EFFICIENT VISUAL ANALYSIS OF STEADY VECTOR FIELDS BASED ON STREAMLINE CHARACTERIZATION

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Shuyu Xu

May 2014

EFFICIENT VISUAL ANALYSIS OF STEADY VECTOR FIELDS BASED ON STREAMLINE CHARACTERIZATION

Shuyu Xu

APPROVED:

Guoning Chen, Chairman
Dept. of Computer Science

Gangbing Song
Dept. of Mechanical Engineering

Zhigang Deng
Dept. of Computer Science

Dean, College of Natural Sciences and Mathematics

EFFICIENT VISUAL ANALYSIS OF STEADY VECTOR FIELDS BASED ON STREAMLINE CHARACTERIZATION

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Shuyu Xu

May 2014

Abstract

Vector fields and its analysis play an important role in many scientific and engineering applications, ranging from automobile design, oceanography, climate study, to medicine. Due to the complex nature of flow fields and the data sizes, it is often difficult to obtain an efficient interpretation of the underlying dynamics of these flow data. Vector field topology that condenses the flow into its skeletal structure is a popular solution to address this challenge. It provides an intrinsic partition of the flow domain based on the different homogeneous flow behaviors at different regions. However, traditional vector field topology computation suffers from the issue of numerical instability, thus, the obtained results are typically not reliable. Recently, Morse decomposition has been introduced to address this numerical instability issue of the previous vector field topology extraction. The limitations of the original Morse decompositions are two-fold. First, the decomposition largely relies on the resolution of the underlying mesh, generating visualization with the serrated look. Second, its computation is expensive. In addition, it does not encode the flow uncertainty information, which may produce mis-leading results. To address these challenges, this project introduces an Image-Space Morse decomposition (ISMD) framework. This technique first converts the original flow defined on a triangle mesh into the pixel-based representation in an image plane using standard graphics hardware. The Morse decomposition is then computed under this image plane. This new framework not only mitigates the serrated boundary issue of the previous method with pixel-level accuracy but also enables a parallel implementation of the Morse decompositions

using OpenMPI and CUDA, respectively. In addition, it allows us to visualize uncertainty and error introduced during the computation by computing an ensemble of the ISMDs of the same flow with different perturbations and integration errors. The developed techniques have been implemented in a visual analysis tool that can handle both 2D and 3D steady vector fields.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Vector field visualization plays an important role in computer graphics and in scientific and engineering applications. For example, the analysis of computational fluid dynamics (CFD) simulations in the aerospace and automotive industries relies on effective visual representations of a variety of vector field data. Another example is the visualization of surface shapes by emphasizing feature curves following the principal eigen vector fields of their curvature tensor fields. There are two main streams to be reported. On one hand, there are flow visualization techniques, that display the local or descriptors of flow behaviors flow behaviour for the entire domain. On other hand, there are feature-based techniques that first extract the features of interest, such as topology and vortices, from the flow then emphasize them on top of the preceding visualization. Our approach is mainly a feature-based method. In particular, we

have developed a method that not only visualize the overall flow behaviour, but also emphasize the topological structure of the vector field.

Visualizing vector field topology is very important in understanding the critical behaviors of flow. For instance, the existence of recirculation zones (periodic orbits) can indicate stagnant flow which may be undesirable in engine design, because stagnant flows indicate trapped heat in the engine. Past work defines the topology of two-dimensional vector fields as fixed points and periodic orbits as well as the separatrices that connect them. This leads to a graph representation of the vector field which is referred to as Entity Connection Graph, or ECG. However, However, the computation of ECG largely relies on numerical integration. Therefore, the resulting ECGs are sensitive to the integration errors and other uncertainty contained in the flow data, making the analysis results unreliable. Chen at al present a rather different approach to the representation, extraction and visualization of flow topology. The representation of the global dynamics is done in terms of an acyclic directed graph called the Morse connection graph (MCG). The nodes in this graph, which we refer to as Morse sets, correspond to polygonal regions in the phase space, which we define to be Morse neighbourhoods. All the recurrent dynamics is contained in the Morse neighbourhoods. The edges in an MCG indicate how the flow moves from one Morse neighbourhood to another. In contrast to trajectory-based topological analysis, such as vector field skeleton and ECG, an MCG is stable with respect to perturbations, i.e. given sufficient information on errors of the vector field it is possible to make rigorous interpretations about the underlying dynamics. In other words, a well defined error, $\epsilon > 0$, can be bounded and included into the map of the flow domain.

To perform Morse decomposition, i.e., compute MCGs, Chen et al. [3] first construct another directed graph by considering the behaviours of the vector field along edges of the triangles, which we refer to as the geometry-based method. This conversion is referred to as the *flow combinatorialization* that encodes the original flow into a directed graph, F. The nodes of F are the triangles of the original data, and the edges of F indicate the flow mapping relations between triangles. Next, the Morse sets are identified as the strongly connected components of F. Since each node in F corresponds to a triangle in the original data, the extracted strongly connected components are the triangular Morse neighborhoods of the flow. Because the triangulation is not adapted to the vector field, this can result in coarse Morse sets. To counter this, a $\tau-$map approach was introduced later, which computes the flow combinatorialization by integrating a finite, that is obtained by integrating a finite set of points for a finite amount of time. Theoretically this method can produce as detailed an MCG as is desired and in practice it produces a finer MCG is than the geometry-based method. The key challenges with the $\tau-$map approachs are choosing an appropriate temporal discretization of the flow and constructing a high-quality flow combinatorialization.

There are a number of limitations of the original Morse decomposition computation. First, the resolution of MCG is constrained by the resolution of the underlying mesh. That is, if a coarse triangle mesh is used to, the boundaries of the identified Morse sets may not be smooth (e.g., zig-zag like or serrated). This visual representation is against the smooth nature of the flow. See Figure 1.1 for an example.

In addition, small-scale features could be missed using coarse triangle mesh representation. See Figure 1.2 for an illustration. Second, the computation of Morse decomposition requires the tracing of a large number of particles, resulting in a generally slow performance for large-scale data. Third, the existing definition of Morse decomposition does not address the representation of uncertainty in the result introduced during computation. In order to address these challenges, we present a different approach called Image Based Mosrse Decomposition method. Compare with the original method that use the input data directly, the image-based Mosrse Decomposition has an extra pre-compute step. Before analyzing the flow behaviour, we transfer the flow from its triangle-based representation to a pixel-based representation with the aid of standard graph hardware. In other words, we project the original flow that is defined on a triangle mesh into an image plane so that the R, G, B color channels of a pixel of an image encode a 2D/3D vector (the last component is set to 0 in 2D case). More details are provided in Section 5. We also study the uncertainty of the stability of the resulting MCG by computing an ensemble of Morse decompositions of the same flow under different parameter settings. To simulate the uncertainty and noise that may be contained in the flow data, we artificially introduce small perturbation to the original flow. That is, for each run of the ISMD, we introduce different but bounded perturbation to the flow, and then extract its Morse sets. Once we get the results of all these different runs, overlap them to obtain the statistical information of each pixel, e.g., given a pixel how many times it falls in the same Morse sets in these different runs. The advantage of this method is after we apply the uncertainty and stability study on it, the result tends to emphasize the

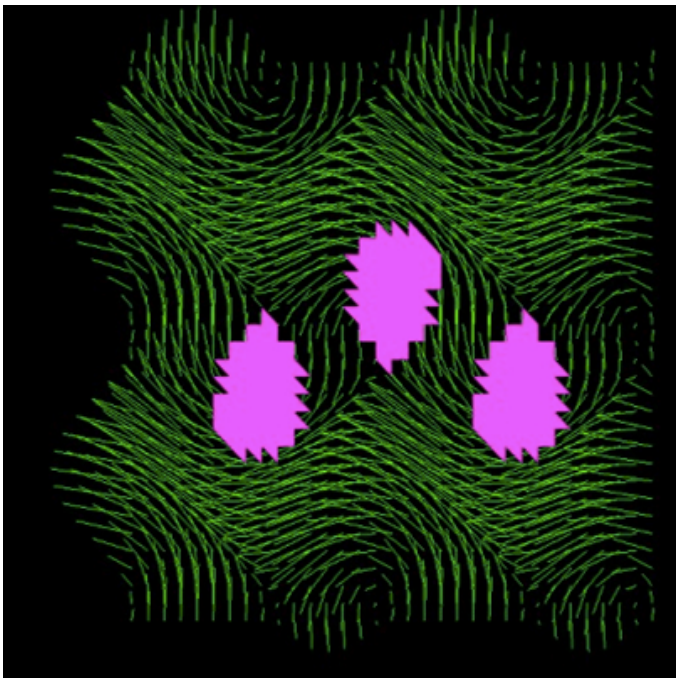

Figure 1.1: Original Morse decomposition result of data1, the purple area is the Morse set.

Morse sets with pixel-level accuracy, and it also provides the reliability information of the obtained Morse sets. We design a visualization method that highlights the more reliable regions in the obtained Morse sets. The accuracy of original Morse Decomposition method is not guaranteed and the result is unable to visualize the reliability. However, the new approach still suffers from the large computational cost. It's a heavy task to analysis the entire vector field pixel by pixel. Therefore, we employ a number of CPU/GPU parallel computation techniques in our algorithm to speed up the computation. In section 5, we provide all the mathematics details and implementation.

Rather than vector filed topology visualization, we also developed a method to

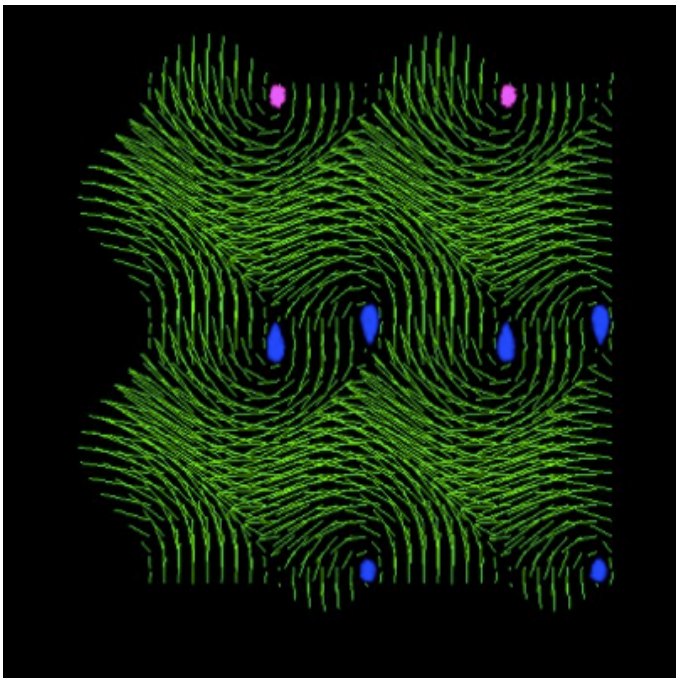

Figure 1.2: Image-Space Method result of data1 combined with arrow plot, the colored area is the Morse set.

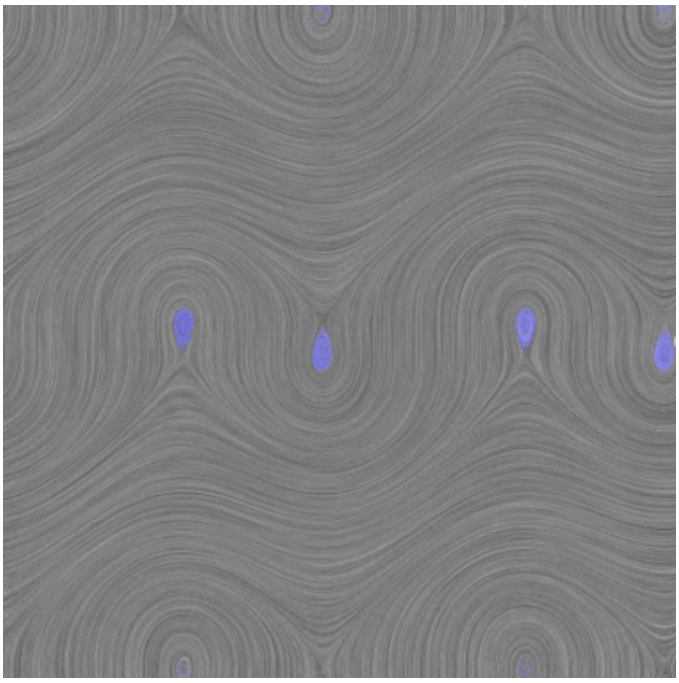

Figure 1.3: Image-Space result combined with LIC, background is the LIC structure and the colored area is the Morse set.

visualize the entire flow behaviour algorithm based on the observation that streamlines cluster together in attracting regions like sinks or attracting separatrices, while they thin out in repelling regions like sources or repelling separatrices. Many approaches exist to overcome this problem. We, instead, would like to make use of this phenomenon by counting the number of streamlines that cross each cell of the underlying grid structure. This accumulation step leads to a scalar density field. The density is high for attracting regions and low for repelling regions when we use forward integration, and vice versa when we use backward integration. Rendering (and possibly blending) these two density fields leads to a visualization that highlights the structural information of the underlying flow field. Using an appropriate seeding mechanism for the streamlines, the visualization also exhibits the overall flow behaviour leading to line integral convolution(LIC)-type visualizations in regions with no structural information. The first step of our algorithm is the flow integration step. We use a geometric approach and higher-order integration methods to precisely trace the path, i.e., streamline of a (mass-less) particle moving under the influence of the underlying flow field. Instead of storing the geometry of the streamline, we increase the counter of each cell the streamline passes through. Starting with all counters initially being zero and running an animation with many streamlines moving over time leads to an accumulated intensity value for each cell. We store the result in a texture representing the density field mentioned above, the details are provided in section 6. The second step is the evaluation of the density field using texture-based rendering methods. Using texture-based methods for rendering supports a dense visualization, which is important, if we want to capture the entire vector field. We use transfer

functions to extract attracting and repelling regions and explore their interplay by blending the two density fields together for forward and backward integration.

