studied the differences among these groups using ANOVA. Since the questions may be correlated, we also applied MANOVA (Multivariate ANOVA) to investigate the overall differences. For MANOVA, Wilk's Lambda test was used to consider all questions at the same time. The results are discussed

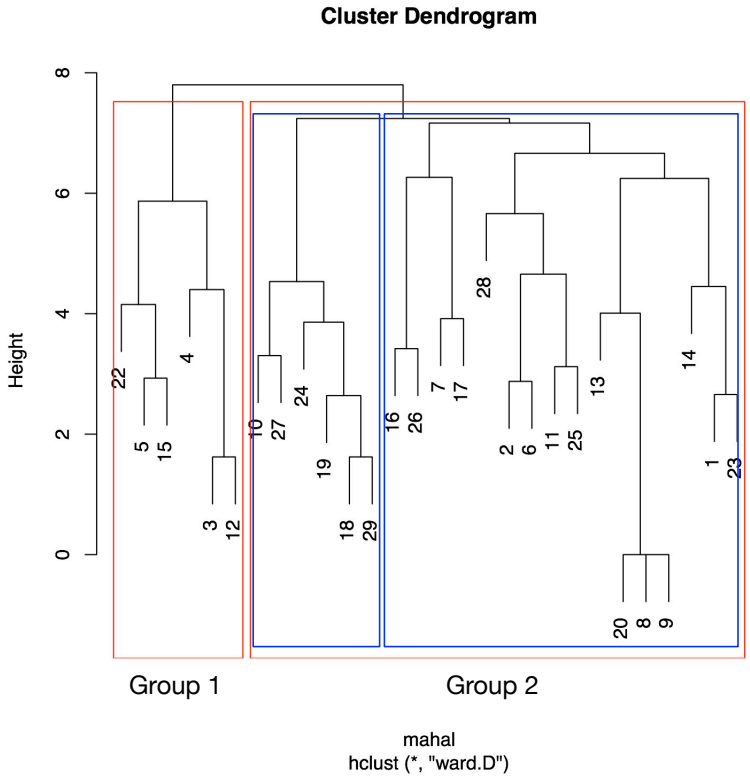



**Figure 7.9:** Cluster analysis result for ECvisual

at significance level of 0.05 as the following. We did not find significant difference for all questions using MANOVA, since the $p$-value was 0.79. No significant difference was found for any questions using ANOVA either. The smallest $p$-values were 0.17 for Q9 and 0.22 for Q7. The $p$-value for the other questions were all larger than 0.32. This suggests that the rating of students from different disciplines did not vary significantly.

To better understand the possible differences among students, we applied cluster analysis to group the students. The Ward's method was used for a hierarchical agglomerative clustering based on Mahalanobis distance. Figure 7.9 shows two groups

found by our cluster analysis, as highlighted in the two red rectangles. MANOVA showed that the two groups of students rated very differently with $p$-value of 0.000007. ANOVA for each individual question suggested significant differences for Q2 and Q4 with $p$-values of 0.027 and 0.020, respectively. The $p$-values for Q6 and Q9 were also small, being 0.071 and 0.075, respectively. The mean values of rating indicated that the students in Group 1 offered higher ratings than those in Group 2 for Q2 (3.86 against 3.00) and Q6 (3.72 against 3.00), but lower ratings for Q4 (3.72 against 4.50) and Q9 (3.72 against 4.33). Since Group 2 contained most of the students, we further grouped the students in Group 2, as highlighted in the two blue rectangles. No significant difference was found for all questions using MANOVA ($p$-value=0.31), or for individual question using ANOVA (the smallest $p$-value was 0.21 for Q3). This indicated that only the six students in Group 1 rated the questions differently from others. However, no clear and significant evidence, such as disciplines, could be found to explain the differences. It was more likely to be personal preference. This was also consistent with our findings from the results where students were grouped by their disciplines.

#### 7.2.6.3 Student Comments

The set of 13 write-in questions is designed to allow students to make suggestions which can be used for future development. We focus on the following issues: (1)

whether elliptic curves modulo $p$ for $p \leq 17$ is good enough, (2) whether the representation of the identity element (infinity) is intuitive, (3) whether the representation of subgroups of prime order is useful, (4) whether the elliptic curve version of the ElGamal cipher needs improvement, (5) the evaluation of the demo and practice modes, (6) frequency of using ECvisual for self-study, and (7) software installation problems.

Student comments showed that the $p \leq 17$ restriction is sufficient for understanding the concepts. Only a few mentioned $p$ should be much larger to be "realistic". However, this is impractical because screen asset is not enough for large $p$ visualization. One way to somewhat overcome this restriction would be adding a zooming capability and allowing the user to mouse over to see the details such as coordinates of a point.

There were no very negative comments on the design of ECvisual. Typical comments were "It is easy to use", "Perfect, except for the $p \leq 17$ thing", "Good design, easy to follow and very helpful in learning the system" and "Simple and to the point." Some issues were raised. Major ones were (1) should support $p > 17$ as mentioned earlier, (2) should use symbols and notations exactly the same as in the textbook, (3) finite field computation tables should always be visible and available rather than putting them under a tab, and (4) providing comments and descriptions for each step would be more user friendly and more convenient.

Students were very positive about the identity element and subgroups visualization. Some indicated "the identity element helped me understand exactly how infinity was

represented" while one student believed the identity element should only be above the top edge. Comments for the subgroups of prime order were nearly all positive. Students said "It was very clear and useful", "I like that it lights up all the dots, a very useful setup", and "I like being able to cycle through subgroups and orders". Again, some students wished to have step-by-step comments and descriptions so that they can follow the flow easier.

The practice mode was also very welcome with comments like "The practice mode is also good. If an answer is wrong there will be a big warning sign to inform you", "The practice mode helped check that you are doing the work correctly, so that is useful", "The practice mode works well and allows for some user interaction", and "Good to check answers. It is useful to be able to switch between practice and demo".

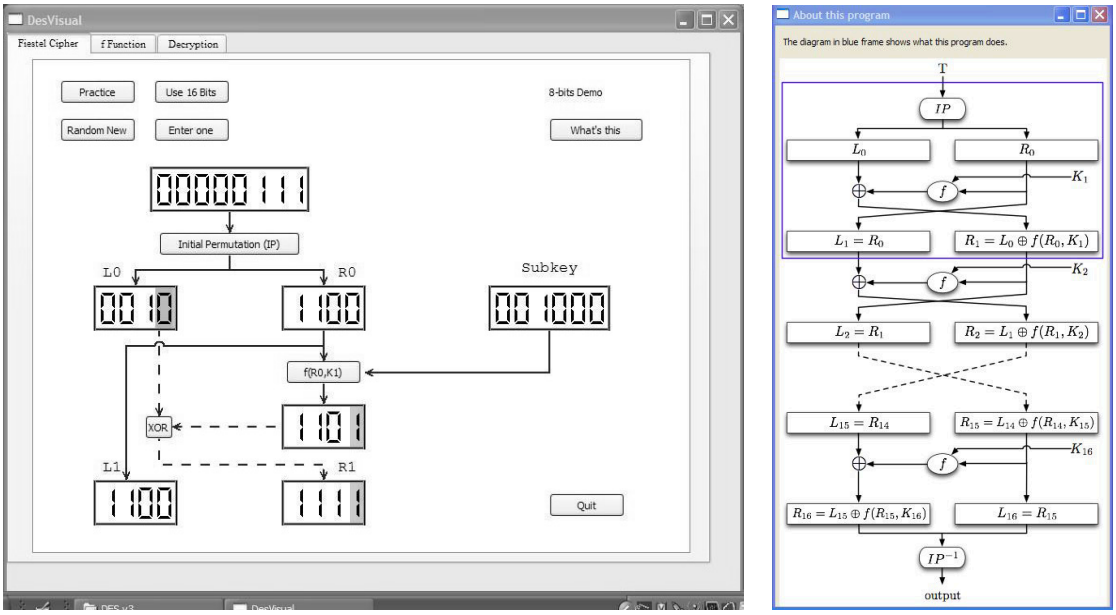
Because the students only had a week to play with ECvisual before taking this survey, the frequency of using this tool is not very high. Most of them used the tool a few times, and a few of them played with the tool "quite often". In general, they used ECvisual when they were solving problems, checking for some details, forgot the inner working of the algorithm, and used it for practice and further understanding. Since the encryption and decryption algorithms are simple once the concepts of finite fields and elliptic curves are understood, we are not surprised by the fact that a few students only used it once or twice, or did not use it at all. As a result, some students said this component is useful or somewhat useful. Students did not report any installation

issues, although three of them complained about system crashes.