Also, we extend our 2D analysis framework to the 3D vector field visualization. The algorithm is similar with the 2D case. The first step is flow integration as well, trace out the streamline by geometric approach and hinger-order integration method. The different is, in 2D case we increase the counter of each cell the streamline passes through. In 3D case we increase the counter of the small block or a voxel. To visualize the 3D density scalar field, we utilize the ray-casting volume rendering method. The technique of volume ray casting can be derived directly from the rendering equation. It provides results of very high quality rendering, as the computation emanates from the output image and not the input volume data, as is the case with object-based techniques. In order to indicate the advantage of this method and let the user has different view point to watch the 3D vector field. We build a 3D vector multi-visualization system including stream lines, 2D cutting slices and volume rendering. It allows the user to select the view mode by the user friendly interface.

1.2 My Contribution

- Image-Space Morse-Decomposition : we design a new Morse-Decomposition algorithm to visualize the vector filed topology in a more accurate and smooth fashion.

- Study of probability and uncertainty of Image-Space Morse-Decomposition:

we are the first to apply the stability and uncertainty to analysis the Morse-Decomposition result,in order to analyze the reliability of the obtained Morse decompositions.

- High-Performance Computation: We apply HPC techniques (OpenMp) on Morse-Decomposition algorithm to improve the performance, We also redesign the algorithm to make it more suitable for the HPC API.

- 3D Vector filed visualization system: We developed a 3D Vector field visualization system that integrates 4 different visualization methods.

- We utilize a number of visualization techniques on different analysis results. We present why choosing a proper visualization is crucial to reveal useful information, and we also demonstrate how to utilize transfer functions to decrease perturbations of data sets. We also compare different flow data.

1.3 The Structure of the Thesis

The rest of the thesis is organized as follows: The overview of the whole system is provided in Chapter 1. Chapter 2 introduces the recent popular data visualization methods. Chapter 3 explains the definition of vector field and vector field topology. Image-Space Mores Decomposition methods and stability analysis of Image-Space Morse Decomposition are detailed in Chapter 4 and 5. Chapter 6 describes our 3D vector field visualization system. Chapter 7 provides the results, discussions and conclusions.

Chapter 2

Related Work

2.1 Vector Field Visualization

Vector field visualization approaches can be categorized into rendering approaches such as direct, geometric, and texture-based flow visualization, and analytical topology extracting approaches, also called feature-based approaches.

Early attempts such as arrow and hedgehog plots or color coding fall into the category of direct flow visualization . They provide an intuitive image of local flow properties. For a better understanding of global flow dynamics with respect to long-term behavior, integrationbased approaches have been introduced. These integrate flow data leading to trajectories of no-mass particles moving over time. Geometric flow visualization approaches render the integrated flow using geometric objects such as lines, tubes, ribbons, or surfaces.

In particular, streamlines are widely used and have been integrated into various flow visualization systems [1]. Streamlines naturally lead to a sparse representation of the vector field, such that seeding strategies become a critical issue. Dense representations are desirable, as they provide information concerning overall flow behavior and serve as a context for chosen visualization methods. Park et al. [10] presented a dense geometric flow visualization approach by using a high number of randomly seeded streamlines with short life times, generated via particle advection in texture space.

In texture-based flow visualization, a texture is used for a dense representation of a flow field. The texture is filtered according to the local flow vectors leading to a notion of overall flow direction [8]. The most prominent approaches are line integral convolution (LIC) [2] and texture advection [9].

The LIC primitive is a noise texture, which is convolved in the direction of the flow using filter kernels. In texture advection, the primitive is a moving texel, while the motion is directed by the flow field [6]. One major drawback of texture-based methods for volume data is occlusion, which can be alleviated by the application of multi-dimensional transfer functions (MDTF) [7].

Feature-based flow visualization is concerned with the extraction of specific patterns of interest, or features. Various features such as vortices, shock waves, or separatrices have been considered. Each of them has specific physical properties, which can be used to extract the desired feature. Once a feature has been extracted, standard visualization techniques are used for rendering [12].

2.2 Vector Field Analysis

Helman and Hesselink [5] introduce vector field topology for the visualization of vector fields. They also propose efficient algorithms to extract vector field topology. Following their footsteps, much research has been conducted in topological analysis of vector fields. For example, Scheuermann et al. [13] use clifford algebra to study the non-linear fixed points of a vector field and propose an efficient algorithm to merge nearby first-order fixed points. Tricoche et al. [15] and Polthier and Preub give efficient methods to locate fixed points in a vector field. Wischgoll and Scheuermann [11] develop a method to extract closed streamlines in a 2D vector field defined on a triangle mesh. Note that closed streamlines are in fact attracting and repelling periodic orbits. Theisel et al. [14] propose a mesh-independent periodic orbit detection method for planar domains. In contrast to these approaches, our automatic detection algorithm is extended to surfaces. Furthermore, this is the first time periodic orbit extraction and visualization has found utility in a real application.

Chapter 3

Background

In this chapter, we present the concepts of the vector fields, vector field topology and a compact summary of the theories of dynamical systems upon which our work is built. Our discussion will focus on time-independent flow only.

3.1 Vector Fields and Vector Field Topology

A vector field is an assignment of a vector to each point in a subset of Euclidean space. A vector field in the plane, for instance, can be visualized as a collection of arrows with a given magnitude and direction each attached to a point in the plane. Vector fields are often used to model, for example, the speed and direction of a moving fluid throughout space, or the strength and direction of some force, such as the magnetic or gravitational force, as it changes from point to point. The elements of differential and integral calculus extend to vector fields in a natural way. When a vector field

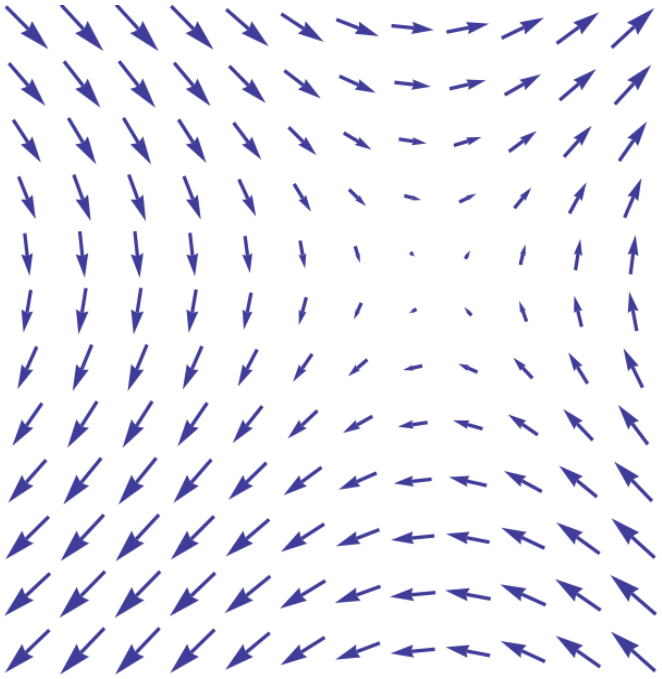

Figure 3.1: A vector field example

represents force, the line integral of a vector field represents the work done by a force moving along a path, and under this interpretation conservation of energy is exhibited as a special case of the fundamental theorem of calculus. Vector fields can usefully be thought of as representing the velocity of a moving flow in space, and this physical intuition leads to notions such as the divergence (which represents the rate of change of volume of a flow) and curl (which represents the rotation of a flow). In coordinates, a vector field on a domain in n-dimensional Euclidean space can be represented as a vector-valued function that associates an n-tuple of real numbers to each point of the domain. This representation of a vector field depends on the coordinate system, and there is a well-defined transformation law in passing from one coordinate system to the other. Vector fields are often discussed on open subsets of

Euclidean space, but also make sense on other subsets such as surfaces, where they associate an arrow tangent to the surface at each point (a tangent vector). More generally, vector fields are defined on differentiable manifolds, which are spaces that look like Euclidean space on small scales, but may have more complicated structure on larger scales. In this setting, a vector field gives a tangent vector at each point of the manifold (that is, a section of the tangent bundle to the manifold). Vector fields are one kind of tensor field. It typically can be expressed as an ordinary differential equation (ODE)

$$\frac{d\varphi(x)}{dt} = v(x)$$

Streamlines are a family of curves that are instantaneously tangent to the velocity vector of the flow. These show the direction a fluid element will travel in at any point in time. By definition, different streamlines at the same instant in a flow do not intersect, because a fluid particle cannot have two different velocities at the same point. It provides a snapshot of some flow field characteristics. It's the derivative information of flow data. Mathematically, the streamline can be integrated over time,

$$S(t) = s_0 + \int_0^t V(s(u))du$$

We also call it trajectory, solution, integral curve. All my visualization methods are based on the streamline tracing. The numerical integration method is used for streamline tracing. The simplest integration method for tracing streamline is the Euler's method. Since streamline is defined as $S(t) = s_0 + \int_0^t V(s(u))du$, the Euler's

method can be expressed as the following iterative form:

$$S_{i+1} = S_i + dt \times V(S_i)$$

where $V(S_i)$ is the vector value defined at the current integration point S_i, and dt is a small positive real number that is used to control the integral stepsize. Since the Euler's integrator has first order accuracy based on the Taylor expansion of $S(t)$. One needs to choose an infinitely small dt value in order to minimize the integration error, resulting in large number of integration steps and computational cost. To mitigate that, we employ the popular fourth-order Runge-Kutta integration. The mathematical definition of nth order Runge-Kutta method given below.

$$w_0 = \alpha$$

Given w_i, for $i = 0, 1, \ldots, N-1$, set

$$\begin{aligned}
k_1 &= hf\left(t_i, w_i\right) \\
k_2 &= hf\left(t_i + \frac{h}{2}, w_i + \frac{k_1}{2}\right) \\
k_3 &= hf\left(t_i + \frac{h}{2}, w_i + \frac{k_2}{2}\right) \\
k_4 &= hf\left(t_i + h, w_i + k_3\right) \\
w_{i+1} &= w_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right).
\end{aligned}$$

Vector field topology provides qualitative (structural) information of the underlying dynamics. It usually consists of certain critical features and their connectivity, which can be expressed as a graph. eg. fix point, periodic orbits and invariant sets.

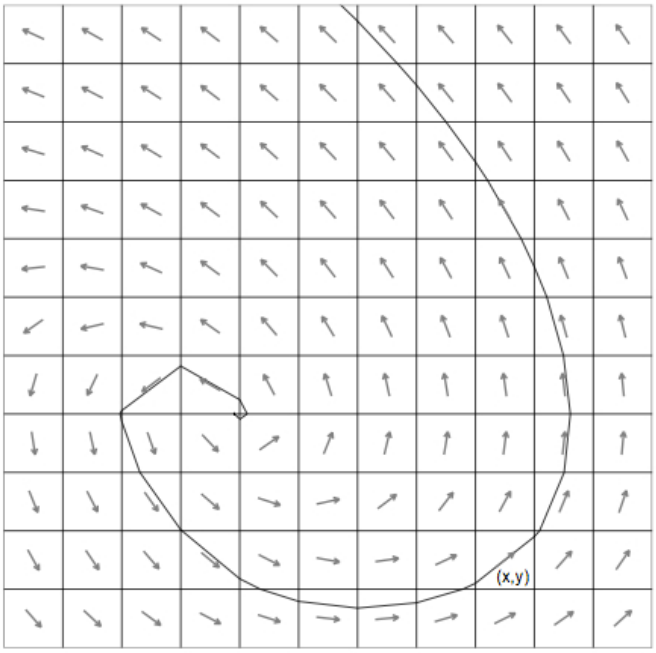

Figure 3.2: A 2-Dimensional vector field showing the stream line starting from cell(x,y)

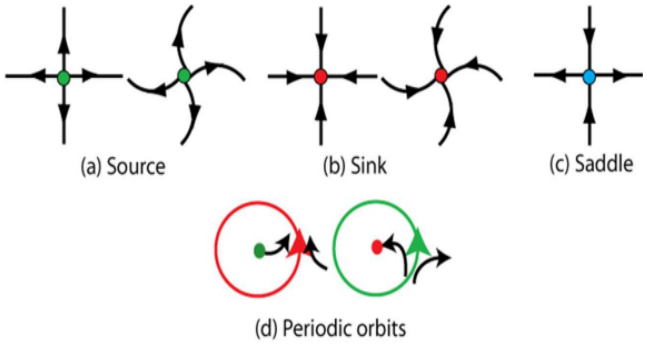

Figure 3.3: Examples of fixed points and periodic orbits.

- Fix Point: A fix point x is $x \in X$ when $\varphi(t, x) = x$ for all $t \in R$. For example, there are 3 different types of fixed points such as sink, source and saddle.
- periodic orbits: x is a periodic point if there exist a $T > 0$ such that $\varphi(T, x) = x$. The trajectory of a periodic point is called a periodic orbit.
- Invariant sets: An invariant set $S \subset X$ satisfies $\varphi(R, s)$. A trajectory is an invariant set, fixed points and periodic orbits are invariant sets.

3.2 Morse Decomposition

We are interested in describing the topological structures of the flow generated by a vector field $x = f(x)$ defined on a triangulated surface $X \subset M$. However, the information we are given consists of a finite set of vectors

$$\{f_d(v_i) | v_i \ a \ \ vertex \ of \ \ X\} \tag{3.1}$$

obtained either by a numerical simulation or from experiment. This means that at best we can assume that we have a uniform bound on the errors of the observed vector field versus the true vector field, that is for each v_i,

$$\|f(v_i) - f_d(v_i)\| \leq \varepsilon \tag{3.2}$$

In addition, since we are only given the data (Eq. 3.1) we extend f_d to a vector field on X by some means of interpolation (typically linear interpolation). Assuming that f is well approximated by f_d it is reasonable to assume that the bounds of (Eq. 3.2) are global, that is

$$\|f(v_i) - f_d(v_i)\| \leq \varepsilon$$

for all $x \in X$

The easiest way to encode the aforementioned information is to consider a family of vector fields F defined on the surface X and parametrized by some abstract parameter space Λ. We assume that for each $\lambda \in \Lambda$ the vector field

$$\dot{x} = f(x, \lambda)$$

gives rise to a flow

$$\varphi_\lambda : R \times X \to X$$

In this setting we assume that there exist parameter values λ_0,$\lambda_1 \in \Lambda$ such that $f(x) = F(x, \lambda_0)$ and $f_d(x) = F(x, \lambda)$. Bifurcation theory tells us that even if $\lambda_0 \approx \lambda_1$. the orbits, i.e. fixed points, periodic orbits, separatrices, of $\varphi(\lambda_0)$ and $\varphi(\lambda_1)$ need not agree. The implication is that computing such orbits for the vector field f_d does not imply that these orbits exist for the true vector field f . This leads us to weaken the

topological structures which we use to classify the dynamics. A Morse decomposition of X for a flow φ_λ is a finite collection of disjoint compact invariant sets, called Morse sets [14].

$$M(X, \varphi) := \{M_\lambda(p) | p \in (P_\lambda, \succ_\lambda)\}$$

where $\succ_\lambda$ is a strict partial order on the indexing set P_λ, such that for every $x \in X \setminus \cup$ $p \in P_\lambda$, there exist indices $p \succ_\lambda q$ such that

$$\omega(x) \subset M_\lambda(q)$$

and

$$\alpha(x) \subset M_\lambda(p)$$

It is easy to verify that any structures associated with recurrent dynamics of φ_λ , i.e. fixed points, periodic orbits, chaotic dynamics, must lie in the Morse sets. The dynamics outside the Morse sets is gradient-like. Morse decompositions of invariant sets always exist, though they may be trivial, i.e. consisting of a single Morse set M.

Chapter 4

An Image-Space Morse Decomposition

In this chapter, we review the previous Morse Decomposition computation pipeline and present our Image-Space Morse Decomposition method, Then, we compare the results generated by the previous Morse decomposition computation and our ISMD method.

4.1 MCG Construction

We now summarize the pipeline of constructing an MCG given the vector field V defined on a triangulated surface X. First, we perform flow combinatorialization. That is, we encode the flow dynamics into a directed graph, denoted by F, whose nodes represent the elements (e.g. triangles) of the underlying mesh and edges indicate the

flow dynamics.

Second, we find the strongly connected components of the directed graph F, which gives rise to the Morse neighbourhoods that are the polygonal regions constrained by the given mesh in the phase space. They contain the Morse sets $M(X;V)$ of the flow and have a non-trivial Conley index.

Third, we compute a quotient graph $\mathcal{F}$ from F by treating each strongly connected component of F as a node. The nodes in this quotient graph $\mathcal{F}$ include Morse sets (non-trivial Conley index) and the intermediate nodes corresponding to the polygonal regions with gradient-like flow behaviours (i.e, trivial Conley index).

Finally, we extract the MCG from $\mathcal{F}$ by removing intermediate nodes from $\mathcal{F}$ as illustrated in Figure 4.2. To visualize the MCG, we classify the nodes of the MCG into three types: Source Morse sets, Ri, are nodes absent of incoming edges in the MCG; Sink Morse sets, Ai, are nodes without outgoing edges in the MCG; Saddle Morse sets, Si, are neither source Morse sets nor sink Morse sets. According to the partial order determined by the edges in the MCG, we lay out the nodes such that the source Morse sets appear at the top of the graph, the sink Morse sets are placed at the bottom of the graph and the saddle Morse sets are placed between the source and sink Morse sets. MCG has a multi-layer structure, which provides more information than the ECG. Furthermore, unlike ECGs, saddle-saddle connection is a generic case in MCG.

The limitation of this original method is the non-smooth boundaries of the obtained Morse sets in the visualization and the lack of the ability for the stability

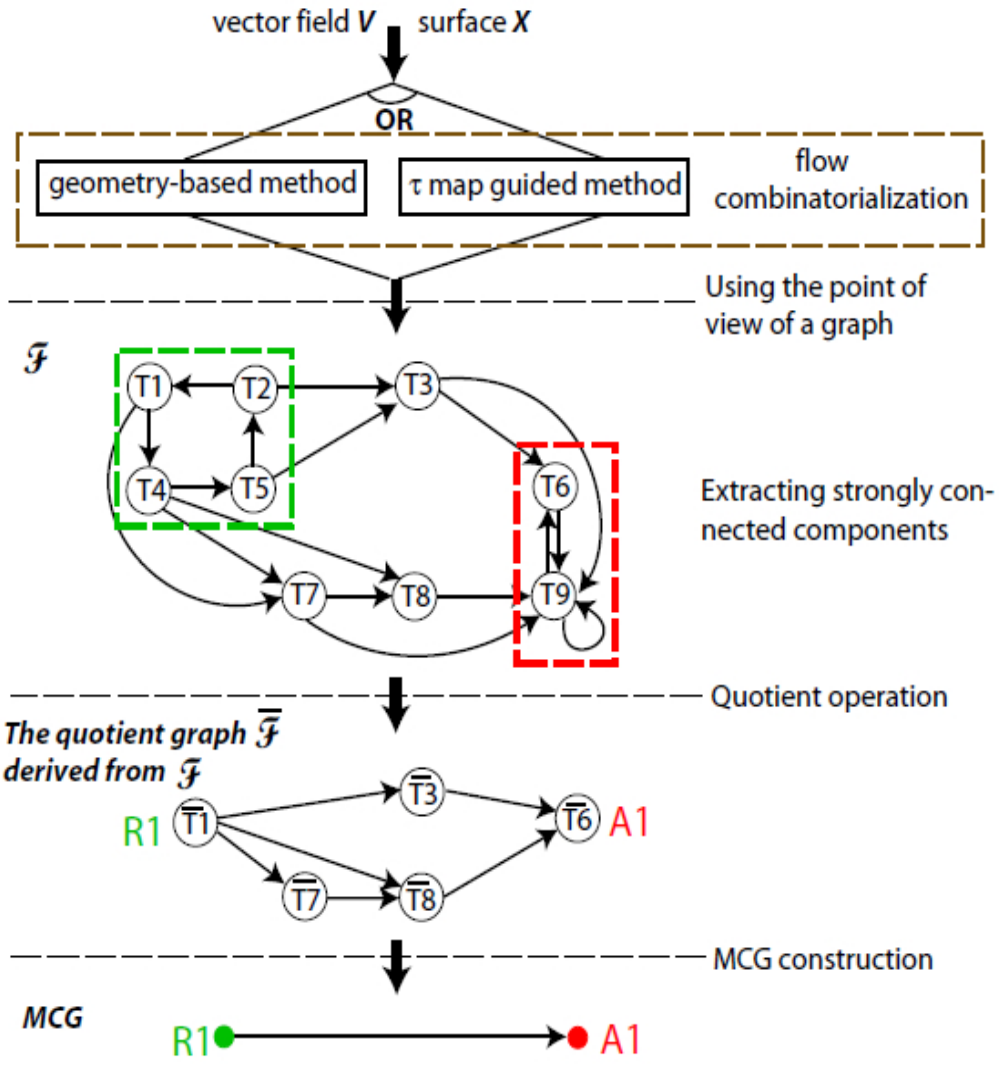

Figure 4.1: This figure illustrates the pipeline of MCG construction.

analysis. This method would some the important flow behaviours at the boundary and the rendering result got the wired shape. The details and the improvement will be discussed in next section.

4.2 Implementation of Image-Space method

As we present before, the processing unit of the original Morse-Decomposition method is a single triangle mesh. When we analyze the flow behaviour, we consider each triangle as one node in the graph. Once we finish the data analysis, the visualization result is generated bases on the triangle as well. There are two main issues of this method. First, because the rendering unit of the Mores Sets is a single triangle, it makes the result look serrate. There are some sharp corners on the boundaries of some Morse sets (see Figure 1.1 for an example). This does not reflect the true Morse sets that are smooth in the nature. The shape of all Morse sets would be smooth curve like Figure 1.2. It's easy to explain that, the reason why we got this issue is the size of triangle is too large.

The second issue is the possible missing of small-scale features. Since the processing unit is a triangle, we assume all the vector field within that unit have the same flow behaviour. But if the input triangle has a large size, it may contains sub-triangle flow behaviour. Which example or figure are you talking about right now? The centre area of triangle is the special area which we would like to mark as a Morse-Set. And the rest area is just some feature less area, we don't want to highlight it. However, the whole triangle is considered as one topology node in the

original algorithm. The original method will mark all of them. So the normal area in this triangle is kind of noise or error in the result. In some cases, the size of the input triangle is large enough so that it can even contain some small period obits within itself. Therefore, we would loss some information, because it's impossible to catch the flow behaviour in this case. Compare with the result Figure 1.2 and Figure 1.1, the Morse sets of the two result located in the same position, all of them catch the fixed points feature in the middle of the vector filed. However, the size of the Morse sets are totally different. The Morse set in result Figure 1.2 are much smaller, which means the Figure 1.2 is more accurate.

In order to address these two issues. We present the Image-Space Morse-Decomposition to improve the accuracy and quality. This method can be separated into three stages: preprocessing, analysis and visualization. The pipeline is highlighted in Figure Figure 4.2. In the preprocessing stage which is the key step of our method. We generate a pixel based input data to encode the original triangle mesh input data, which is the focus of our work to reduce the size of the input unit. We then compute the flow combinatorialization based on the flow in the obtained image plane. It follows the the image-space LIC pipeline. The analysis and the visualization stages are similar as the original Morse-Decomposition method. Again, the difference is we consider a pixel as an input unit instead of a triangle.

In particular, the preprocessing stage is converting the original vector fields defined on triangle mesh into image-space representation with the help of the standard

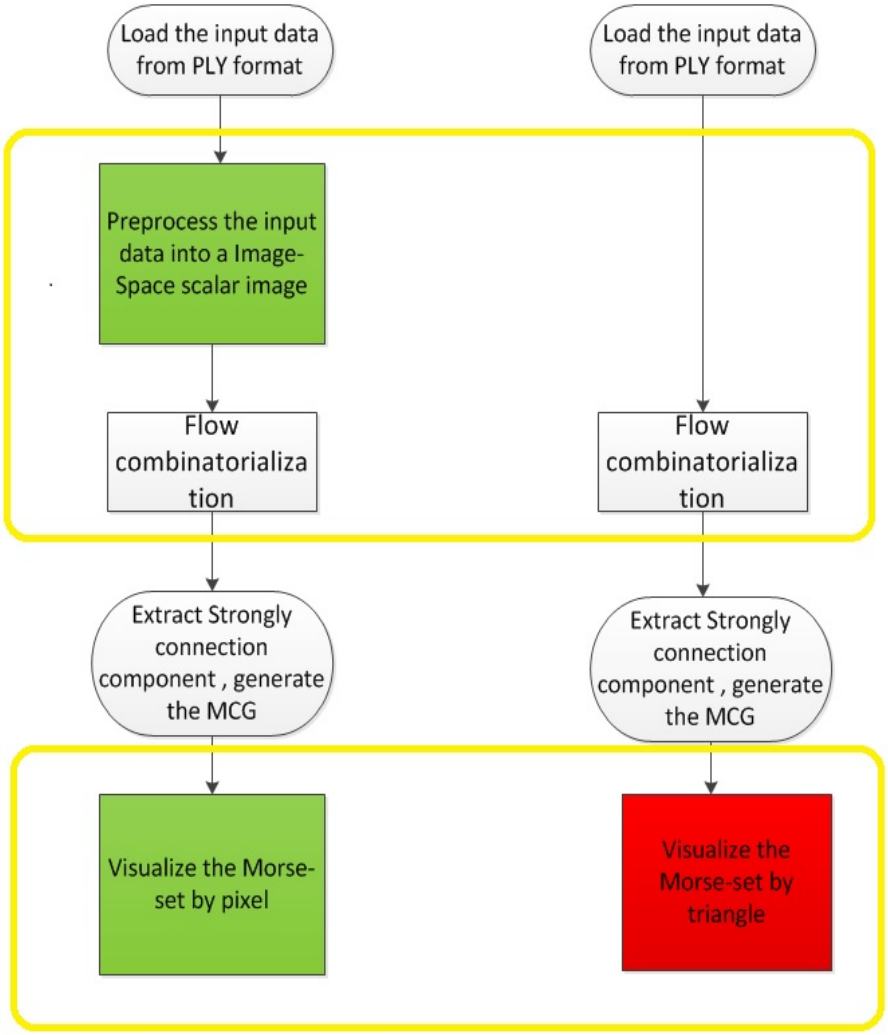

Figure 4.2: This figure illustrates the pipelines of Original method and Image-Space method.

graphics hardware, then decrypt the vector fields data back form the scheme.

$$v = \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$$

The the rendering colour scheme of the modern computer hardware is RGB colour scheme. In RGB model, every pure colour can be expressed as a mixture of the additive primary colour: red, green, and blue. For example, cyan, magenta, and yellow, the secondary colour of RGB, are formed by the mixture of two of the additive primary colour and the exclusion of the third. Yellow comes from the combination of red and green, cyan from green and blue, magenta from blue and red. If we add red, green, and blue together in full intensity, we get white. Naturally, when all colour of the electromagnetic spectrum converge in full intensity, white colour is created. Both of the 3D vector and the RGB model have 3 components. Our approach use a transfer function to encode the 3D vector information into RGB value. However, the size of the input triangle is typically not as the same as the pixel size. Therefore, not every pixel can get the corresponding vector value directly from the original vector field that is defined at the vertices (i.e., data sample points) of the triangle mesh. That said, the vector values at most pixels need to be interpolated from the input sample points.