In summary, with the statistics and student comments presented above, we believe ECvisual has fulfilled its purpose, helping students learn and the instructor teach the ElGamal cryptosystem based on elliptic curves over finite fields. With the comments and suggestions, we should be able to improve ECvisual significantly in the near future.

## 7.3 DESvisual: A Visualization Tool for the DES Cipher

The Data Encryption Standard (DES) is an encryption algorithm developed by IBM and published in 1977. It was the official data encryption standard from 1977 to 2000 and has been an important part of the field of cryptography. DESvisual helps students understand the building blocks of symmetric encryption. In particular, it depicts the primitive operations required to perform the initial permutation and one Feistel round of DES using an 8- or 16-bit input. A student can trace through an encryption performed by the tool, or can be guided through an encryption or decryption, computing the output of each operation herself. This helps students understand the primitive operations, how these operations are composed into the DES algorithm, and how functions and their composition are depicted and documented. The opportunity for self-study provides an instructor greater flexibility in selecting a




(a) (b)

**Figure 7.10:** DESvisual interface. (a) Main window - IP & Feistel Cipher. (b) Overview window. (© 2011 CCSC. Reprinted by permission.)

lecture pace over this detail-filled material.

### 7.3.1 Software Overview

DES encryption consists of an initial permutation (IP), sixteen Feistel rounds, and a final permutation. DESvisual visualizes an IP and one Feistel round. Figure 7.10 (a) depicts the Main window containing the IP and Feistel computation. The Overview window of Figure 7.10 (b) appears when the What's This button is clicked from the Main window; it shows the relationship between the tool computations and a full DES encryption. Computations are performed on either 8 or 16 bit inputs and 6 or

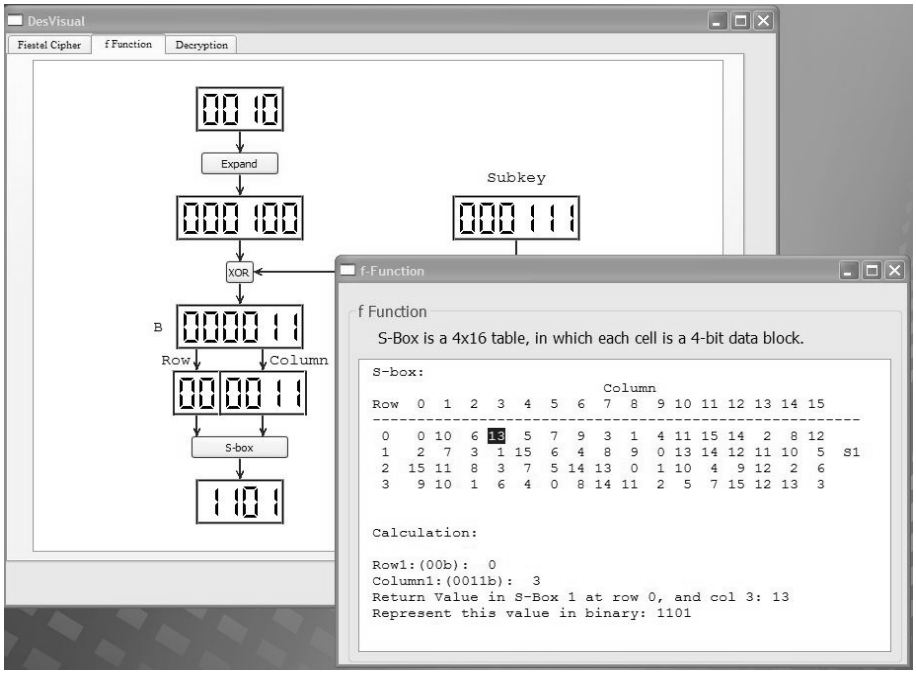



**Figure 7.11:** F function and S-box table (© 2011 CCSC. Reprinted by permission.)

10 bit keys (Use 16 bits in the Main window). A user may either have the system generate a random input and subkey (Random New) or enter their own input and subkey (Enter one). DESvisual provides Demo and Practice modes for both encryption and decryption procedure.

### 7.3.2 Demo

A user can trace an encryption (or decryption) by tracking a specific bit across each operation in the Demo mode. In this mode, the tool performs all computations. Clicking on a bit traces that bit across the operation from which it was derived. The

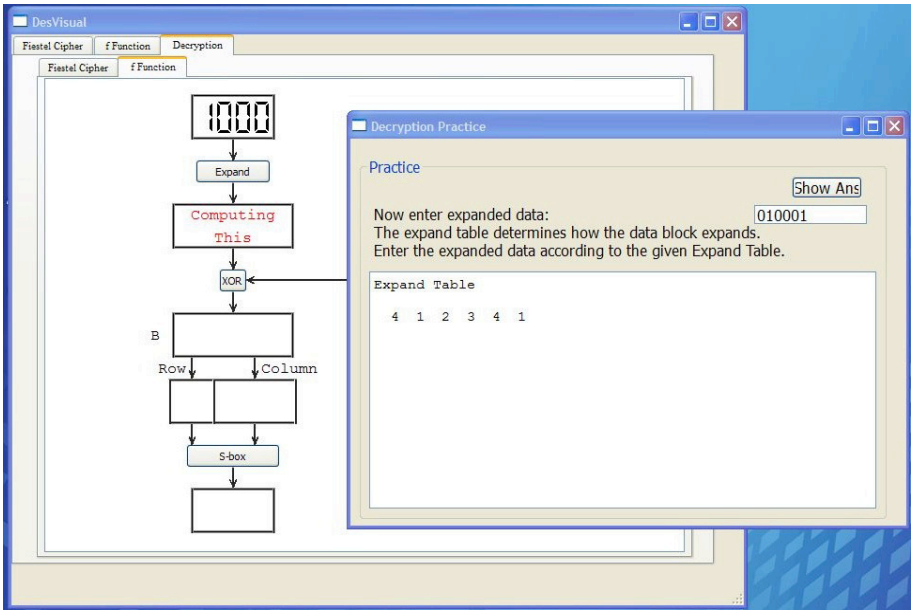



**Figure 7.12:** Practice mode for decryption (© 2011 CCSC. Reprinted by permission.)

dashed arrows and highlighted bits of Figure 7.10 (a) depict a bit trace. (These bits and arrows are highlighted in red in the tool.) The $f$ function is traced in a separate window that appears by pressing the f(R0,K1) button from the Main window. The $f$ function window is shown in Figure 7.11. Pressing the Initial Permutation or Expand buttons opens a window that contains the corresponding table. Depressing the S-box button causes the corresponding table to be displayed and the output element is highlighted. This is depicted in Figure 7.11.

### 7.3.3 Practice

The Practice mode of DESvisual consists of two guided mode for encryption and decryption. In each of these guided modes, the tool steps through each operation of the cipher and asks the student to compute the output from the current operation. A guided encryption begins when the Practice button is depressed from the Main window. A decryption begins by selecting the Decryption tab from the Main window. One step of a guided decryption is depicted in Figure 7.12. Input to the decryption is the output from the encryption calculation.