Fortunately, this can be automatically achieved with the aid of the model graphics hardware. A straightforward solution would be using the rendering functionality that is provided in standard graphics libraries like OpenGL. To better understand how this works, let us first briefly describe the process of how OpenGL draws a

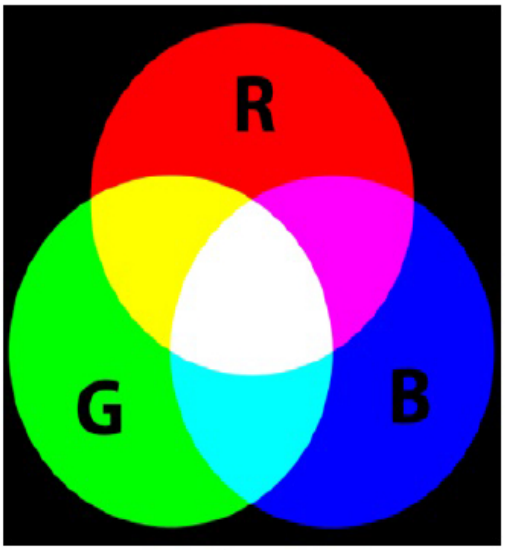

Figure 4.3: RGB model

triangle. Figure 4.4 shows the details of this process. First of all,the coordinates of the three vertices that form this triangle need to be provided to OpenGL as well as their individual colors that are represented using RGB model. Then, OpenGL determines the pixels that fall in this triangle and light them up with their respective colors computed from the three colors assigned to the three vertices using linear interpolation. If we zoom into the picture, we can see the triangle drawn by OpenGL is not exactly a real triangle, but rather a collection of a set of squares which are pixels. Since a pixel is sufficiently small so that human eye, so the rendered shape looks smooth in the regular view. We make use of this advantage to improve the Morse Sets visualization quality.

Since we know the original input data is a vector field defined on a triangle mesh, and it's possible to transfer the vector information into a color plot. Thus, Thus,

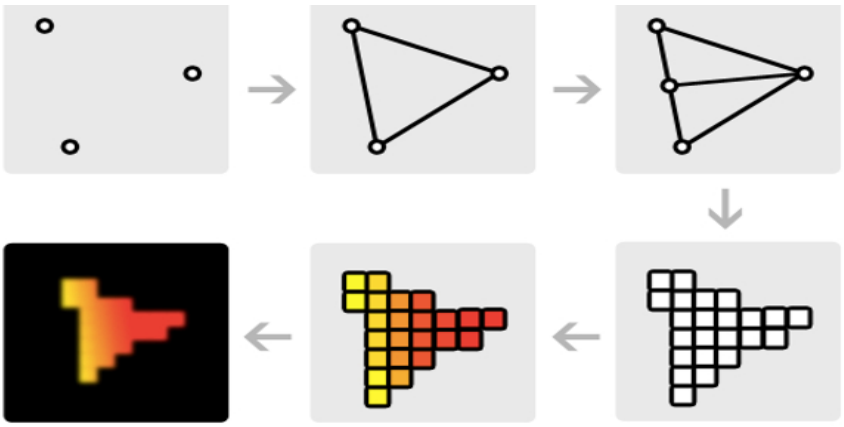

Figure 4.4: This figure illustrates the pipeline of drawing a triangle by OpenGL.

with this conversion, the original vector field is not encoded in a vector field defined in an image plane (i.e., the color plot) with much higher resolution. The following formula (i.e., transfer function) describes how we encode the vector value into RGB color channels:

$$R = \frac{V_x - V_{xmin}}{V_{xmax} - V_{xmin}}$$
$$G = \frac{V_y - V_{ymin}}{V_{ymax} - V_{ymin}}$$
$$B = \frac{V_z - V_{ymin}}{V_{zmax} - V_{zmin}}$$

To decode the vector value from the color of a given pixel for the later streamline integration, we use the following formula:

$$V_x = V_{xmin} + R * (V_{xmax} - V_{xmin})$$

$$V_y = V_{ymin} + R * (V_{ymax} - V_{ymin})$$

$$V_z = V_{zmin} + R * (V_{zmax} - V_{zmin})$$

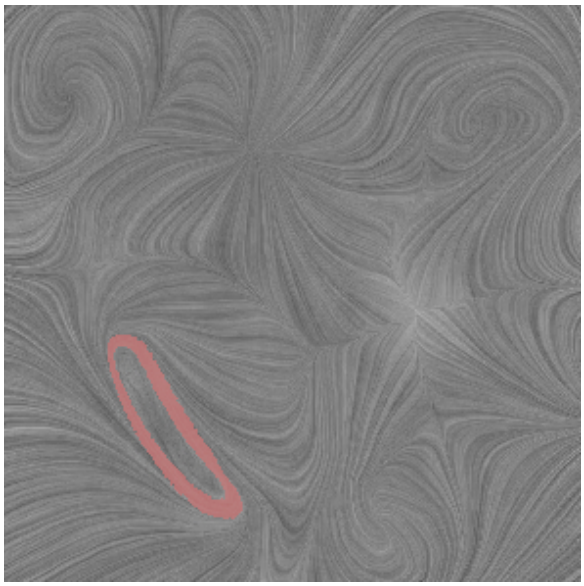

Figure 4.5: Result of Image-Space Morse Decomposition method, pink area is the Morse set.

Then, algorithms as used by the original Morse decomposition pipeline. Now, each Morse set is a collection of pixels, which we render them one-by-one.It is obvious that the resulting visualization of Morse sets has better smoothness compared to the triangle-based visualization due to the increasing resolution. Figure 1.1, Figure 1.2 provide a comparison.

4.3 High-Performance Computation

In this section, we will discuss the performance of the algorithm and explain how to improve it. Either the triangle-based or pixel-based Morse decomposition pipeline requires to trace a large number of streamlines. It's a very time consuming step,

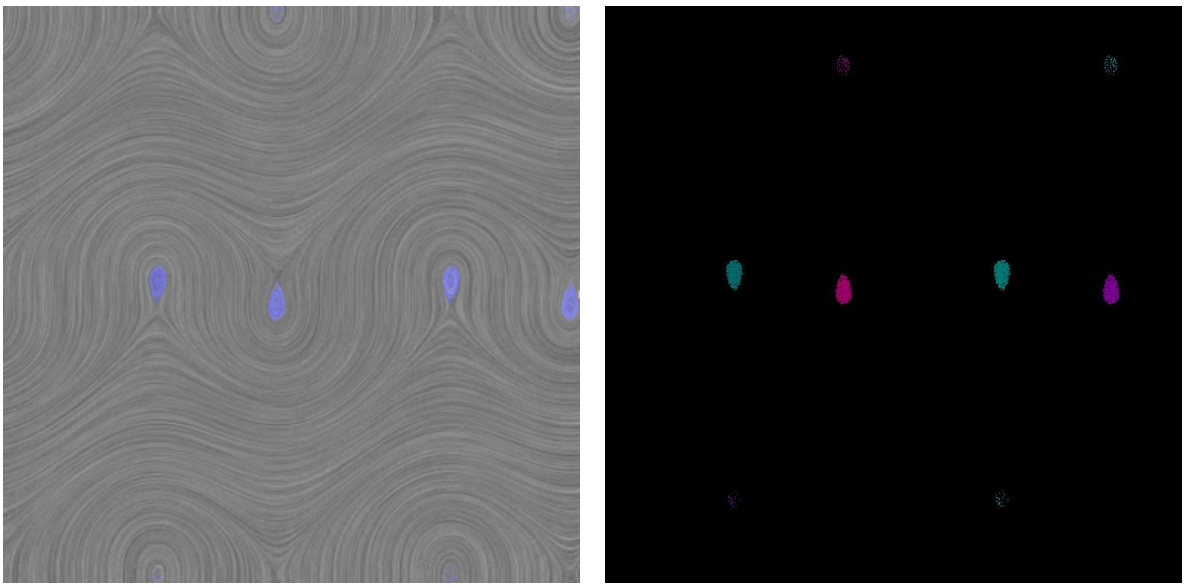

Figure 4.6: Result of Image-Space Morse Decomposition method, the colored area is Morse set. The left figure has LIC background.

because each streamline tracing procedure requires a series of integration computations. Since the image plane has much higher resolution than the triangle mesh (e.g., more samples), we have to handle much larger number of streamlines in the image-space method. The execution time will be much longer. This is a kind of trade off, i.e., higher resolution leads to more accurate result, but also larger time complexity. In order to obtain the better result in a shorter time, we employ techniques from High Performance Computation (HPC), such as OpenMP and CUDA, in our implementation. OpenMP is an API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran on most processor architectures and operating systems. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behaviour. The

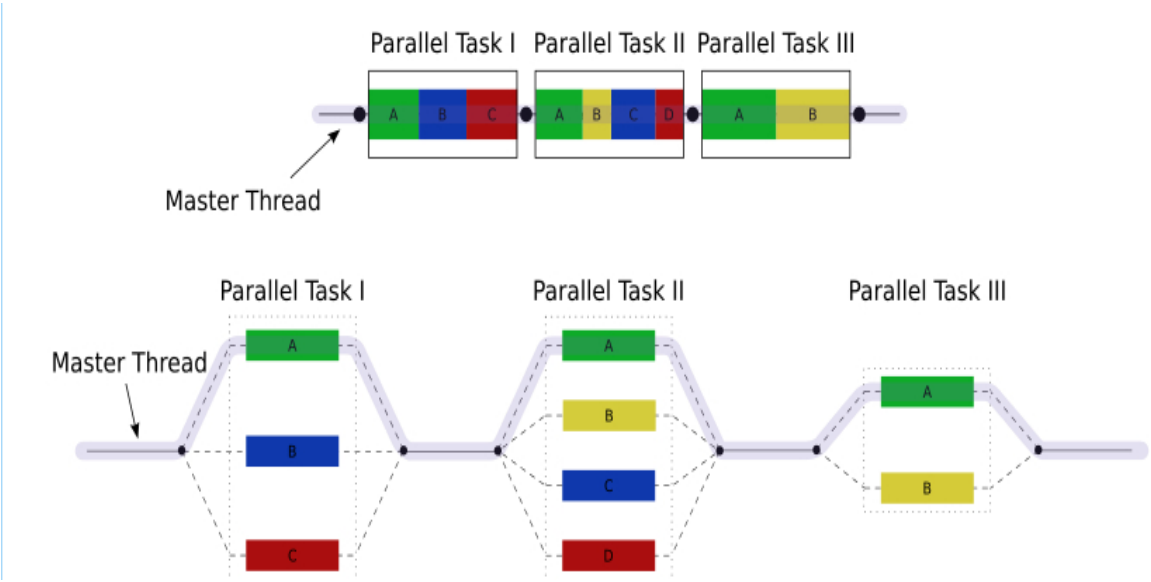

Figure 4.7: OpenMP Pipeline

pipeline of OpenMP is showing in Figure 4.7.

All OpenMP programs begin as a single process: the master thread. The master thread executes sequentially until the first parallel region construct is encountered. The master thread then creates a team of parallel threads and assigns the tasks into other threads. When the team threads complete the statements in the parallel region construct, they synchronize and terminate, leaving only the master thread. This technique executes the tasks in parallel, reducing the execution time. However, only independent data can be parallelized using OpenMP. The seeding method we used for the triangle-based Morse-Decomposition method is adaptive, which is a recursive function call. It's impassible to apply parallel computing on it, because the input of the current step depends on the out put of the previous step. According to the feature of the recursive function, there is an algorithm called Breadth first

```
procedure BFS(G,v) is
    create a queue Q
    create a set V
    enqueue v onto Q
    add v to V
    while Q is not empty loop
        t ← Q.dequeue()
        if t is what we are looking for then
            return t
        end if
        for all edges e in G.adjacentEdges(t) loop
            u ← G.adjacentVertex(t,e)
            if u is not in V then
                add u to V
                enqueue u onto Q
            end if
        end loop
    end loop
    return none
end BFS
```

Figure 4.8: BFS

search (BFS) can change the recursive function into a iterative way. The idea is using a data structure queue to order the all the tasks, then run these tasks in a "recursive" order. Figure 4.8 shows the details of this conversion. After conversing the recursive function, , we can assign all the tasks to different processors. The performance enhancement depends on the hardware, we did some experiments on the local desktop with Intel Xeon CPU (8 cores) and 16GB RAM. Table 4.3 illustrates the details.