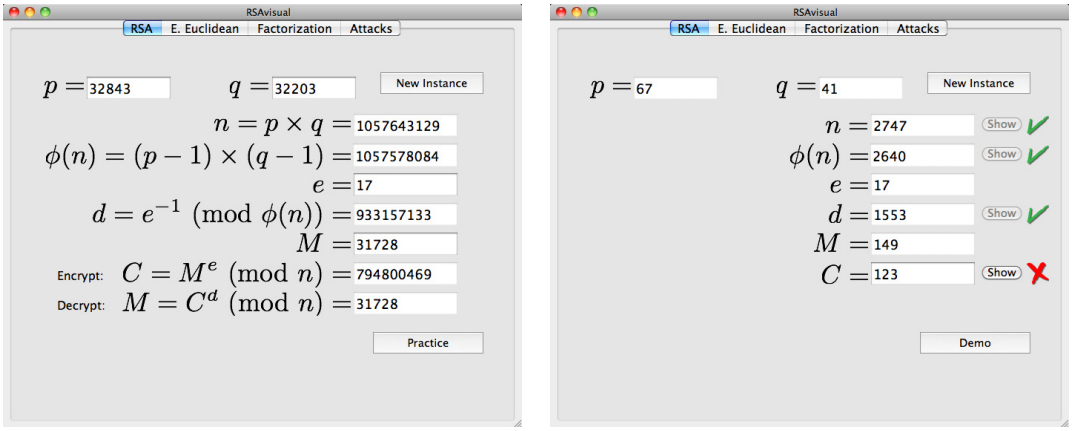
## 7.4 RSAvisual: A Visualization Tool for the RSA Cipher

RSA is a public-key cryptosystem that is widely used in data transmission. RSA is named after Ron Rivest, Adi Shamir and Leonard Adleman, who first described this algorithm in 1977. RSAvisual leverages visualization in order to meet this challenge for the RSA algorithm. It is designed to help students understand how the RSA algorithm operates, including encryption, decryption, use of the Extended Euclidean algorithm to calculate the private key, and Fermat and Pollard $p-1$ factorization.

RSAvisual is flexible in that it can be used for in-class demonstrations or it can be made available to students for self-study.

### 7.4.1 Software Overview

RSAvisual is designed to help students learn the RSA algorithm. It has four components: RSA, E. Euclidean (Extended Euclidean algorithm), Factorization and Attacks, each of which corresponds to a page in the system. The RSA component has a demo mode and a practice mode. The demo mode shows the details of the computations step by step and is useful in classroom demonstration. The practice mode allows the user to step through the computations, fill in the answers for each step and check for correctness. The two prime numbers $p$ and $q$ are restricted to 5-digit numbers in the demo mode for the user to easily follow the computation steps. Moreover, $p$ and $q$ are restricted to three digits in the practice mode so that the user can perform the computations by hand. RSAvisual always starts from the RSA page and the user can switch to other pages freely. The E. Euclidean page illustrates the use of the Extended Euclidean algorithm to calculate the inverse of a number. The Factorization page demonstrates how to factorize a number with Fermat's algorithm and Pollard's $p-1$ algorithm, respectively. The Attacks page has three elementary attacks on the RSA cryptosystem.




(a) (b)

**Figure 7.13:** (a) Demo mode and (b) Practice mode of the RSA page (© 2014 Association for Computing Machinery, Inc. Reprinted by permission.)

### 7.4.2 The RSA algorithm

The demo mode of the RSA component provides the user with an overall procedure of the RSA algorithm. Given two prime numbers $p$ and $q$, a public key $e$, and the plaintext $M$, it shows how $n$, $\phi(n)$ and the private key $d$ are computed. Two equations are displayed to show how a sender encrypts the plaintext with public key $e$ and the receiver decrypts that ciphertext with private key $d$ (Figure 7.13 (a)). The user can change the two prime numbers $p$ and $q$, the public key $e$ and the plaintext $M$, and the computation will be updated automatically. The user can also click the New Instance button to randomly generate a new set of $p$, $q$, $e$ and $M$.

In the practice mode, the user can step through the computation (Figure 7.13 (b)); however, all equations are hidden. In each step, a correct result is required to advance

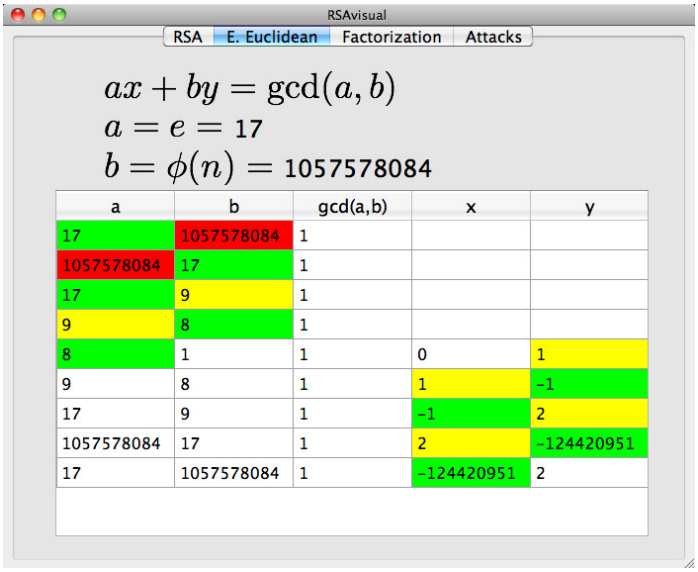



**Figure 7.14:** Computing the inverse of $e$ using the extended Euclidean algorithm (© 2014 Association for Computing Machinery, Inc. Reprinted by permission.)

to the next step. RSAvisual verifies the input and displays a green tick if the answer is correct. Otherwise, a red cross is shown so that the user can either enter a new value or skip a step by clicking the corresponding show button for the system to fill in the correct answer.

### 7.4.3 The Extended Euclidean algorithm

The E. Euclidean page demonstrates how to compute the inverse of a number using the Extended Euclidean algorithm. Given two integers $a$ and $b$, it illustrates the computation of $x$, $y$ and $\texttt{gcd}(a, b)$ in $ax + by = \texttt{gcd}(a, b)$, where $\texttt{gcd}(a, b)$ is the greatest common divisor of $a$ and $b$. The values of $a$ and $b$ are set to the public key

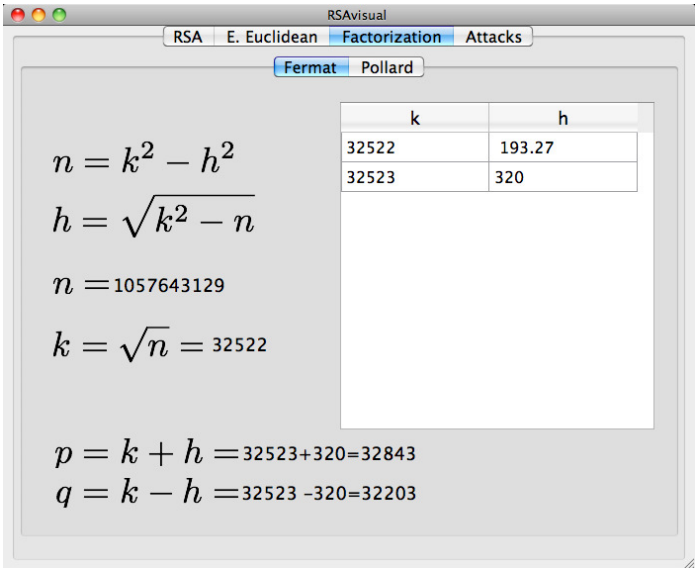



**Figure 7.15:** Factorizing $n$ using Fermat's algorithm (© 2014 Association for Computing Machinery, Inc. Reprinted by permission.)

$e$ and $\phi(n)$, respectively, so that $x$ gives the value of private key $d$. The computation is shown as a table in which each row represents the intermediate results for each step (Figure 7.14). Two cells of the same color in adjacent rows indicate that the lower one inherits the value from the upper one. The user follows the color of cells to trace the numbers across steps to learn how the values of $a$ and $b$ are exchanged between steps. The values of $x$ and $y$ are not filled bottom-up from where $\texttt{gcd}(a, b)$ is calculated. Instead, they are filled top-down so that it is easier for the user to follow.