$\tau-$ **value/time**	**Original**	**Speedup**	**Improvement**
0.5	14.904s	4.510s	69.78%
1.0	34.639s	8.769s	74.68%
2.0	54.185s	17.210s	68.23%

Table 4.1: The data set we use in this experiment is cutting slice of ABC flow. It contains 262144 triangles and 522242 vertices. The image restitution is 512×512

Chapter 5

Stability Analysis of Vector Field Topology Via Image-Space Morse Decomposition

5.1 The Stability of MCGs

The definition of an outer approximation and the fact that the triangles in the strongly connected components of $\mathcal{F}$ form isolating neighborhoods for the Morse sets demonstrate why the MCG remains constant under small perturbations of the vector field. Since f_τ is a continuous map and each triangle T is compact, the image $f_\tau(T)$ is a compact set. If $\mathcal{F}$ is an outer approximation, then by definition, $f_\tau(T)$ is contained in the interior of the set $|f_\tau(T)|$. Thus, this property will also hold for any sufficiently small perturbation of $f_\tau(T)$, which means that given a multi-valued map for $f_\tau(T)$

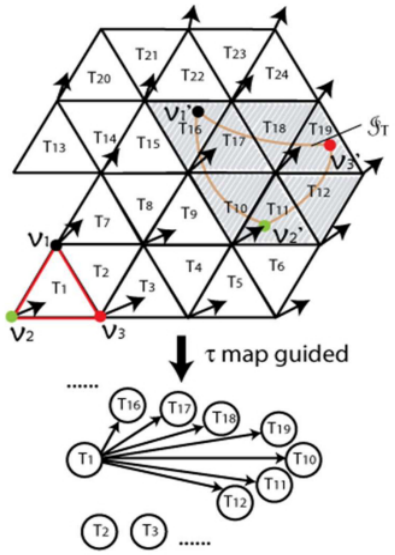

Figure 5.1: Compute flow combinatorialization

we have the same $\mathcal{F}$, for any sufficiently small perturbation of $f_\tau(T)$. Figure 5.1 provides an illustrative example of this property of an outer approximation. In this figure, a triangle $T = T_1$ is advected according to the original flow (represented by the black arrows). Its image ϕ_T is shown as the closure bounded by a blue curve. It intersects with a set of triangles (the shaded triangles) $T_4; T_5; T_6; T_7; T_8; T_9; T_{10}$ of the mesh, which form the outer approximation of ϕ_T .

When we artificially introduce a random perturbation for each vector value and advect the triangle T under the new flow, we obtain a new image $\phi_T{}'$ of it. If we bound the perturbation of each vector to guarantee that the new obtained image $\phi_T{}'$ will intersect the same set of triangles as ϕ_T obtained under the original vector field[4], we will obtain the same outer approximation of the image of T. Hence, the corresponding portion of the directed graph ϕ_T will not change. The MCG is

consequently stable. In other words, the outer approximation provides more space for error in the given data.

However, the obtained Morse decomposition and its corresponding visualization does not provide the information of the livelihood of the locations of the true Morse sets that are either fixed points or periodic orbits by previous mathematical theory. That is, all the points within a Morse neighborhood (i.e., a triangle strip) have the same likelihood to be on the true Morse sets (i.e., having the same color). In addition, given different amounts of uncertainty and error introduced in different stages of the data processing pipeline, the points within a Morse neighborhood also have different probabilities (or stability) to be on the true Morse sets. That said, the information conveyed in the resulting visualization may not be complete and sufficiently accurate. The introduced image-space Morse decomposition pipeline provides us an opportunity to study such a likelihood and stability information due to its pixel-level accuracy property. In the following, we will describe how we model and analyze this probability information.

5.2 The Experiments of Stability Analysis of MCGs

We now describe our experiments on the stability and likelihood analysis of MCGs using the image-space Morse decomposition framework. To study how this affects the stability of the MCGs, we alter the step sizes of an elected Runge-Kutta integrator during the streamline computation. In this experiment, an RK4 integrator is employed.

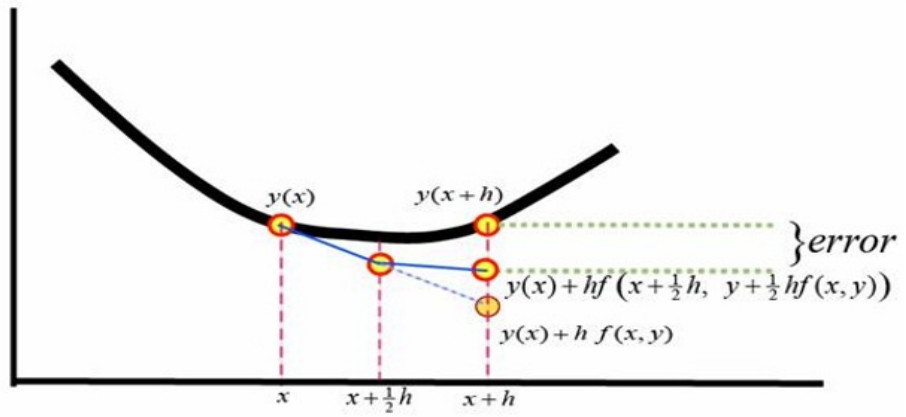

Figure 5.2: Integration error

- Integration step size : As we described in section 3.2. We use the numerical integration method to trace the stream line. The solution is an approximate result. At each single step, an integration error will be generated. Figure 5.2 demonstrates how the error arises. The larger step size we use, the larger the error is. Even if we apply the RungeKutta method for tracing, the error still can not be eliminated.Since the computation of Morse decompositions (both object-space and image-space) relies on streamline integration, such error will naturally be encoded in the resulting MCG. To study how this affects the stability of the MCGs, we alter the step sizes of an elected Runge-Kutta integrator during the streamline computation. In this experiment, an RK4 integrator is employed.

- Integration time/steps : Previous work[4] has shown that the larger the integration time (or length), the finer the obtained MCG will be. This property still holds for the image-space implementation. In the meantime, the longer we

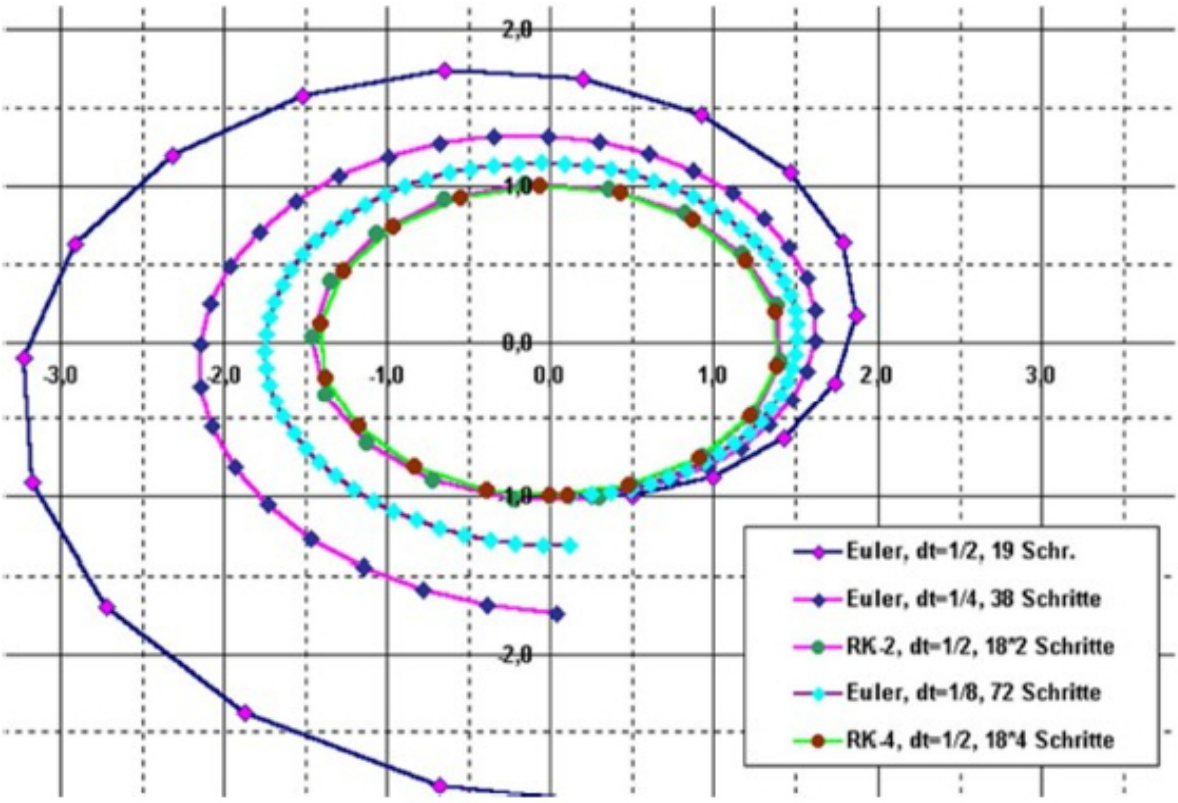

Figure 5.3: Integration accumulative error

trace out a streamline, the more error it may accumulate based on the illustration of Figure 5.3. How much this accumulated error affects the accuracy and stability of the MCGs is not known. Since this error has intrinsic relation with the error introduced by varying step sizes, we study them together in our experiments.

- Amount of flow perturbation/noise : As we mentioned in section 5.1, the outer approximation provides more space to tolerate error in given data. In order to analyse the the stability of MCG under different amount of noise, we artificially

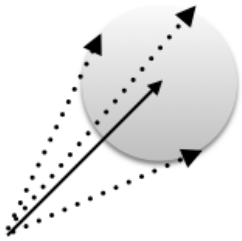

Figure 5.4: Add perturbation on the original vector

insert different amount of perturbation into the original data to simulate the noise. The perturbation that is added to each vector value is illustrated in Figure 5.4. Specifically, given a vector V, a random offset vector V_{off} is generated so that its length $|V_{off}| < r$, and its angle $d\theta = atan2(Vx_{off}, Vy_{off}) \in [0, 2\pi)$. For a given vector field, the largest amount of perturbation is bounded by a percentage p so that $r \leq p \times |V_{max}|$ where $|V_{max}|$ is the largest magnitude among the vector values in the vector field.

5.3 Results and Discussions

According to the previous description, , we have two types of parameters, the first two parameters we discussed previously belongs to the same category which is numerical integration error, and the second type is noise. Figure 5.5, Figure 5.9, Figure 5.10 show comparisons of the obtained MCGs with different parameters.

For each set of parameters, a unique Morse decomposition was computed. The results shown in these figures were obtained by combining a set of resulting MCGs

with different sets of parameters. For instance, Figure 5.5 shows the ensemble of the resulting MCGs using the following permeate: perturbation is 0.2, 0.5, 0.3 of the original magnitude and trace each streamline for 20 steps. To combine different resulting MCGs together for the ensemble visualization, we assign a counter to each pixel to see how many times it is identified as part of a Morse set. Once the pixel is marked within a Morse neighborhood for one execution, we increase the counter value by one. After that, we highlight the Morse decomposition with different colors based on their counter values. The low occupancy area will be assigned a weak color, the high occupancy area will be assigned a strong color. In this color scheme, the strong color area contains less error space. It is very close to the actual Morse range. And the weak color area can be considered as error space. In order to visualize the color contrast, we set the background black.

As we compare these results, it is easy to see if one uses a larger number of integration step, the Morse sets are small, thus, the corresponding Morse neighborhoods do not include many regions with weak color. That means the result is more stable. This result proves our theory, the less integration error we have, the more accurate the Morse Decomposition results are.

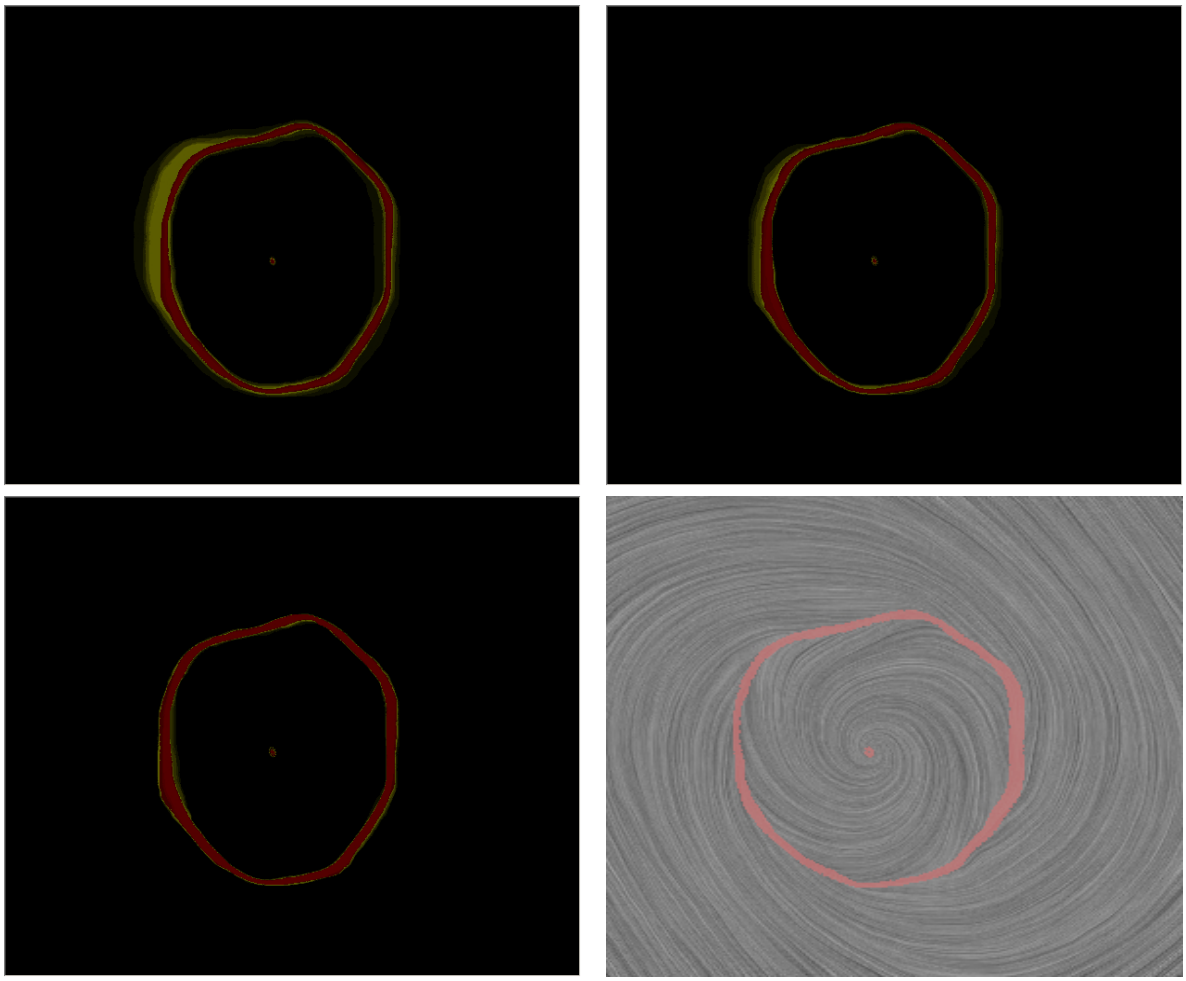

Figure 5.5: Comparison of the Stability Analysis of Image-Space Morse Decomposition results in different perturbation, the number of integration steps is 20.