### 7.4.4 Factorization

The Factorization component consists of the visualization of two factorization algorithms: Fermat's algorithm and Pollard's $p - 1$ algorithm. They are on two different

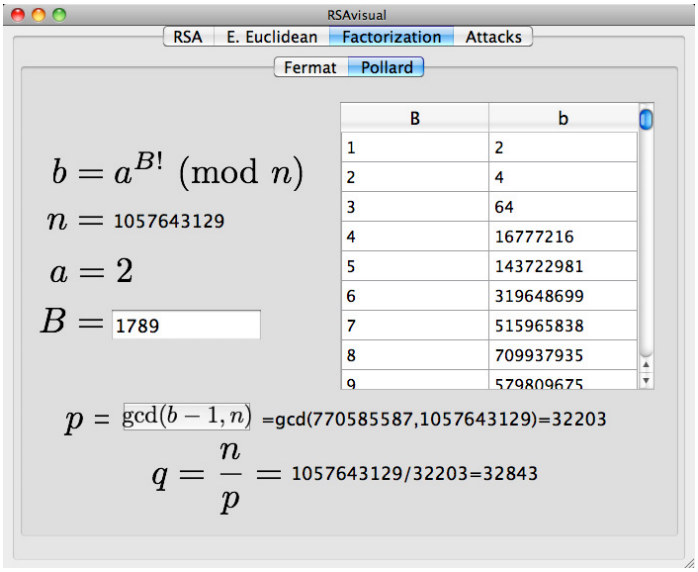



**Figure 7.16:** Factorizing $n$ using Pollard's algorithm (© 2014 Association for Computing Machinery, Inc. Reprinted by permission.)

sub-pages. Fermat's algorithm starts with $k = \lceil \sqrt{n} \rceil$. At each step, it calculates $h = \sqrt{k^2 - n}$. The values of $k$ and $h$ are recorded and displayed in a table for each step until $h$ is an integer. Finally, the values of $p$ and $q$ are given by $p = k + h$ and $q = k - h$, respectively (Figure 7.15).

The Pollard algorithm page illustrates how to factorize $n$ by computing $\texttt{gcd}(b-1, n)$, where $b = a^{B!} \pmod{n}$. When $B$ is large enough, $\texttt{gcd}(b-1, n)$ yields a non-trivial factor of $n$. The value of $B$ is initialized to be the smallest $B$ with such a property. The value of $B$ can be edited by the user, and RSAvisual will update the value of $\texttt{gcd}(b-1, n)$. In this way, the user will be able to discover that if $B$ is small we have $\texttt{gcd}(b-1, n) = 1$, and that only if $B$ is large enough $\texttt{gcd}(b-1, n)$ is a non-trivial factor. The values of $B$ and $b$ are listed in a table, so that the user can verify this property easily (Figure 7.16). Note that $\texttt{gcd}(b-1, n)$ is a button for the system to

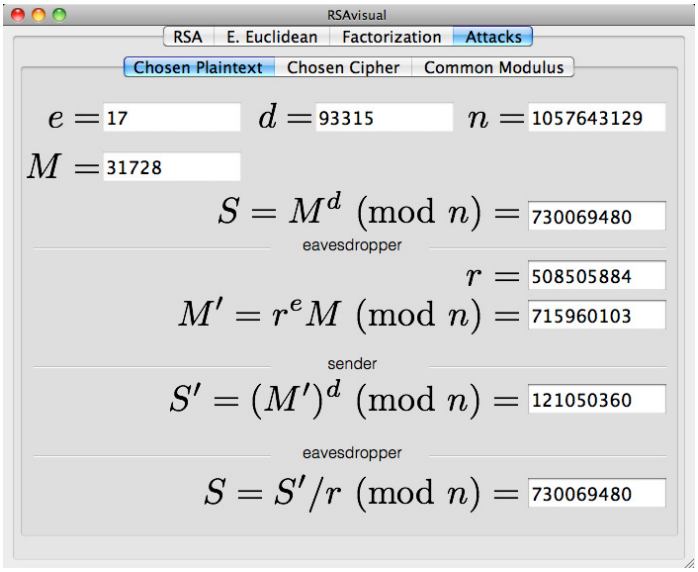



**Figure 7.17:** Chosen plaintext attack to forge the signature of the sender (© 2014 Association for Computing Machinery, Inc. Reprinted by permission.)

show the computation details.

### 7.4.5 Attacks

The Attacks component has three elementary attacks on the RSA cryptosystem: chosen plaintext attack, chosen ciphertext attack and common modulus attack [82, 132]. Each of these attacks occupies a sub-page on the Attacks page. The same values of $e$, $d$, $n$ and $M$ are used and interfaces are similar. For each attack, RSAvisual gives the initial conditions and then displays the attack operations in chronological order. The role of each operation (*i.e.*, sender, receiver and eavesdropper) is specified explicitly. Figure 7.17 demonstrates the interface of chosen plaintext attack.

# Chapter 8

# Conclusions

This dissertation focuses on the visualization and exploration of 3D flow fields using streamlines. A successful flow visualization should effectively assist users to obtain the information contained in their data effectively. It helps users observe the flow patterns, especially discover the feature patterns and locate them in flow fields. However, there are many challenges to overcome in order to achieve a clear observation of the desired features. First, this usually involves projecting the streamlines from 3D flow fields to a 2D plane. This projection may greatly reduce the information that can be perceived from a streamline. Even worse it often causes serious occlusions that hinder the flow pattern to be seen. Second, because the flow patterns of interest are usually application specific, an effective approach in one application may fail in another.

Our approaches improve the existing techniques and tackle these challenges in the following aspect. First, by considering the information of the 2D projection of a 3D streamline, we leverage an information channel to simultaneously select those informative streamlines under appropriate viewpoints to observe them. Second, we provide a finner level of control granularity that operates on streamline segments instead of entire streamlines, so that the densities in different regions can be manipulated more flexibly. Third, user interactions are highly involved to achieve visualization results depending on users' own needs.

## 8.1 Limitations and Future Extensions

While there are limitations in our current approaches, they also provide directions for future research.

First, we develop solutions that tackle the problem in different aspects, but less effort is spent on connecting these solutions. Each individual approach described in this dissertation focuses on one stage in the flow visualization pipeline. More precisely, the information channel is leveraged to select the most informative streamlines; FlowString queries the streamline segments of interest from a pool of streamlines; and the focus+context flow visualization customizes the visualization results in the rendering

stage. The current approaches are isolated and have different problems. The streamline selection provides a generally good pool for users to observe a flow field, but lacks of flexibility for users to investigate certain flow patterns. The focus+context approach requires users to specify a focus or automatically focuses on all features, but it is likely that users are not interested in all features and they are not aware of the locations of the features of interest. FlowString displays all the matched segments, but the contextual information is lost. It will provide greater value to combine our current approaches into a single visualization pipeline. The streamline selection can provide a better data base for FlowString to query, and the focus+context flow visualization can be used to enhance the query results and provide the missing contextual information.

Second, there are more characteristics of flow features to be discovered other than their shape. Our current approaches assist users to locate features based on the shape of streamline segments. In a later development of FlowString, we further allow users to filter the query results based on their scales. However, there are more ways to describe what the features are. For example, users may be interested in turbulent flows, flows that connect two critical points, flow with low velocity magnitudes, or flows related to features in associated scalar properties, etc. In other words, users may look for flows that share certain characteristics instead of flows of a certain shape. Therefore, the characterization of features should be high dimensional. Extending our current approaches to support the discovery of features with a more general description may

be worth future research efforts.

Third, our approaches focus on observing flow patterns, and have limited power in terms of revealing the relationship between a flow field and its associated scalar fields, which is often practically important. Flow fields are usually studied to discover the cause and formation of certain phenomenons/outcomes. For example, it is clinically beneficial to learn whether a certain pattern of flows will cause the change of pressure and wall shear stress and lead to the deformation of vessel wall, the formation and rupture of aneurysms. Currently, our approaches only support the exploration of the relationship among scalar fields and properties derived from flows (e.g., velocity and vorticity magnitudes), as described in Chapter 6. It is worth further investigating into developing the connection between a flow field and its possible outcome.

Finally, our current visualization style can be enriched. Through out this dissertation, the streamlines are drawn as opaque tubes. This may be because the major research interest is to facilitate effective exploration of flow patterns. However, we have to admit that a rich rendering may not only provide more visually pleasing visualization results, but also enhance knowledge discovery. For example, streamtapes with arrow heads are able to represent more flow features other than flow directions [5]. In some cases, we may find that using semi-transparent streamlines is helpful. The segments that are not matched in FlowString queries may be displayed semi-transparently. This allows the matched segments to be observed clearly and provides contextual

information at the same time.

## 8.2 Future Directions

Visualization is a rather diversified field where innovation plays an important role. There is no a single best way to visualize a flow field. In the future, we would like to investigate into the following directions.

### 8.2.1 Structure Discovery Using Persistent Topology

Persistent topology may be used to discover the structure of a flow field. The similarity measure of streamlines is extensively studied, as described in Section 2.3. However, little effort has been spent on utilizing the similarities among streamlines to reveal their underlying structure. Persistent topology starts from defining the local neighborhood relationship among data points based on their distances, i.e., two data points are considered to be neighbors if the distance between them is smaller than a threshold. By considering different distance thresholds simultaneously, the persistent topology provides a robust feature analysis of the hidden structure. In our case, we may sample a flow field using a number of streamlines. The similarity measure provides the distance between any two streamlines. The streamlines and the distances

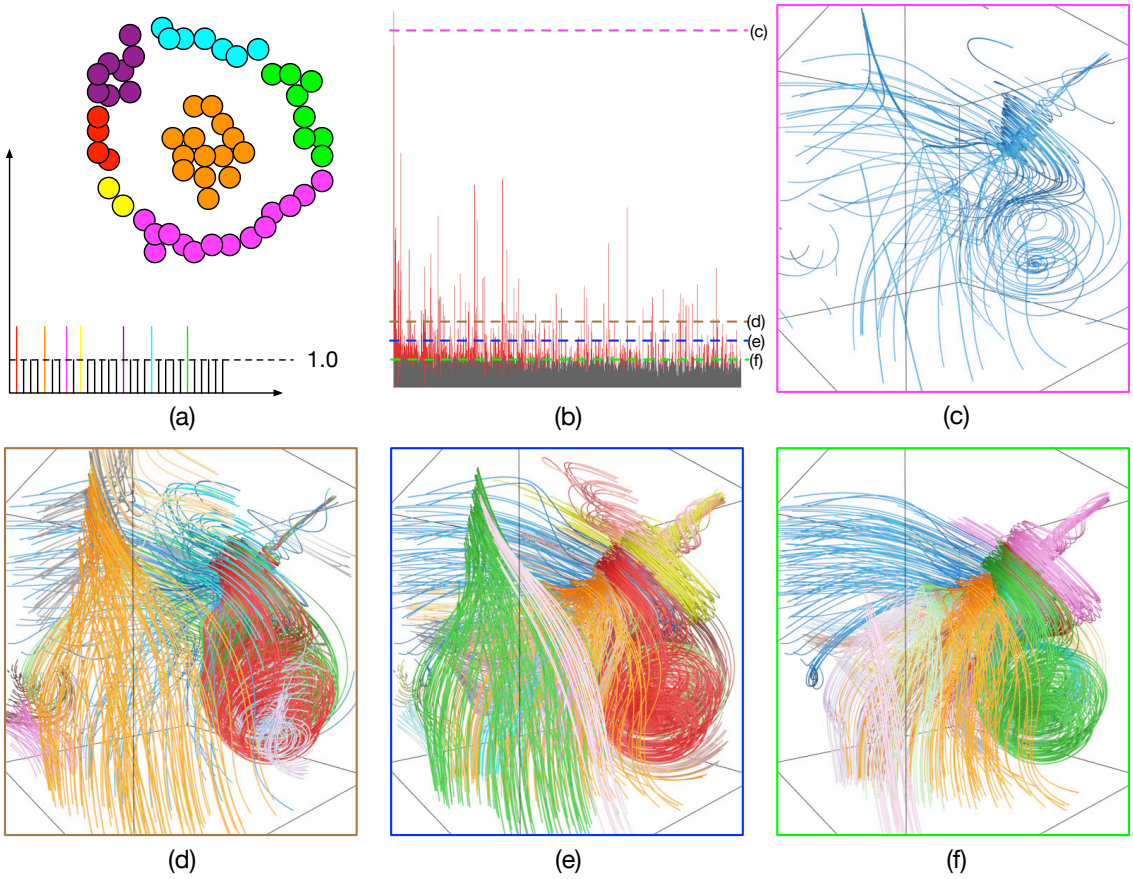



**Figure 8.1:** Streamline clustering based on persistence.

among them are then analyzed by the topological persistence algorithm [24] to discover the structures. Unlike the clustering algorithms, which only identify groups of streamlines, persistent topology is able to provide high dimensional features (e.g., tunnels and voids).

Figure 8.1 shows preliminary results in this direction. In (a), we illustrate how the 0-dimensional features (i.e., connected components) are determined given a threshold. We represent each streamline as a circle in a 2D plane. Assume that the distance between two circles represents the distance between the two corresponding streamlines,

and the size of a circle is equal to the given threshold. Then, two streamlines are considered to be connected if their corresponding circles overlap. To start with, each streamline is considered to be one component at the threshold of 0, since the circles do not overlap with each other with radius of 0. In this example, the streamlines form seven connected components at the threshold of 1.0, as shown in (a). The life spans of the components are shown by the barcode. At the threshold of 1.0, the life spans of the seven remaining components continue, and the life spans of others are terminated. The topological persistence algorithm provides an efficient way to compute the life span of each component. In (b), we show the life spans of components with 3000 streamlines. Note that the number of living components reduces with the increasing threshold. Figure (c), (d), (e) and (f) show the largest 20 components with decreasing thresholds, where the colors of the image borders correspond to the dashed lines representing the thresholds in (b). For a clearer observation, at most 100 streamlines will be shown for one component. In (c), all streamlines form one component with the largest threshold. The most prominent components (i.e., the ones with longer life spans than the others) are captured and shown in (d), (e) and (f). These components are mostly stable revealing similar structures, but also demonstrate the trend of being finer when the threshold decreases. Overall, it provides a robust analysis to determine persistent components using the life span barcode by considering all thresholds at the same time.

However, we also notice drawbacks of our current solution. Although it provides a

robust analysis over a given streamline pool, it is vulnerable to the samples (i.e., the streamlines in the pool). Adding a streamline may cause two components to be connected with a smaller threshold, while removing one streamline may delays the merge of two components. In the future, we plan to investigate two possible solutions. First, we may perform the persistent algorithm of several pools of streamlines and combine the resulting estimates, inspired by [9]. Second, we may define the neighborhood relationship on inter-connected streamline groups instead of individual streamlines, so that two streamline groups are connected if many of their members are connected. This works similarly as "averaging" the connections, which may reduce the sensitivity of sampled streamlines. In addition, we would like to take a further step to discover the higher dimensional structure using the persistent topology.

### 8.2.2 Graph-based Flow Visualization

Graph-based approaches may be applied to provide an abstract view to summarize the flow field and an effective interface to interact with. Graphs provide a proper solution to visual exploration of data sets in two aspects. For one thing, a graph can be used as the underlying data structure to capture the neighborhood relationships of elements, which is essential to data analysis. For another, a graph can be naturally visualized by drawing the nodes and edges. Graphs are not only concise representations of the data but also serve as convenient interfaces, as demonstrated

by our VesselMap approach (Chapter 6). In fact, graph-based approaches were previously developed for flow visualization [71, 72, 119, 131]. However, these approaches only represent streamline clusters and spatial regions as nodes, and use the edges to capture the spatial relationship among the nodes. As described in the previous section, because the description of features in flow fields is usually high dimensional, the spatial relationship is not enough to describe complicated data sets. We propose to use additional properties to construct the graph. The additional properties include the critical points in flow fields, features detected in the associated scalar fields, and other properties derived from flow fields. These properties may be used to generate new types of nodes, assign attributes to the existing nodes (e.g., nodes that represent streamlines), or redefine the relationship (i.e., edges). In this way, the more sophisticated relationship can be discovered on the graph and linked back to the original flow visualization for details. For example, a medical expert may find the connection between spiral flow patterns and low wall shear stress regions by observing the adjacency of corresponding nodes on the graph. By clicking those nodes, the corresponding regions and streamlines will be visualized for further analysis.