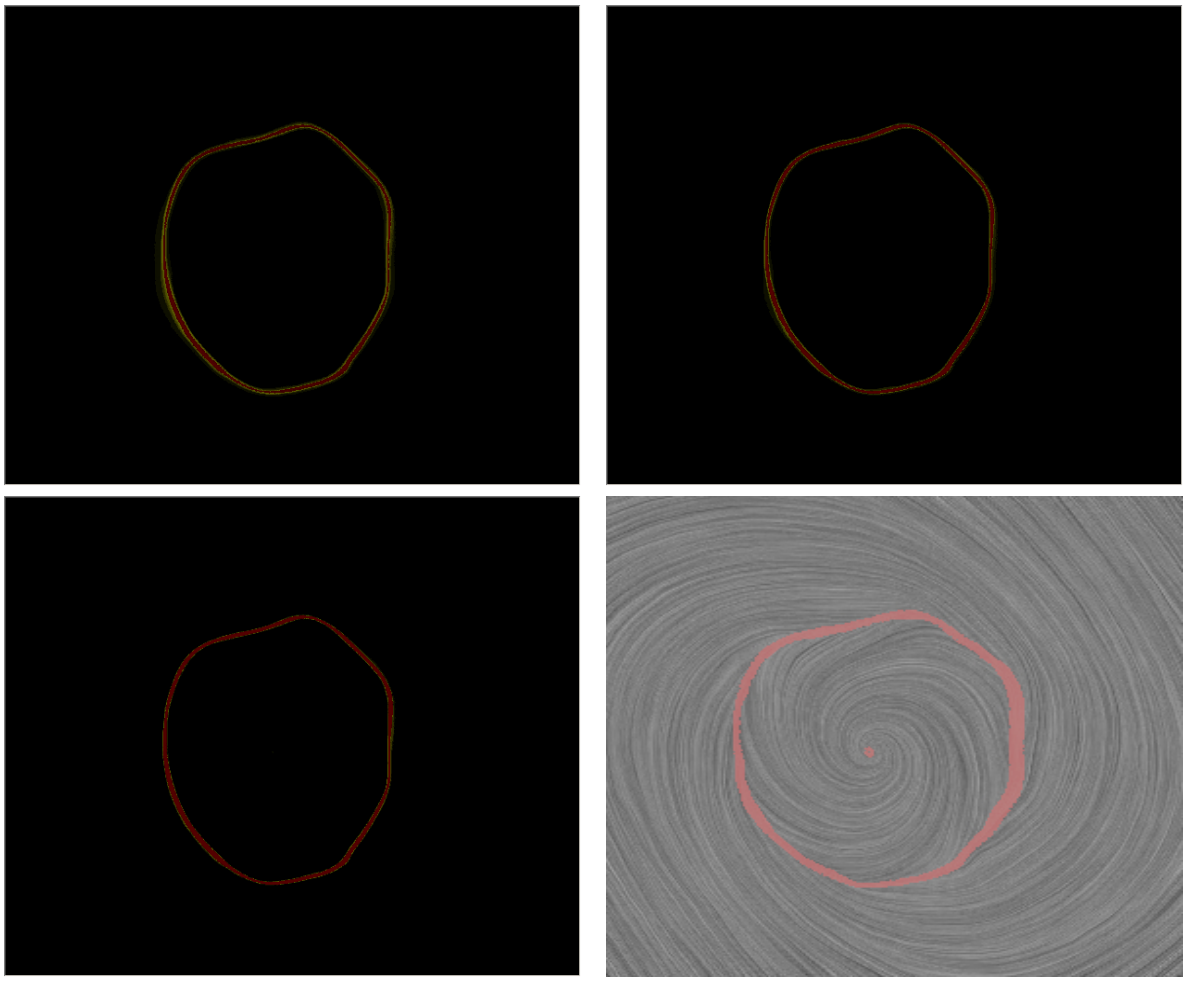

Figure 5.6: Comparison of the Stability Analysis of Image-Space Morse Decomposition results in different perturbation, the number of integration steps is 40.

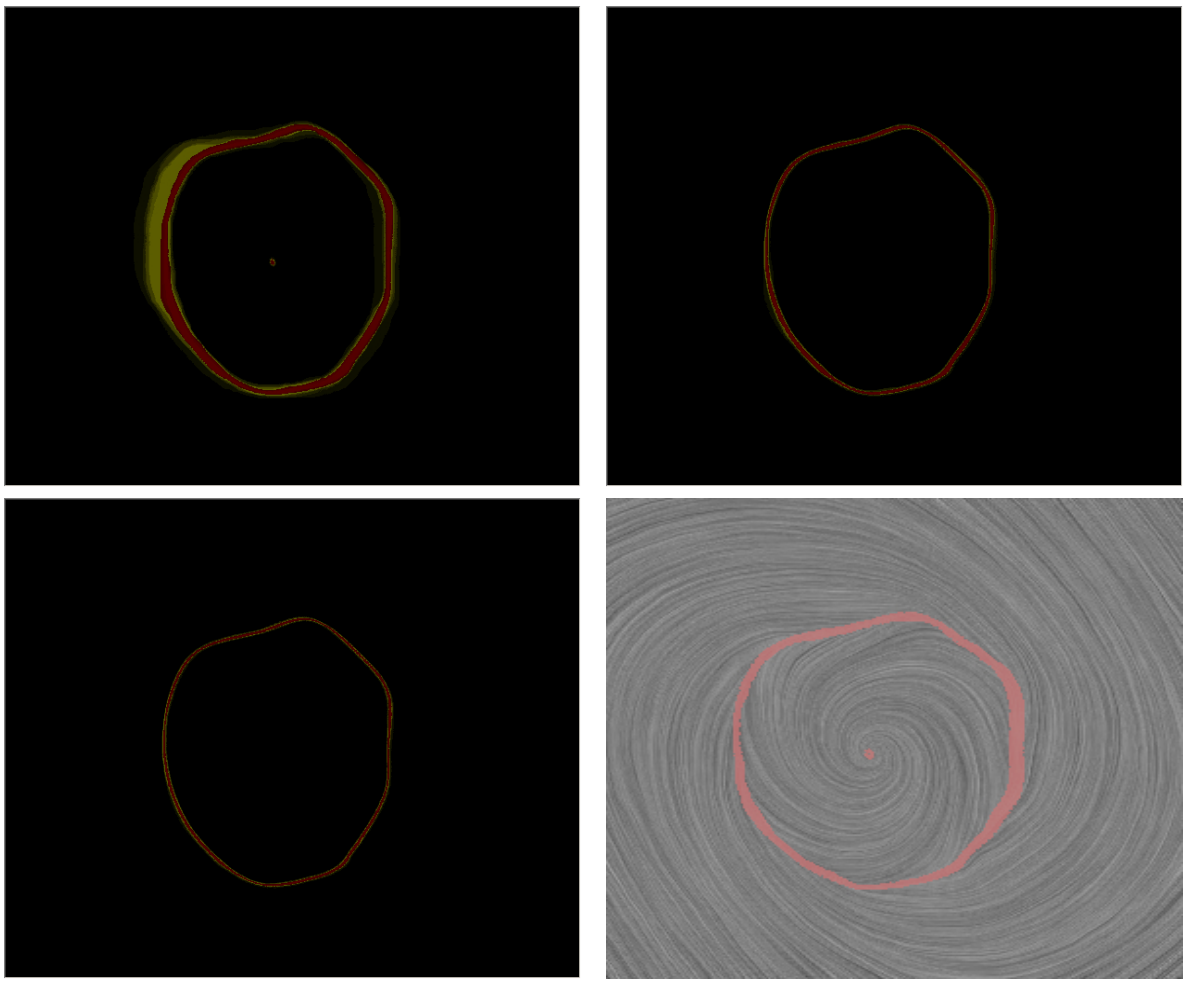

Figure 5.7: Comparison of the Stability Analysis of Image-Space Morse Decomposition results in same perturbation, different steps (20, 40, 60).

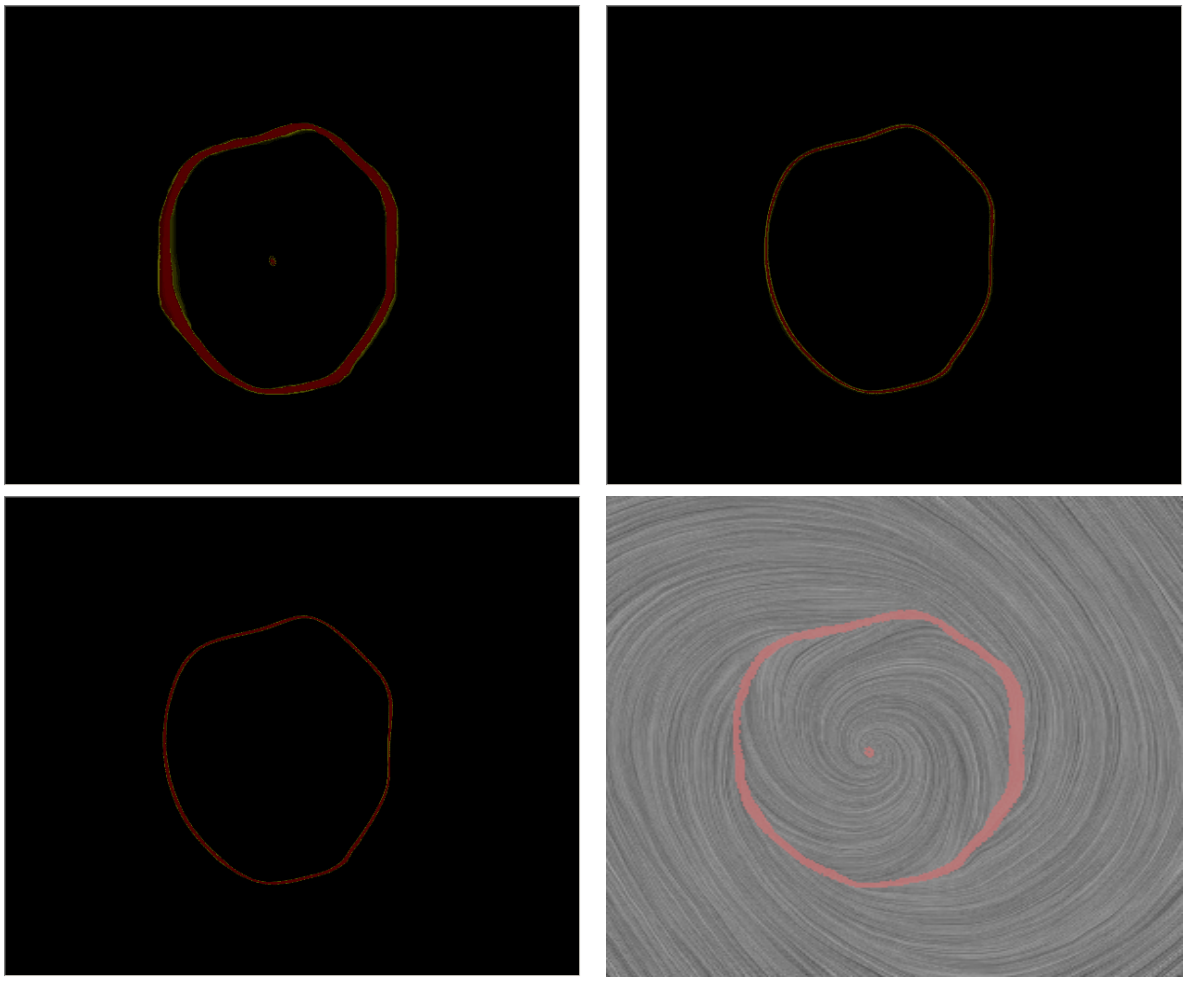

Figure 5.8: Comparison of the Stability Analysis of Image-Space Morse Decomposition results in different perturbation, the number of integration steps is 60.