## 8.3 Conclusions

In this dissertation, we describe a series of visualization approaches toward the expressive exploration of flow fields. These approaches provide a fine level of control

over the streamlines, so that the visualization results are highly customizable. They focus on different aspects of flow visualization with high flexibility to fulfill different needs. Due to the relative independence of each approach, they can be incorporated into other approaches to provide a richer set of functionality. This suggests that these approaches not only provide useful tools, but also build up a solid foundation for future research.

# References

[1] A. Arbel and F. P. Ferrie. Viewpoint selection by navigation through entropy maps. In *Proceedings of International Conference on Computer Vision*, pages 248–254, 1999.

[2] U. D. Bordoloi and H.-W. Shen. View selection for volume rendering. In *Proceedings of IEEE Visualization Conference*, pages 487–494, 2005.

[3] M. Borenstein, L. V. Hedges, J. P. T. Higgins, and H. R. Rothstein. *Introduction to Meta-Analysis*. Wiley, 2009.

[4] M. Borkin, K. Gajos, A. Peters, D. Mitsouras, S. Melchionna, F. Rybicki, C. Feldman, and H. Pfister. Evaluation of artery visualizations for heart disease diagnosis. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2479–2488, 2011.

[5] S. Born, M. Markl, M. Gutberlet, and G. Scheuermann. Illustrative visualization of cardiac and aortic blood flow from 4D MRI data. In *Proceedings of IEEE*

*Pacific Visualization Symposium*, pages 129–136, 2013.

[6] A. Brun, H. Knutsson, H.-J. Park, M. E. Shenton, and C.-F. Westin. Clustering fiber traces using normalized cuts. In *Proceedings of International Conference on Medical Image Computing and Computer Assisted Intervention*, pages 368–375, 2004.

[7] L. Buatois, G. Caumon, and B. Lévy. Concurrent number cruncher - a GPU implementation of a general sparse linear solver. *International Journal of Parallel, Emergent and Distributed Systems*, 24(3):205–223, 2009.

[8] J. M. Chambers, W. S. Cleveland, P. A. Tukey, and B. Kleiner. *Graphical Methods for Data Analysis*. Duxbury Press, 1983.

[9] F. Chazal, B. T. Fasy, F. Lecci, B. Michel, A. Rinaldo, and L. Wasserman. Subsampling methods for persistent homology. *arXiv preprint arXiv:1406.1901*, 2014.

[10] C.-K. Chen, S. Yan, H. Yu, N. Max, and K.-L. Ma. An illustrative visualization framework for 3D vector fields. In *Computer Graphics Forum*, volume 30, pages 1941–1951, 2011.

[11] M. Chen and H. Jänicke. An information-theoretic framework for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1206–1215, 2010.

[12] W. Chen, S. Zhang, S. Correia, and D. S. Ebert. Abstractive representation and exploration of hierarchically clustered diffusion tensor fiber tracts. *Computer Graphics Forum*, 27(3):1071–1078, 2008.

[13] Y. Chen, J. D. Cohen, and J. H. Krolik. Similarity-guided streamline placement with error evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1448–1455, 2007.

[14] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Earlbaum Associates, Hillsdale, NJ, second edition, 1988.

[15] S. Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer, 2001.

[16] I. Corouge, S. Gouttard, and G. Gerig. Towards a shape model of white matter fiber bundles using diffusion tensor MRI. In *Proceedings of International Symposium on Biomedical Imaging*, pages 344–347, 2004.

[17] C. D. Correa, D. Silver, and M. Chen. Illustrative deformation for data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1320–1327, 2007.

[18] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, second edition, 2006.

[19] R. A. Crawfis and N. L. Max. Texture splats for 3D scalar and field visualization. In *Proceedings of IEEE Visualization Conference*, pages 261–267, 1993.

[20] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of ACM SIGKDD Conference*, pages 89–98, 2003.

[21] Z. Ding, J. C. Gore, and A. W. Anderson. Classification and quantification of neuronal fiber pathways using diffusion tensor MRI. *Magnetic Resonance in Medicine*, 49(4):716–721, 2003.

[22] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of Eurographics/IEEE TCVG Symposium on Visualization*, pages 239–248, 2003.

[23] H. Doleisch and H. Hauser. Interactive visual exploration and analysis of multivariate simulation data. *Computing in Science & Engineering*, 14(2):70–77, 2012.

[24] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511–533, 2002.

[25] P. Embrechts. *Modelling Extremal Events: for Insurance and Finance*, volume 33. Springer, 1997.

[26] M. Feixas, M. Sbert, and F. González. A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Transactions on Applied Perception*, 6(1), 2009.

[27] P. Felkel, R. Wegenkittl, and K. Bühler. Surface models of tube trees. In *Proceedings of Computer Graphics International*, pages 70–77, 2004.

[28] S. Fleishman, D. Cohen-Or, and D. Lischinski. Automatic camera placement for image-based modeling. *Computer Graphics Forum*, 19(2):101–110, 2000.

[29] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[30] A. L. Fuhrmann and M. E. Gröller. Real-time techniques for 3D flow visualization. In *Proceedings of IEEE Visualization Conference*, pages 305–312, 1998.

[31] G. W. Furnas. Generalized fisheye views. *ACM SIGCHI Bulletin*, 17(4):16–23, 1986.

[32] E. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. In *Proceedings of IEEE Symposium on Information Visualization*, pages 175–182, 2004.

[33] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH Conference*, pages 209–216, 1997.

[34] R. Gasteiger, M. Neugebauer, O. Beuing, and B. Preim. The FLOWLENS: A focus-and-context visualization approach for exploration of blood flow in cerebral aneurysms. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2183–2192, 2011.

[35] E. J. Gumbel. *Statistics of Extremes*. Columbia University Press, 1958.

[36] H. K. Hahn, B. Preim, D. Selle, and H.-O. Peitgen. Visualization and interaction techniques for the exploration of vascular structures. In *Proceedings of IEEE Visualization Conference*, pages 395–578, 2001.

[37] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[38] L.-W. He, M. F. Cohen, and D. H. Salesin. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In *Proceedings of ACM SIGGRAPH Conference*, pages 217–224, 1996.

[39] B. Heckel, G. H. Weber, B. Hamann, and K. I. Joy. Construction of vector field hierarchies. In *Proceedings of IEEE Visualization Conference*, pages 19–25, 1999.

[40] K. H. Höhne, B. Pflesser, A. Pommert, M. Riemer, R. Schubert, T. Schiemann, U. Tiede, and U. Schumacher. A realistic model of the inner organs from the visible human data. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 776–785, 2000.

[41] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[42] H. Jänicke, T. Weidner, D. Chung, R. S. Laramee, P. Townsend, and M. Chen. Visual reconstructability as a quality metric for flow visualization. *Computer Graphics Forum*, 30(3):781–790, 2011.

[43] G. Ji and H.-W. Shen. Dynamic view selection for time-varying volumes. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1109–1116, 2006.

[44] R. Jianu, Ç. Demiralp, and D. H. Laidlaw. Exploring 3D DTI fiber tracts with linked 2D representations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1449–1456, 2009.

[45] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing*, pages 43–56, 1997.

[46] A. Kanitsar, D. Fleischmann, R. Wegenkittl, P. Felkel, and M. E. Gröller. CPR - curved planar reformation. In *Proceedings of IEEE Visualization Conference*, pages 37–44, 2002.

[47] A. Kanitsar, D. Fleischmann, R. Wegenkittl, and M. E. Gröller. Diagnostic relevant visualization of vascular structures. *Scientific Visualization: The Visual Extraction of Knowledge from Data*, pages 207–228, 2006.

[48] A. Kanitsar, R. Wegenkittl, D. Fleischmann, and M. E. Gröller. Advanced curved planar reformation: Flattening of vascular structures. In *Proceedings of IEEE Visualization Conference*, pages 43–50, 2003.

[49] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, January 1987.

[50] N. Koblitz. *Algebraic Aspects of Cryptography*. Springer-Verlag, 1998.

[51] B. Köhler, R. Gasteiger, U. Preim, H. Theisel, M. Gutberlet, and B. Preim. Semi-automatic vortex extraction in 4D PC-MRI cardiac blood flow data using line predicates. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2773–2782, 2013.

[52] J. Kretschmer, B. Preim, and M. Stamminger. Bilateral depth filtering for enhanced vessel reformation. In *Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 117–126, 2014.

[53] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[54] E. LaMar, B. Hamann, and K. I. Joy. A magnification lens for interactive volume visualization. In *Proceedings of Pacific Graphics Conference*, pages 223–232, 2001.

[55] R. S. Laramee, C. Garth, H. Doleisch, J. Schneider, H. Hauser, and H. Hagen. Visual analysis and exploration of fluid flow in a cooling jacket. In *Proceedings of IEEE Visualization Conference*, pages 623–630, 2005.

[56] G. Läthé, S. Lindholm, R. Lenz, A. Persson, and M. Borga. Automatic tuning of spatially varying transfer functions for blood vessel visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2345–2354, 2012.

[57] T.-Y. Lee, O. Mishchenko, H.-W. Shen, and R. Crawfis. View point evaluation and streamline filtering for flow visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 83–90, 2011.

[58] C. Li, J. Ma, J. Tao, J. Mayo, C.-K. Shene, M. Keranen, and C. Wang. VIGvisual: A visualization tool for the vigenére cipher. In *Proceedings of ACM Technical Symposium on Computer Science Education*, pages 129–134, Vilnius, Lithuania, 2015.

[59] L. Li, H.-H. Hsieh, and H.-W. Shen. Illustrative streamline placement and visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 79–86, 2008.

[60] L. Li and H.-W. Shen. Image-based streamline generation and rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):630–640, 2007.

[61] Y. Li, S. Carr, J. Mayo, C.-K. Shene, and C. Wang. Dtevisual: a visualization

system for teaching access control using domain type enforcement. *Journal of Computing Sciences in Colleges*, 28(1):125–132, 2012.

[62] Y. Li, C. Wang, and C.-K. Shene. Extracting flow features via supervised streamline. *Computers & Graphics*, 52:79–92.

[63] Y. Li, C. Wang, and C.-K. Shene. Streamline similarity analysis using bag-of-features. In *IS&T/SPIE Electronic Imaging*, pages 90170N–90170N, 2013.

[64] A. Light and P. J. Bartlein. The end of the rainbow color schemes for improved data graphics. *EOS Transactions of the American Geophysical Union*, 85(40), 2004.

[65] T. Lindeberg. Feature detection with automatic scale selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):234–254, 1990.

[66] Z. Liu, R. J. Moorhead, and J. Groner. An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):965–972, 2006.

[67] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of ACM SIGGRAPH Conference*, pages 163–169, 1987.

[68] K. Lu, A. Chaudhuri, T.-Y. Lee, H.-W. Shen, and P. C. Wong. Exploring

vector fields with distribution-based streamline analysis. In *Proceedings of IEEE Pacific Visualization Symposium*, 2013.

[69] J. Ma, J. Tao, M. Keranen, J. Mayo, C.-K. Shene, and C. Wang. SHAvisual: A secure hash algorithm visualization tool. In *Proceedings of American Society for Engineering Education Annual Conference*, Seattle, WA, 2015.

[70] J. Ma, C. Wang, and C.-K. Shene. Coherent view-dependent streamline selection for importance-driven flow visualization. In *Proceedings of IS&T/SPIE Conference on Visualization and Data Analysis*, 2013.

[71] J. Ma, C. Wang, and C.-K. Shene. FlowGraph: A compound hierarchical graph for flow field exploration. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 233–240, 2013.

[72] J. Ma, C. Wang, C.-K. Shene, and J. Jiang. A graph-based interface for visual analytics of 3D streamlines and pathlines. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1127–1140, 2014.

[73] M. Maddah, W. E. Grimson, S. K. Warfield, and W. M. Wells. A unified framework for clustering and quantitative analysis of white matter fiber tracts. *Medical Image Analysis*, 12(2):191–202, 2008.

[74] S. Marchesin, C.-K. Chen, C. Ho, and K.-L. Ma. View-dependent streamlines for 3D vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1578–1586, 2010.

[75] O. Mattausch, T. Theußl, H. Hauser, and M. E. Gröller. Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. In *Proceedings of Spring Conference on Computer graphics*, pages 213–222, 2003.

[76] T. McLoughlin, M. W. Jones, R. S. Laramee, R. Malki, I. Masters, and C. D. Hansen. Similarity measures for enhancing interactive streamline seeding. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):1342–1353, 2013.

[77] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010.

[78] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. In *Proceedings of IEEE Visualization Conference*, pages 479–486, 2005.

[79] G. Mistelbauer, A. Morar, A. Varchola, R. Schernthaner, I. Baclija, A. Köchl, A. Kanitsar, S. Bruckner, and M. E. Gröller. Vessel visualization using curvicircular feature aggregation. 32(3):231–240, 2013.

[80] G. Mistelbauer, A. Varchola, H. Bouzari, J. Starinsky, A. Köchl, R. Schernthaner, D. Fleischmann, M. E. Gröller, and M. Sramek. Centerline reformations of complex vascular structures. In *IEEE Pacific Visualization Symposium*, pages 233–240, 2012.

[81] B. Moberts, A. Vilanova, and J. J. van Wijk. Evaluation of fiber clustering methods for diffusion tensor imaging. In *Proceedings of IEEE Visualization Conference*, pages 65–72, 2005.

[82] R. A. Mollin. *RSA and Public-Key Cryptography*. Chapman & Hall/CRC, 2003.

[83] S. Oeltze, D. J. Lehmann, A. Kuhn, G. Janiga, H. Theisel, and B. Preim. Blood flow clustering and applications invirtual stenting of intracranial aneurysms. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):686–701, 2014.

[84] S. Oeltze and B. Preim. Visualization of vasculature with convolution surfaces: Method, validation and evaluation. *IEEE Transactions on Medical Imaging*, 24(4):540–548, 2005.

[85] O. Rosanwo, C. Petz, S. Prohaska, H.-C. Hege, and I. Hotz. Dual streamline seeding. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 9–16, 2009.

[86] C. Rössl and H. Theisel. Streamline embedding for 3D vector field exploration. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):407–420, 2012.

[87] M. Ruiz, I. Boada, M. Feixas, and M. Sbert. Viewpoint information channel for illustrative volume rendering. *Computers & Graphics*, 34(4):351–360, 2010.

[88] T. Salzbrunn and G. Scheuermann. Streamline predicates. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1601–1612, 2006.

[89] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of ACM SIGCHI Conference*, pages 83–91, 1992.

[90] M. Schlemmer, M. Heringer, F. Morr, I. Hotz, M.-H. Bertram, C. Garth, W. Kollmann, B. Hamann, and H. Hagen. Moment invariants for the analysis of 2D flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1743–1750, 2007.

[91] M. Schlemmer, I. Hotz, B. Hamann, F. Morr, and H. Hagen. Priority streamlines: A context-based visualization of flow fields. In *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization*, pages 227–234, 2007.

[92] C. Schumann, S. Oeltze, R. Bade, B. Preim, and H.-O. Peitgen. Model-free surface visualization of vascular trees. In *Proceedings of Joint Eurographics - IEEE VGTC Symposium on Visualization*, pages 283–290, 2007.

[93] K. Shi, H. Theisel, H.-C. Hege, and H.-P. Seidel. Path line attributes - an information visualization approach to analyzing the dynamic behavior of 3D time-dependent flow fields. In *Proceedings of Topology-Based Methods in Visualization*, pages 75–88, 2007.

[94] M. Shojima, M. Oshima, K. Takagi, R. Torii, M. Hayakawa, K. Katada,

A. Morita, and T. Kirino. Magnitude and role of wall shear stress on cerebral aneurysm computational fluid dynamic study of 20 middle cerebral artery aneurysms. *Stroke*, 35(11):2500–2505, 2004.