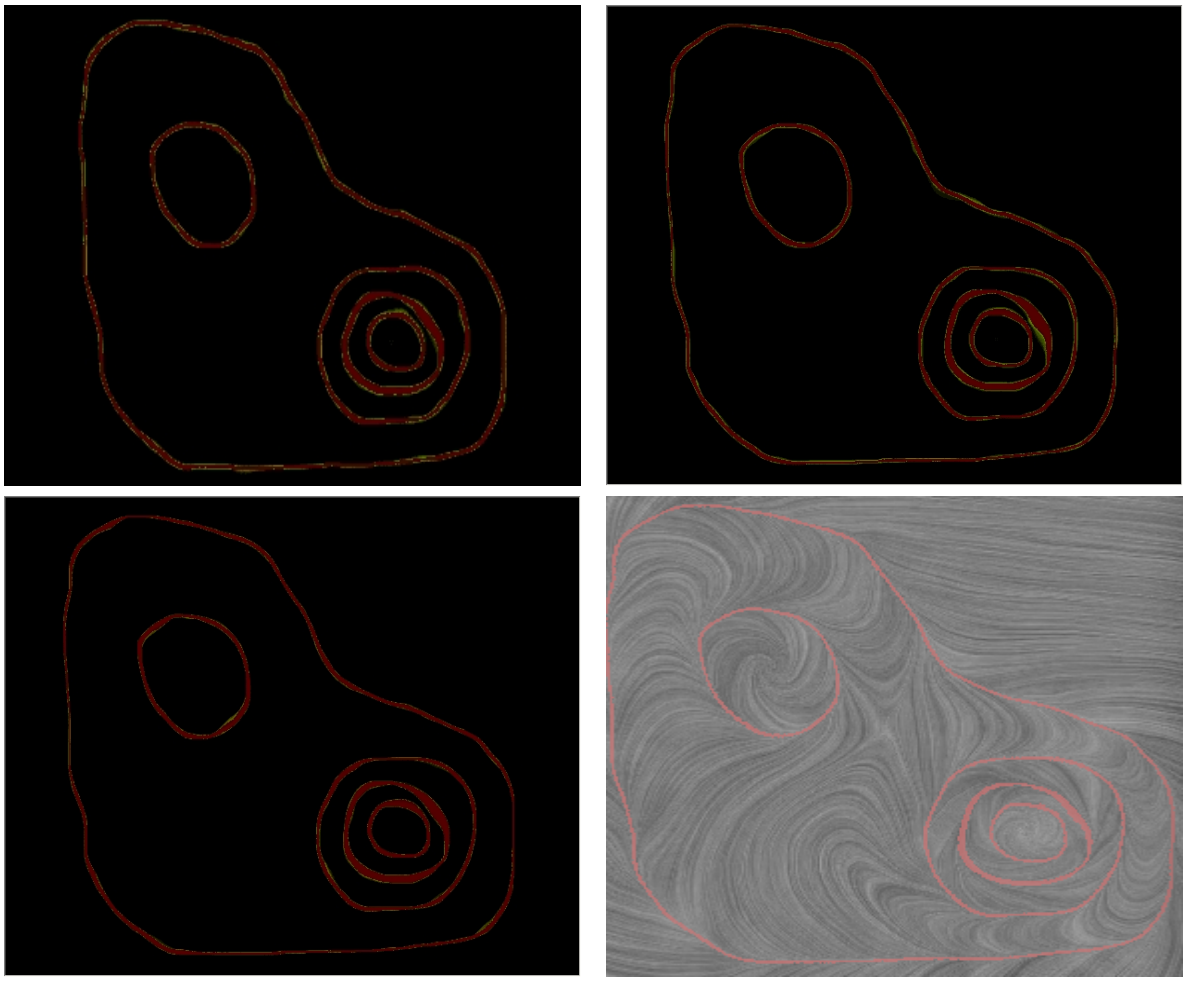

Figure 5.9: Comparison of the Stability Analysis of Image-Space Morse Decomposition results in different perturbation, the number of integration steps is 20.

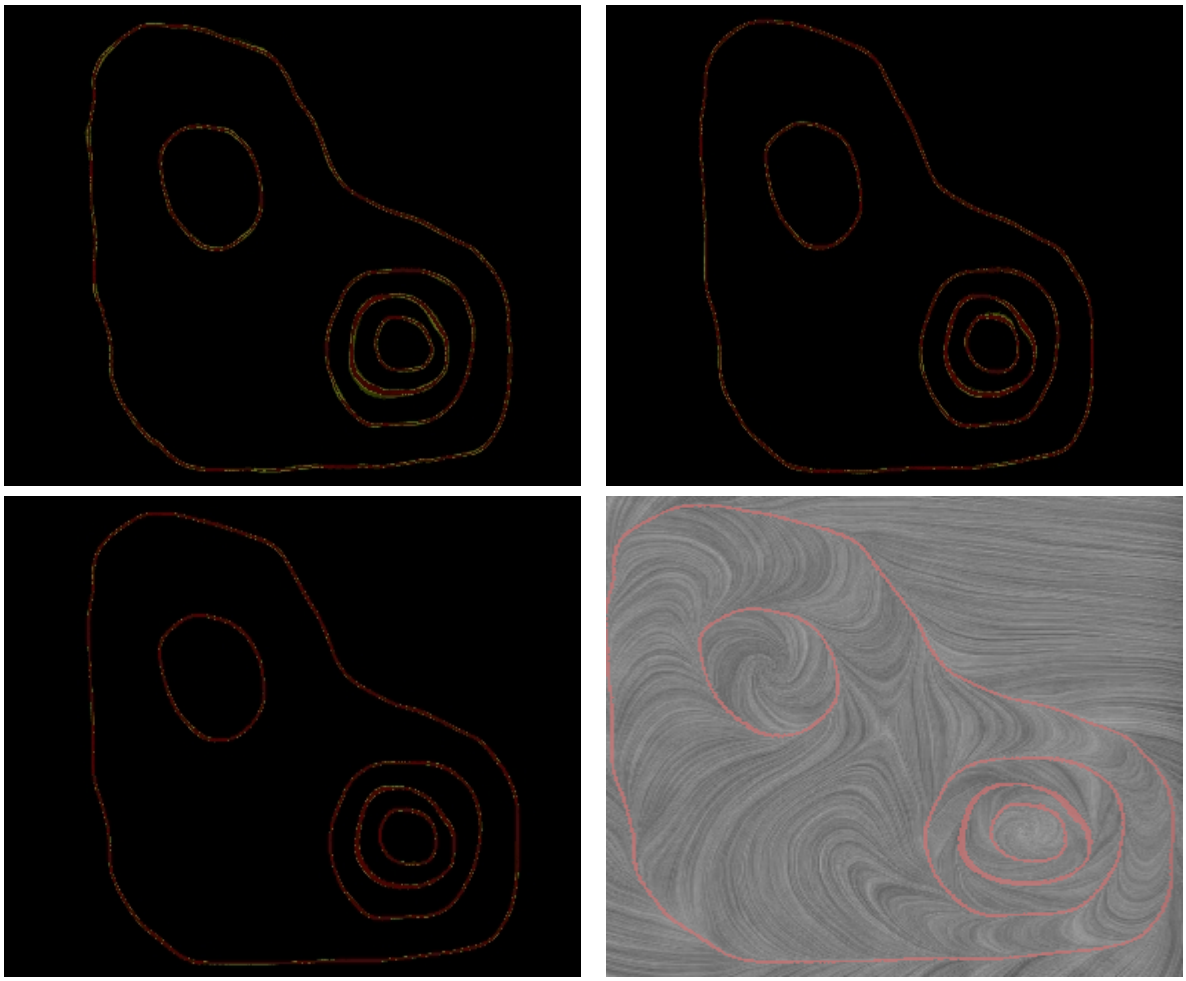

Figure 5.10: Comparison of the Stability Analysis of Image-Space Morse Decomposition results in different perturbation, the number of integration steps is 40.

Chapter 6

Analysis and Visualization Framework for 3D Vector Fields

In this chapter, we describe four new approaches to flow visualization instead of visualize the streamlines directly in a 3D flow.

- Visualize the cutting plane by arrow plot.
- Visualize the cutting plane by Line Integral Convolution method.
- Visualize the cutting plane by Structure-accentuating Dense Flow Visualization.
- Demonstrate the utility of Morse Decompositions in the plane.
- Demonstrate the utility of Image-Space Morse Decompositions in the plane.

The flow data we use to demonstrate our result is the ABC flow. It can be described as

$$U = \begin{pmatrix} Asin(z) + Ccos(y) \\ Bsin(x) + Acos(z) \\ Csin(y) + Bcos(x) \end{pmatrix}$$

Normally, we set

$$A = B = C = 1$$

6.1 Dimensionality Reduction

Streamlines are a very simple way of conveying the structure of 3D vector fields. By following the particle trajectories, they show the flow behavior in an intuitive fashion. However, two major issues arise because of the aforementioned simplicity of the techniques:

- The first issue is the ability of the final pictures to convey the actual vector field structure carried by the data. In order to properly present the vector field information, it is necessary to choose the streamlines which best depict the behavior of the field. However, this means we will lose some features of the flow data.

- The second issue is that of cluttered pictures; it is caused by the occlusion between streamlines and is especially relevant when visualizing 3D fields. It is difficult to choose streamlines streamlines accurately depict the flow - too many lines result in unreadable pictures, while too few lines does not actually

convey the vector field structure. Although this is not a major issue in the case of 2D flows (because occlusion will never completely occlude information in this case, at worse it can drown it).

In order to address these two challenges, we design a 3D visualization system to visualize the entire 3D vector field in an intuitive way. As we described before, because occlusion will never completely occlude information in 2D case, the cluttered pictures is not a major issue in the case of 2D flows. We, then, reduce the dimensionality of the 3D vector field by utilizing the cutting plane technique. Dimensionality reduction involves mapping a set of high dimensional input points onto a low dimensional manifold so that similar points in input space are mapped to nearby points on the manifold. In our case, we produce a function(or mapping) from input which are all 3D vectors that have intersection with the cutting slice to the cutting plane. The result is a 2D vector file that has similar relationship with the original data. Figure 6.1 illustrates the projection of a 3D vector onto a cutting plane. The black arrow line e_1, e_2,are the original 3D vectors which have intersection with the cutting plane P. The red arrow lines $e_1{}'$, $e_2{}'$ are the output of our mapping function.

6.2 Structure-Accentuating Dense Flow Visualization

The Structure-accentuating Dense Flow Visualization method displays the local or integrated flow behavior for the entire domain. It shows the overall flow behaviorit

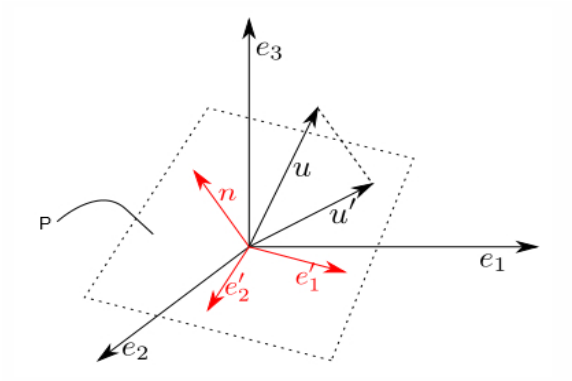

Figure 6.1: This figure illustrates 3D vector projection

also accentuates the topological structure of the vector field. This method is based on a very simple principle that automatically generates an image showing the basic field structure and provides an overview of the flow behavior at the same time. After seeding the entire domain with particles, we trace these particles moving with the flow. Instead of displaying the particle traces themselves, we use them to generate a density field, which serves as the basis for rendering. The density is generated by counting how often a cell is hit in a discretized domain. A forward and backward advection of the particles lead to an accumulation of particles in attracting and repelling points or lines, respectively, resulting in a high high density value near those regions. Such regions are, for example, sources, sinks, and attracting/repelling separatrices.

The first step of our algorithm is the flow integration step. We use a higher-order integration method to precisely trace the path of a (mass-less) particle moving under the influence of the flow field. Instead of storing the coordinates of the integration points along the streamline, we increase the counter of each cell the streamline passes

through. Starting with all counters initially being zero and running an animation with many streamlines moving over time leads to an accumulated intensity value for each cell. We store the result in a texture representing the density field mentioned above. The seeding method that we used is to start from the center of the each cell. The resolution of the chosen grid determines the accuracy of the density field. Thus, resolution governs the quality of our entire visualization method.

The second step is the visualization of the density field using texture-based rendering methods. Using texture-based methods for rendering supports a dense visualization, which is important if we want to capture the entire flow behavior. We use simple linear transfer functions to extract attracting and repelling regions and explore their interplay by blending the two density fields for forward and backward propagation. Unfortunately, the blended density field provides just one scalar value at each grid point, such that no directional information of flow is conveyed. One way to address this issue is to leave the individual density fields for forward and backward integration separately and blend them by applying a multi-dimensional transfer function. Figure 6.2 is an example of structure-accentuating Dense Flow Visualization method. The flow topology information is showing by the color. The brightness area means there are more particle more thought this region. As we can see, the most brightness area is the critical points and the color represents the move direction. In our color scheme green means more forward, red means move backward.

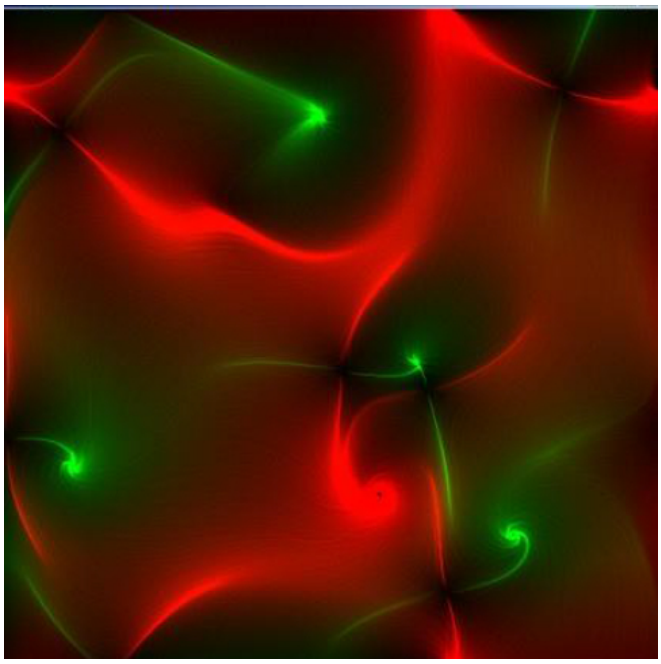

Figure 6.2: The result of the Structure-Accentuating Dense Flow Visualization method on a 2D flow.