[95] G. Singh, F. Memoli, and G. Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *Eurographics Symposium on Point-Based Graphics*, pages 91–100, 2007.

[96] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188, 2004.

[97] B. Spencer, R. S. Laramee, G. Chen, and E. Zhang. Evenly spaced streamlines for surfaces: An image-based approach. *Computer Graphics Forum*, 28(6):1618–1631, 2009.

[98] M. Straka, A. Köchl, M. Červeňanský, M. Šrámek, D. Fleischmann, A. La Cruz, and M. E. Gröller. The VesselGlyph: Focus & context visualization in CT-angiography. In *Proceedings of IEEE Visualization Conference*, pages 385–392, 2004.

[99] S. Takahashi, I. Fujishiro, Y. Takeshima, and T. Nishita. A feature-driven approach to locating optimal viewpoints for volume visualization. In *Proceedings of IEEE Visualization Conference*, pages 495–502, 2005.

[100] J. Tao, J. Ma, M. Keranen, J. Mayo, and C.-K. Shene. ECvisual: a visualization tool for elliptic curve based ciphers. In *Proceedings of the ACM Technical Symposium on Computer Science Education*, pages 571–576, 2012. http://doi.acm.org/10.1145/2157136.2157298.

[101] J. Tao, J. Ma, M. Keranen, J. Mayo, C.-K. Shene, and C. Wang. RSAvisual: A visualization tool for the RSA cipher. In *Proceedings of ACM Technical Symposium on Computer Science Education*, pages 635–640, Atlanta, GA, 2014. http://doi.acm.org/10.1145/2538862.2538891.

[102] J. Tao, J. Ma, J. Mayo, C.-K. Shene, and M. Keranen. DESvisual: a visualization tool for the DES cipher. *Journal of Computing Sciences in Colleges*, 27(1):81–89, 2011.

[103] J. Tao, J. Ma, C. Wang, and C.-K. Shene. A unified approach to streamline selection and viewpoint selection for 3D flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):393–406, 2013.

[104] J. Tao, C. Wang, and C.-K. Shene. Flowstring: Partial streamline matching using shape invariant similarity measure for exploratory flow visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 9–16, Yokohama, Japan, 2014.

[105] J. Tao, C. Wang, C.-K. Shene, and S. H. Kim. A deformation framework

for focus+context flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 20(1):42–55, 2014.

[106] J. Tao, C. Wang, C.-K. Shene, and R. A. Shaw. A vocabulary approach to partial streamline matching and exploratory flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2015. Accepted.

[107] C. Teitzel, R. Grosso, and T. Ertl. Efficient and reliable integration methods for particle tracing in unsteady flows on discrete meshes. In *Proceedings of Eurographics Workshop on Visualization in Scientific Computing*, pages 31–41, 1997.

[108] A. Telea and J. J. van Wijk. Simplified representation of vector fields. In *Proceedings of IEEE Visualization Conference*, pages 35–42, 1999.

[109] G. Turk and D. Banks. Image-guided streamline placement. In *Proceedings of ACM SIGGRAPH Conference*, pages 453–460, 1996.

[110] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[111] A. Valencia, H. Morales, R. Rivera, E. Bravo, and M. Galvez. Blood flow dynamics in patient-specific cerebral aneurysm models: the relationship between wall shear stress and aneurysm area index. *Medical Engineering & Physics*, 30(3):329–340, 2008.

[112] M. van der Zwan, A. Telea, and T. Isenberg. Continuous navigation of nested abstraction levels. In *Proceedings of Eurographics Conference on Visualization (Short Papers)*, pages 13–17, 2012.

[113] R. van Pelt, J. O. Bescós, M. Breeuwer, R. E. Clough, M. E. Gröller, B. ter Haar Romeny, and A. Vilanova. Exploration of 4D MRI blood flow using stylistic visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1339–1347, 2010.

[114] R. van Pelt, J. O. Bescós, M. Breeuwer, R. E. Clough, M. E. Gröller, B. ter Haar Romeny, and A. Vilanova. Interactive virtual probing of 4D MRI blood-flow. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2153–2162, 2011.

[115] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In *Proceedings of Vision, Modeling, and Visualization Conference*, pages 273–280, 2001.

[116] V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *Proceedings of IEEE Visualization conference*, pages 163–190, 2000.

[117] I. Viola, M. Feixas, M. Sbert., and M. E. Gröller. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):933–940, 2006.

[118] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[119] C. Wang. A survey of graph-based representations and techniques for scientific visualization. In *Eurographics Conference on Visualization - State-of-the-Art Reports*, pages 41–60, 2015.

[120] C. Wang and H.-W. Shen. Information theory in scientific visualization. *Entropy*, 13(1):254–273, 2010.

[121] L. Wang, Y. Zhao, K. Mueller, and A. E. Kaufman. The magic volume lens: An interactive focus+context technique for volume rendering. In *Proceedings of IEEE Visualization Conference*, pages 367–374, 2005.

[122] M. Wang, J. Tao, C. Wang, S. C.-K., and S. H. Kim. FlowVisual: Design and evaluation of a visualization tool for teaching 2D flow field concepts. In *Proceedings of American Society for Engineering Education Annual Conference*, Atlanta, GA, 2013.

[123] Y.-S. Wang, T.-Y. Lee, and C.-L. Tai. Focus+context visualization with distortion minimization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1731–1738, 2008.

[124] Y.-S. Wang, C. Wang, T.-Y. Lee, and K.-L. Ma. Feature-preserving volume data reduction and focus+context visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):171–181, 2011.

[125] J. Wei, C. Wang, H. Yu, and K.-L. Ma. A sketch-based interface for classifying and visualizing vector fields. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 129–136, 2010.

[126] T. Weinkauf, H. Theisel, A. V. Gelder, and A. T. Pang. Stable feature flow fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):770–780, 2011.

[127] A. P. Witkin. Scale-space filtering. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983.

[128] L. Wong, C. Dumont, and M. Abidi. Next best view system in a 3-D modeling task. In *Proceedings of International Symposium on Computational Intelligence in Robotics and Automation*, pages 306–311, 1999.

[129] C. Xu and J. L. Prince. Gradient vector flow: A new external force for snakes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 66–71, 1997.

[130] L. Xu, T.-Y. Lee, and H.-W. Shen. An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1216–1224, 2010.

[131] L. Xu and H.-W. Shen. Flow web: A graph based user interface for 3D flow field exploration. In *Proceedings of IS&T/SPIE Conference on Visualization and Data Analysis*, 2010.

[132] S. Y. Yan. *Cryptanalytic Attacks on RSA*. Springer-Verlag, 2008.

[133] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3D streamlines. In *Proceedings of IEEE Visualization Conference*, pages 471–478, 2005.

[134] H. Yu, C. Wang, and K.-L. Ma. Parallel hierarchical visualization of large time-varying 3D vector fields. In *Proceedings of ACM/IEEE Supercomputing Conference*, 2007.

[135] H. Yu, C. Wang, C.-K. Shene, and J. H. Chen. Hierarchical streamline bundles for visualizing 2d flow fields. In *IEEE VisWeek Posters*, 2010.

[136] H. Yu, C. Wang, C.-K. Shene, and J. H. Chen. Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1353–1367, 2012.

[137] S. Zhang, Ç. Demiralp, and D. H. Laidlaw. Visualizing diffusion tensor MR images using streamtubes and streamsurfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):454–462, 2003.

[138] S. Zhang, S. Correia, and D. H. Laidlaw. Identifying white-matter fiber bundles

in DTI data using an automated proximity-based fiber-clustering method. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1044–1053, 2008.

[139] X. Zhao, W. Zeng, X. Gu, A. Kaufman, W. Xu, and K. Mueller. Conformal magnifier: A focus+context technique with local shape preservation. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1928–1941, 2012.

[140] L. Zhu, S. Haker, and A. Tannenbaum. Flattening maps for the visualization of multibranched vessels. *IEEE Transactions on Medical Imaging*, 24(2):191–198, 2005.