6.3 Combine the Accentuating Method With the Volume Rendering

In order to visualize the entire vector filed, a direct 3D visualization is necessary since dimensionality reduction leads to information lost anyway. Our 3D visualization approach combines the accentuating method with volume rendering. Volume rendering techniques have been developed to overcome problems of accurate representation of surfaces in iso-surface techniques. In short, these problems are related to making a decision for every volume element regardless if the surface passes through it and this can produce false positives (spurious surfaces) or false negatives (erroneous holes in surfaces), particularly in the presence of small or poorly defined features.

Our combined volume rendering involves the following steps: 1) forming an RGBA volumetric image from the data, 2)reconstruction of a continuous function from this discrete data set, 3)projecting it onto the 2D viewing plane (the output image) from the desired point of view. An RGBA volumetric image assigns a 4-tuple to each voxel, where the first three components are the R, G, and B color components and the last component, A, represents opacity. An opacity value of 0 means completely transparent and a value of 1 means completely opaque. Behind the RGBA volumetric image an opaque background is placed. The mapping of the data to opacity values acts as a classification of the data one is interested in. Isosurfaces can be shown by mapping the corresponding data values to almost opaque values and the rest to transparent values. The appearance of surfaces can be improved by using shading techniques to form the RGB mapping. However, opacity can be used to see the interior of the data volume too. These interiors appear as clouds with varying density and color. A big advantage of volume rendering is that this interior information is not thrown away, so that it enables one to look at the 3D data set as a whole. Disadvantages are the difficult interpretation of the cloudy interiors and the expensive rendering cost in comparison to surface rendering.

The algorithm that we used for computing the data set is very similar with Structure-accentuating Dense Flow Visualization method. In 2D case, we used a counter for each pixel. And in 3D case, we set a counter for each voxel to count how many streamlines go through this voxel, and use the same color scheme to compute the RGB value based on the counters.

Figure 6.3 is the visualization results of the Combined Accentuating Volume

Rendering Method of ABC flow. Figure 6.4 illustrates the different visualization options for the 2D projected plane of our system.

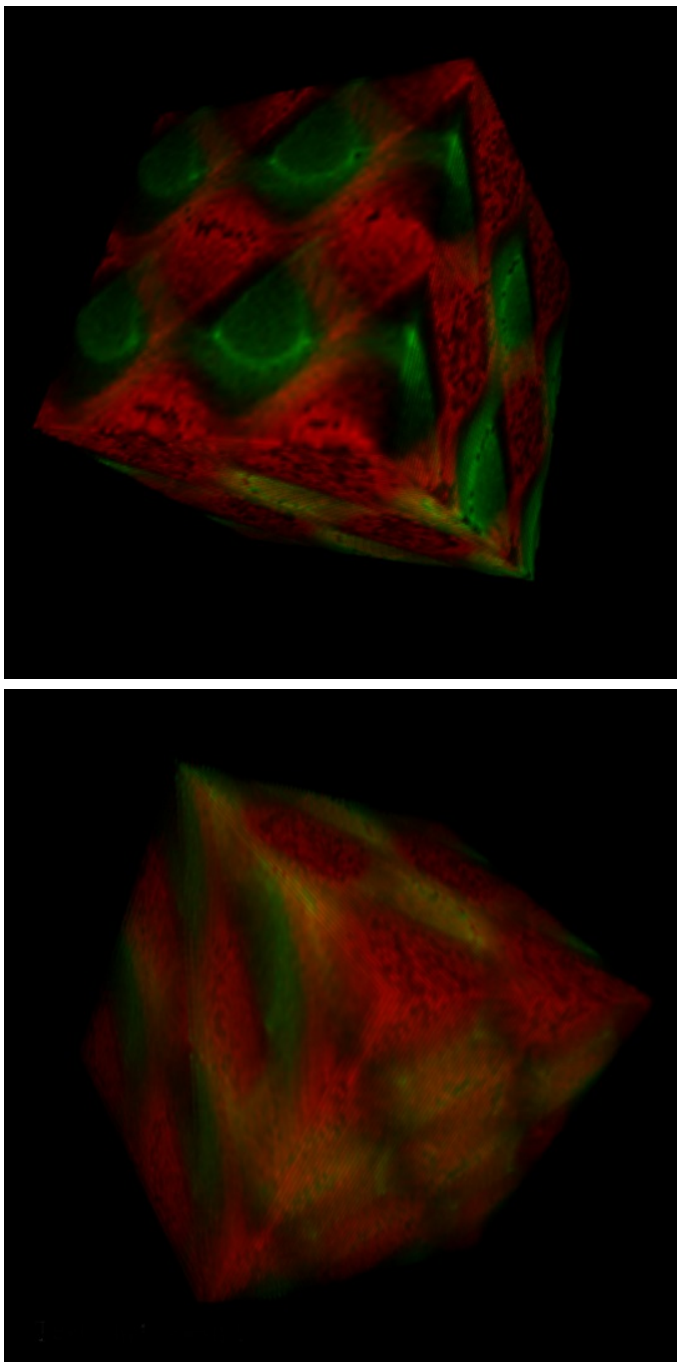

Figure 6.3: The volume rendering of the density field computed using the Structure-Accentuating Dense Flow Visualization method for the ABC flow.

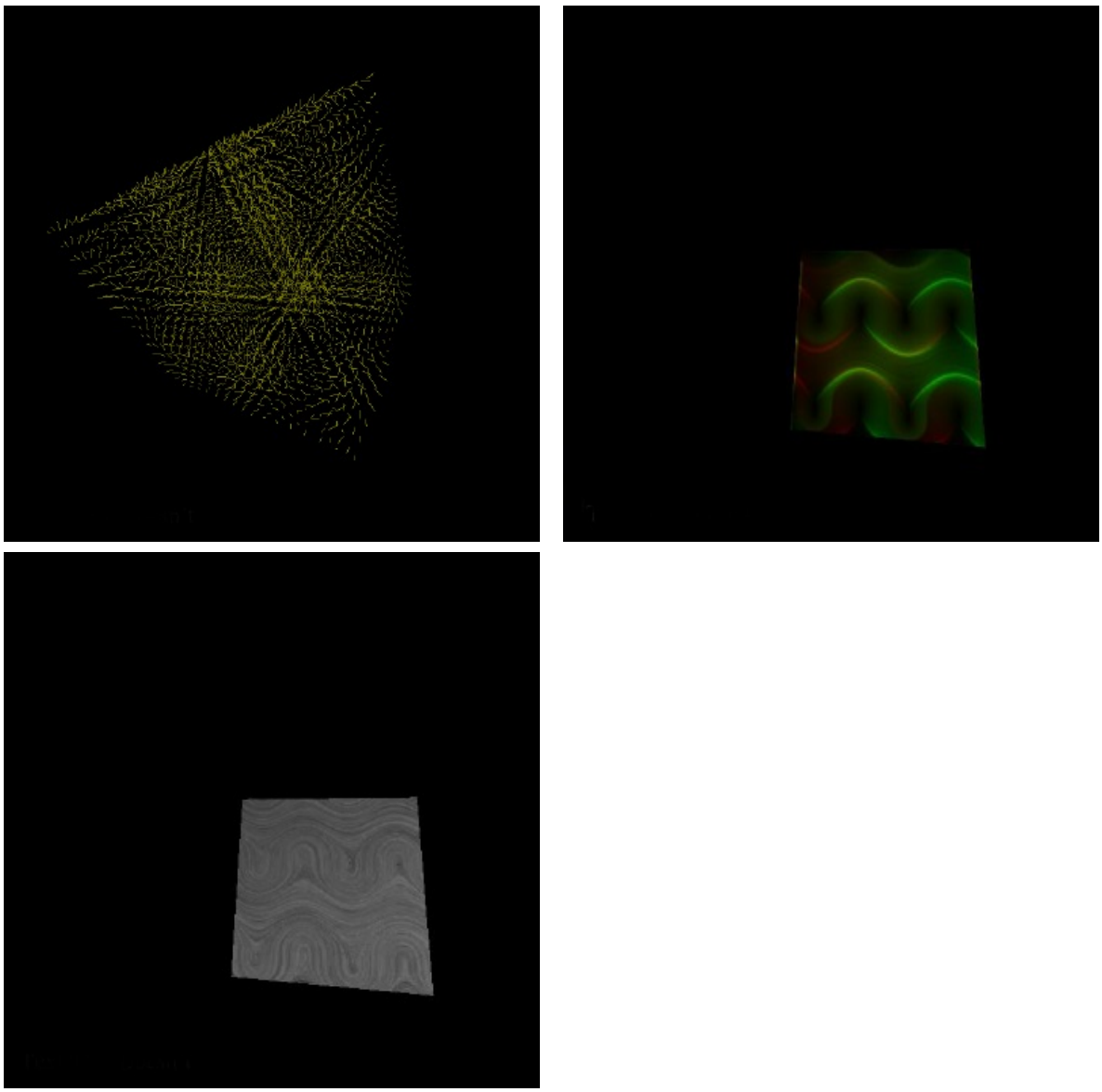

Figure 6.4: Different views of our 3D flow visualization system. The upper-left image shows the arrow plot of the original ABC flow in 3D, while the upper-right image displays the density field of a 2D slice of the ABC flow using the accentuating method. The bottom left image is the LIC result of the corresponding 2D projected flow.

Chapter 7

Conclusion

In this thesis we have demonstrated the fundamental difficulties associated with the Morse Decomposition computation based on individual trajectories. As a solution, we advocate the use of an Image-Space Morse decomposition method. We generate a pixel based input data to encode the original triangle mesh input data, in order to increase the sample resolution of the data. This stage follows the Image-Space LIC pipeline. Then, we compute the flow combinatorialization based on the flow in the obtained image plane by the same algorithm as the original Morse Decomposition. Our results are more realistic in terms of visualizing the Morse sets. We also show the comparisons between different visualization results based on various analysis results. We also perform experiments for the study of the stability and likelihood analysis of MCGs using the image-space Morse decomposition framework. We set the experiments with different parameters, such as integration time/steps, integration step size, amount of flow perturbation/noise to study how these parameters affect

the Morse set detection. The difference between the Morse sets detected in different trials provide the needed stability and likelihood information. We then develop proper visualization technique to visualize this information.

We have developed a 3D visualization system instead of visualizing the 3D streamlines directly. It provides 2D cutting plane visualization and 3D volume rendering visualization to visualize the complete flow information. Some classic techniques are also applied on the 2D cutting plane mode, such as LIC, arrow plot and Structure-Accentuating dense method.

There are a number of future directions based on this work. First, we can extend our Image-Space Morse decomposition to by replacing pixels with voxels in the computation. Second, the current method does not support the vector field defined on 3D curved surfaces, which we plan to address in the future. Finally, developing more rigorous uncertainty model for the Morse set detection and visualization is needed.

Bibliography

[1] G. V. Bancroft, F. J. Merritt, T. C. Plessel, P. G. Kelaita, R. K. McCabe, and A. Globus. Fast: a multi-processed environment for visualization of computational fluid dynamics. In *Proceedings of the 1st conference on Visualization'90*, pages 14–27. IEEE Computer Society Press, 1990.

[2] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 263–270. ACM, 1993.

[3] G. Chen, Q. Deng, A. Szymczak, R. S. Laramee, and E. Zhang. Morse set classification and hierarchical refinement using conley index. *Visualization and Computer Graphics, IEEE Transactions on*, 18(5):767–782, 2012.

[4] G. Chen, K. Mischaikow, R. S. Laramee, and E. Zhang. Efficient morse decompositions of vector fields. *Visualization and Computer Graphics, IEEE Transactions on*, 14(4):848–862, 2008.

[5] J. L. Helman and L. Hesselink. Representation and Display of Vector Field Topology in Fluid Flow Data Sets. *IEEE Computer*, 22(8):27–36, August 1989.

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

[7] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of the conference on Visualization'01*, pages 255–262. IEEE Computer Society, 2001.

[8] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. In *Computer Graphics Forum*, volume 23, pages 203–221. Wiley Online Library, 2004.

[9] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *Proceedings of the 4th conference on Visualization'93*, pages 19–24. IEEE Computer Society, 1993.

[10] S. W. Park, B. Budge, L. Linsen, B. Hamann, and K. I. Joy. Dense geometric flow visualization. In *Proceedings of the Seventh Joint Eurographics/IEEE VGTC conference on Visualization*, pages 21–28. Eurographics Association, 2005.

[11] K. Polthier and E. Preuss. Identifying vector field singularities using a discrete hodge decomposition. In *Visualization and Mathematics III*, pages 113–134. Springer, 2003.

[12] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. In *Computer Graphics Forum*, volume 22, pages 775–792. Wiley Online Library, 2003.

[13] G. Scheuermann, H. Kruger, M. Menzel, and A. P. Rockwood. Visualizing nonlinear vector field topology. *Visualization and Computer Graphics, IEEE Transactions on*, 4(2):109–116, 1998.

[14] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Grid-independent detection of closed stream lines in 2d vector fields. In *VMV*, pages 421–428, 2004.

[15] X. Tricoche, G. Scheuermann, and H. Hagen. Continuous topology simplification of planar vector fields. In *Proceedings of the conference on Visualization'01*, pages 159–166. IEEE Computer Society, 2001.